

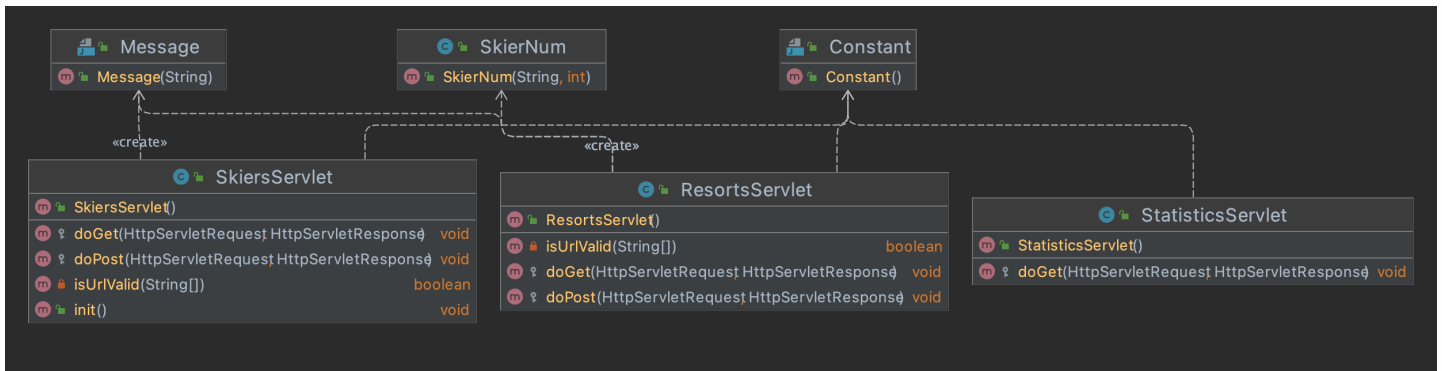
CS6650 Assignment2 Report

URL to Repo

Please visit [Github Repository](#).

Design

Server



I adopt the server design from previous project, except adding a `BlockingQueue` to store the channels. Each time a channel is taken from the queue, a new channel will be added in to the queue.

Since we only test the `POST`

`/skiers/{resortID}/seasons/{seasonID}/days/{dayID}/skiers/{skierID}` from <https://app.swaggerhub.com/apis/cloud-perf/SkiDataAPI/1.16#/>, the main changes happen in `SkiersServlet` class.

In this class, I firstly use `ConnectionFactory` to build a connection with my rabbitmq which is located at `ec2-54-203-208-182.us-west-2.compute.amazonaws.com`. Since I have the private ipv4 address of this ec2 instance, I configured all needed information in `rabbitmy.conf` inside resource folder like this:

```
172.31.5.13
5672
admin
fuckjava
```

Once a skier post is received by Server, the configuration file will be automatically loaded and connect to Rabbitmq. Each post will be handled in `doPost` function. The primary function is listed below.

```
channel = channelPool.take();
channel.basicPublish("", Constant.QUEUE_NAME, null,
msg.toString().getBytes());
res.setStatus(HttpServletResponse.SC_CREATED);
System.out.println("Sent " + msg + " to rabbitmq");
res.getWriter().write("Sent " + msg + " to rabbitmq");
channelPool.add(channel);
```

Others

For detailed design, please see the codes.

For your convinence, I build a docker image based on the `Dockerfile`. You can also `docker pull zjdx1998/consumer:mar11-amd64-latest` then `docker run -it --rm --name consumer zjdx1998/consumer:mar11-amd64-latest` to start the consumer service.

Or you can build manually to get a consumer run in the local environment.

Here is a output window for remote Consumer.

ec2-user@ip-172-31-5-13:~

```
ec2-user@ip-172-31-5-13:~ (ssh)
cat Se... 23 - thread received {"time":203,"liftID":34,"waitTime":8,"skierID":14832}
.servi... 23 - thread received {"time":114,"liftID":5,"waitTime":9,"skierID":12105}
ice fa... 23 - thread received {"time":264,"liftID":25,"waitTime":7,"skierID":14071}
mcat S... 23 - thread received {"time":234,"liftID":38,"waitTime":5,"skierID":7317}
sting ... 23 - thread received {"time":111,"liftID":28,"waitTime":3,"skierID":19359}
oving/... 23 - thread received {"time":272,"liftID":27,"waitTime":10,"skierID":14455}
cat Se... 23 - thread received {"time":316,"liftID":25,"waitTime":4,"skierID":7976}
23 - thread received {"time":104,"liftID":3,"waitTime":7,"skierID":11205}
23 - thread received {"time":198,"liftID":28,"waitTime":9,"skierID":7183}
24 - thread received {"time":308,"liftID":26,"waitTime":10,"skierID":9418}
24 - thread received {"time":105,"liftID":31,"waitTime":3,"skierID":19498}
24 - thread received {"time":317,"liftID":1,"waitTime":9,"skierID":10821}
23 - thread received {"time":247,"liftID":5,"waitTime":0,"skierID":4516}
24 - thread received {"time":106,"liftID":25,"waitTime":9,"skierID":4224}
23 - thread received {"time":230,"liftID":38,"waitTime":1,"skierID":13207}
23 - thread received {"time":110,"liftID":37,"waitTime":8,"skierID":19781}
23 - thread received {"time":330,"liftID":8,"waitTime":3,"skierID":5426}
23 - thread received {"time":180,"liftID":8,"waitTime":1,"skierID":2421}
23 - thread received {"time":202,"liftID":26,"waitTime":9,"skierID":3049}
23 - thread received {"time":149,"liftID":40,"waitTime":1,"skierID":13649}
24 - thread received {"time":293,"liftID":35,"waitTime":1,"skierID":7730}
```

For Clients, there is no change since last project. But the parameter should be changed from single instance public url to load balancing url.

```
java Client -nt 128 -ns 20000 -nl 40 -nr 10 -server CS6650-LB-
987462913.us-west-2.elb.amazonaws.com ec2-54-149-212-65.us-
west-2.compute.amazonaws.com localhost:8080/Server_war
```

Load Balancing:

| Targets | Monitoring | Health checks | Attributes | Tags |
|--|-------------------------------------|---------------|------------|------------|
| Registered targets (4) <input type="text" value="Filter resources by property or value"/> | | | | |
| <input type="checkbox"/> | Instance ID | Name | Port | Zone |
| <input type="checkbox"/> | i-05ba806c2981f3f15 | | 80 | us-west-2c |
| <input type="checkbox"/> | i-095e40c72251d7146 | | 80 | us-west-2b |
| <input type="checkbox"/> | i-0e5d07d98f3df02e0 | | 80 | us-west-2b |
| <input type="checkbox"/> | i-080f3c1442a187469 | | 80 | us-west-2b |

| | | | | | | |
|-----------|----------------------------|--------|-----------------------|---------------------------|-------------|--------------------------------|
| CS6650-LB | CS6650-LB-987462913.us-... | Active | vpc-0603d1ab1d063aaed | us-west-2b, us-west-2d... | application | March 11, 2022 at 1:56:52 P... |
|-----------|----------------------------|--------|-----------------------|---------------------------|-------------|--------------------------------|

| | |
|---------------------------|--|
| Name | CS6650-LB |
| ARN | arn:aws:elasticloadbalancing:us-west-2:860492115967:loadbalancer/app/CS6650-LB/c3e90bc6ba9b6304 |
| DNS name | CS6650-LB-987462913.us-west-2.elb.amazonaws.com (A Record) |
| State | Active |
| Type | application |
| Scheme | internet-facing |
| IP address type | ipv4 |
| | Edit IP address type |
| VPC | vpc-0603d1ab1d063aaed |
| Availability Zones | subnet-0555059cb885de5e4 - us-west-2b IPv4 address: Assigned by AWS subnet-083f1f1ffd35420ea - us-west-2d IPv4 address: Assigned by AWS subnet-0c0ee495280814f02 - us-west-2a IPv4 address: Assigned by AWS subnet-0db556779e57caeb6 - us-west-2c IPv4 address: Assigned by AWS Edit subnets |
| Hosted zone | Z1H1FL5HABSF5 |
| Creation time | March 11, 2022 at 1:56:52 PM UTC-8 |

Results Analysis

Do we really need load balancing?

64-20000 Without Load Balancing

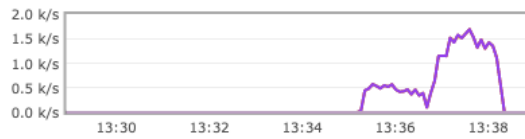
Overview

▼ Totals

Queued messages last ten minutes ?



Message rates last ten minutes ?

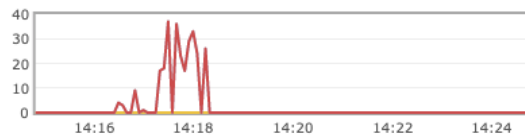


```
/Users/jeromy/.sdkman/candidates/java/11.0.14-librca/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 16 threads with 2500 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 64 threads with 1875 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 6 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 157374
Number of Unsuccessful Requests: 0
Total run time: 178026 (ms)
Total Throughput in requests per second: 883.9944727174683
Mean response time: 44.71920675352064
Median response time: 34.0
Throughput: 22.36175622505362
99th response time: 170.0
min and max response time: min: 12.0 , max: 5049.0

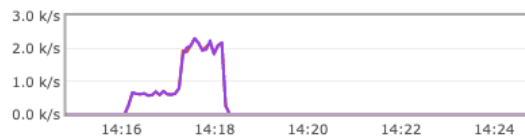
Process finished with exit code 0
```

Here is the result of `numThreads=64 numSkiers=20000` with two instances Load Balanced.

Queued messages last ten minutes ?



Message rates last ten minutes ?



```
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 16 threads with 2500 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 64 threads with 1875 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 6 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 156590
Number of Unsuccessful Requests: 0
Total run time: 122976 (ms)
Total Throughput in requests per second: 1273.3378870673953
Mean response time: 30.33827809835061
Median response time: 24.0
Throughput: 32.96165974740559
99th response time: 110.0
min and max response time: min: 12.0 , max: 3931.0
```

From the chart we know all throughput, total run time, message rates have a significant improvement.

The improvement percentage for each parameter is

$$P_{throughput} = \frac{1277.33}{883.99} = 1.445$$

$$P_{totalRunTime} = \frac{122976}{178026} = 0.6907$$

$$P_{messageRates} = \frac{2.4k/s}{1.6k/s} = 1.5$$

(1)

So we can get the conclusion that we need load balancing!

Here is a result for `numThreads=64, 128, 256 and 512` with 4 instances load balanced.

And the parameter for consumer is `numChannel = 10, basicQos = 0`.

Queued messages last ten minutes ?



64 128 256 512
Message rates last ten minutes ?




```

/Users/jeromy/.sdkman/candidates/java/11.0.14-librca/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 16 threads with 2500 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 64 threads with 1875 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 6 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 157917
Number of Unsuccessful Requests: 0
Total run time: 123584 (ms)
Total Throughput in requests per second: 1277.8110435007768
Mean response time: 30.27768464336549
Median response time: 24.0
Throughput: 33.02762452871779
99th response time: 110.0
min and max response time: min: 12.0 , max: 2503.0

Process finished with exit code 0

```

```

Client x
/Users/jeromy/.sdkman/candidates/java/11.0.14-librca/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: 154897
Number of Unsuccessful Requests: 0
Total run time: 73905 (ms)
Total Throughput in requests per second: 2095.893376632163
Mean response time: 37.73866628801569
Median response time: 26.0
Throughput: 26.498021746930686
99th response time: 193.0
min and max response time: min: 12.0 , max: 3224.0

Process finished with exit code 0

```

```

Client x
/Users/jeromy/.sdkman/candidates/java/11.0.14-librca/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: 151081
Number of Unsuccessful Requests: 0
Total run time: 62999 (ms)
Total Throughput in requests per second: 2398.149176971063
Mean response time: 82.25847340137581
Median response time: 38.0
Throughput: 12.156802316529188
99th response time: 890.0
min and max response time: min: 12.0 , max: 8554.0

Process finished with exit code 0

```

```

Client x
/Users/jeromy/.sdkman/candidates/java/11.0.14-librca/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 128 threads with 312 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 512 threads with 234 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 51 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 140411
Number of Unsuccessful Requests: 0
Total run time: 44195 (ms)
Total Throughput in requests per second: 3177.0788550741036
Mean response time: 123.67218634981882
Median response time: 32.0
Throughput: 8.085892467134062
99th response time: 2079.0
min and max response time: min: 12.0 , max: 9800.0

Process finished with exit code 0

```

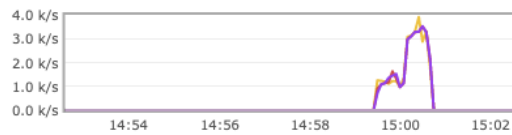
Some analysis on number of threads of consumer

Below are two images with `numThreads=128 & numSkiers=20000 & numChannel = 100 & basicQos = 10` and `numThreads=128 & numSkiers=20000 & numChannel = 100 & basicQos = 100`.

Queued messages last ten minutes ?



Message rates last ten minutes ?



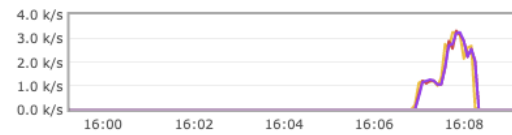
Overview

▼ Totals

Queued messages last ten minutes ?



Message rates last ten minutes ?



From which we can see the queued messages for the one with larger **basicQos** is noticeably smaller than the former.

But if we enlarge the **basicQos** until it become 0, the number of queued message is getting smaller then larger. That's an interesting thing.