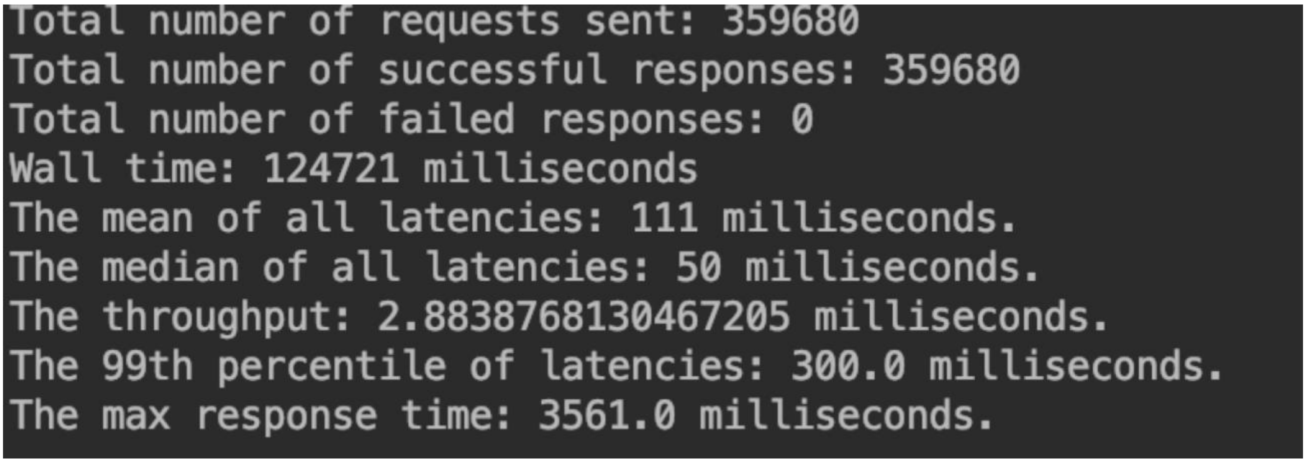
Assignment3

Chenxi Cai

1. Github link:
https://github.com/xiaohaiguicc/DistributedSystem/tree/master/assignment3

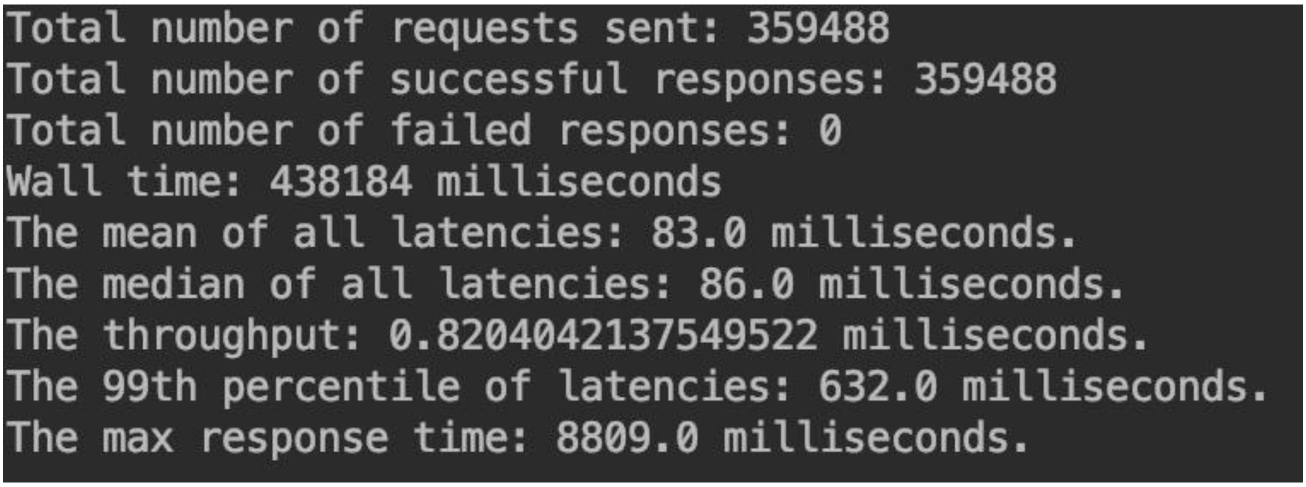
2. Performances

256 Thread



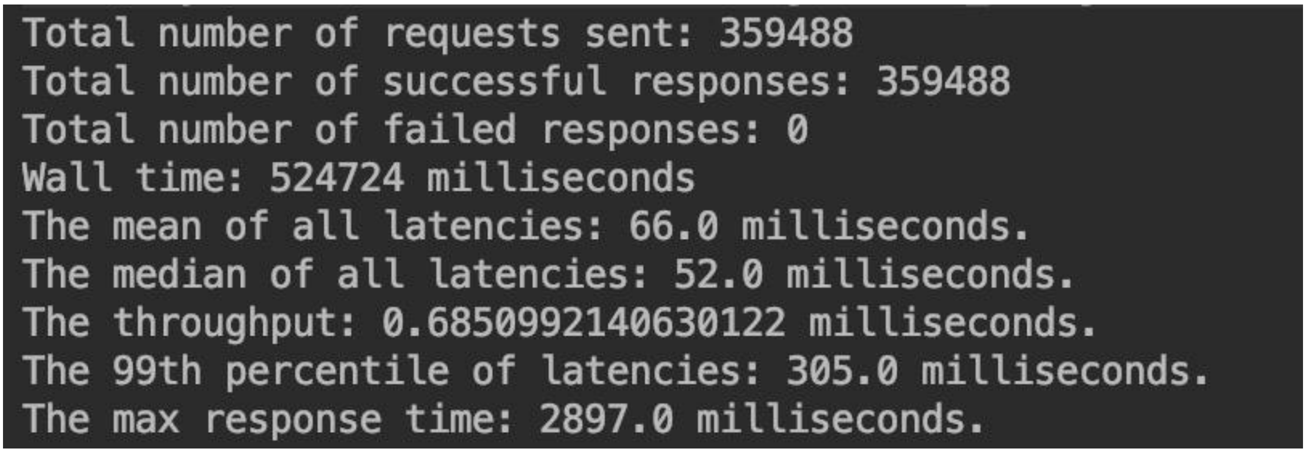
```
Total number of requests sent: 359680
Total number of successful responses: 359680
Total number of failed responses: 0
Wall time: 124721 milliseconds
The mean of all latencies: 111 milliseconds.
The median of all latencies: 50 milliseconds.
The throughput: 2.8838768130467205 milliseconds.
The 99th percentile of latencies: 300.0 milliseconds.
The max response time: 3561.0 milliseconds.
```

128 Thread



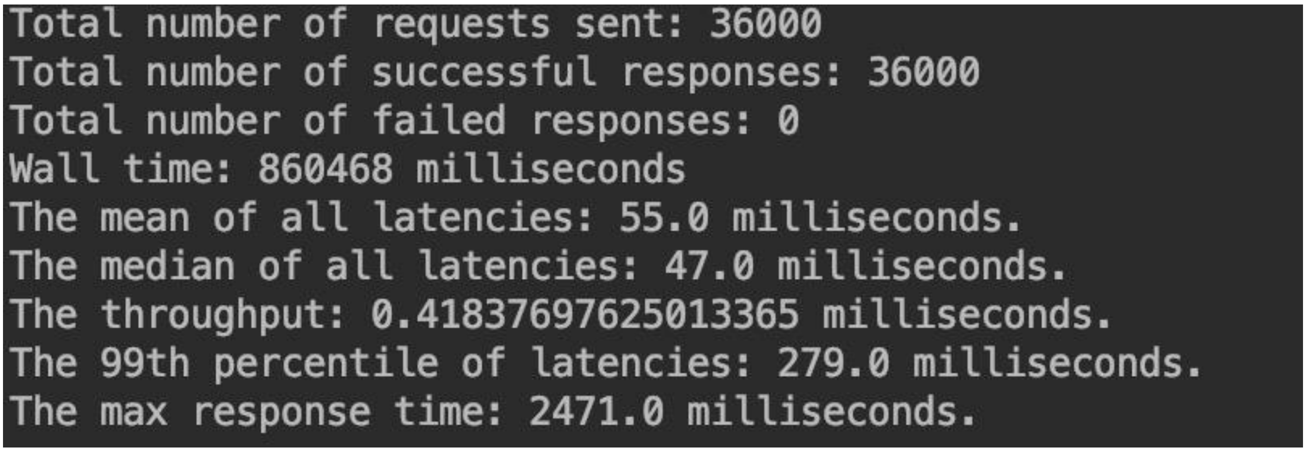
```
Total number of requests sent: 359488
Total number of successful responses: 359488
Total number of failed responses: 0
Wall time: 438184 milliseconds
The mean of all latencies: 83.0 milliseconds.
The median of all latencies: 86.0 milliseconds.
The throughput: 0.8204042137549522 milliseconds.
The 99th percentile of latencies: 632.0 milliseconds.
The max response time: 8809.0 milliseconds.
```

64Thread

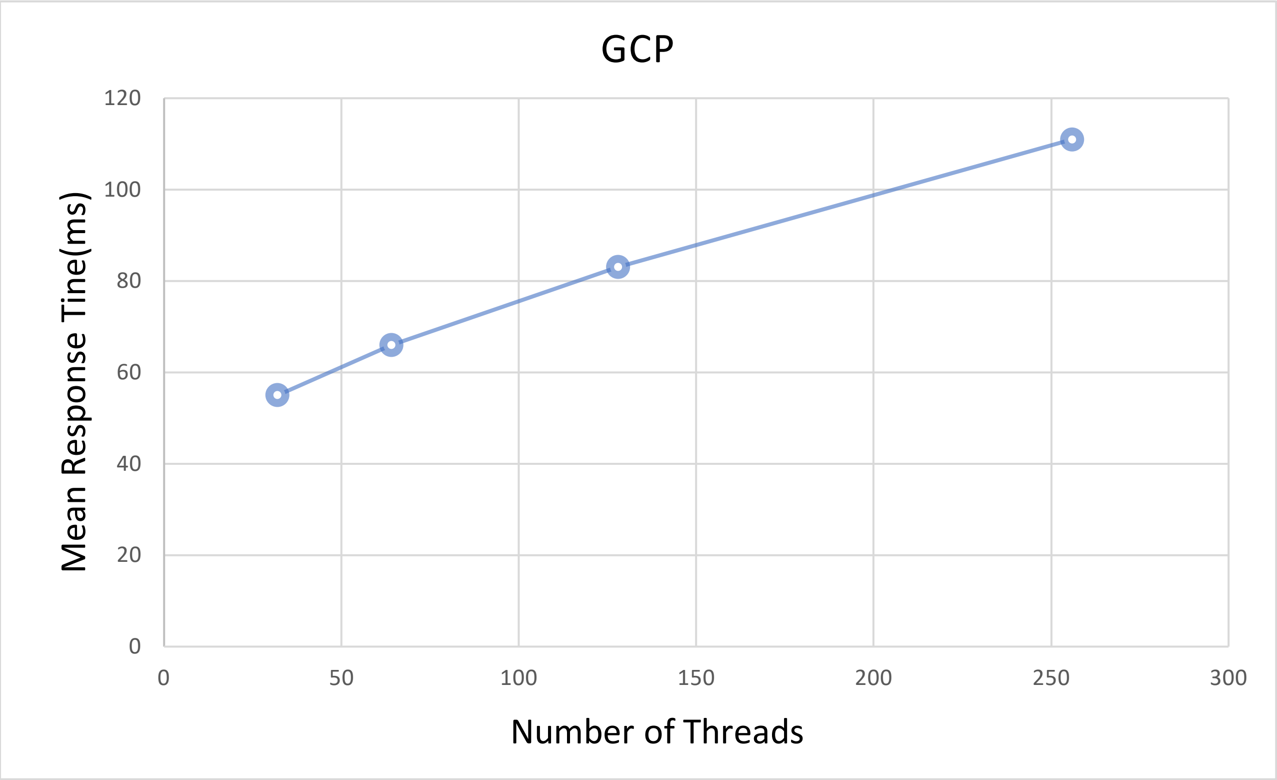


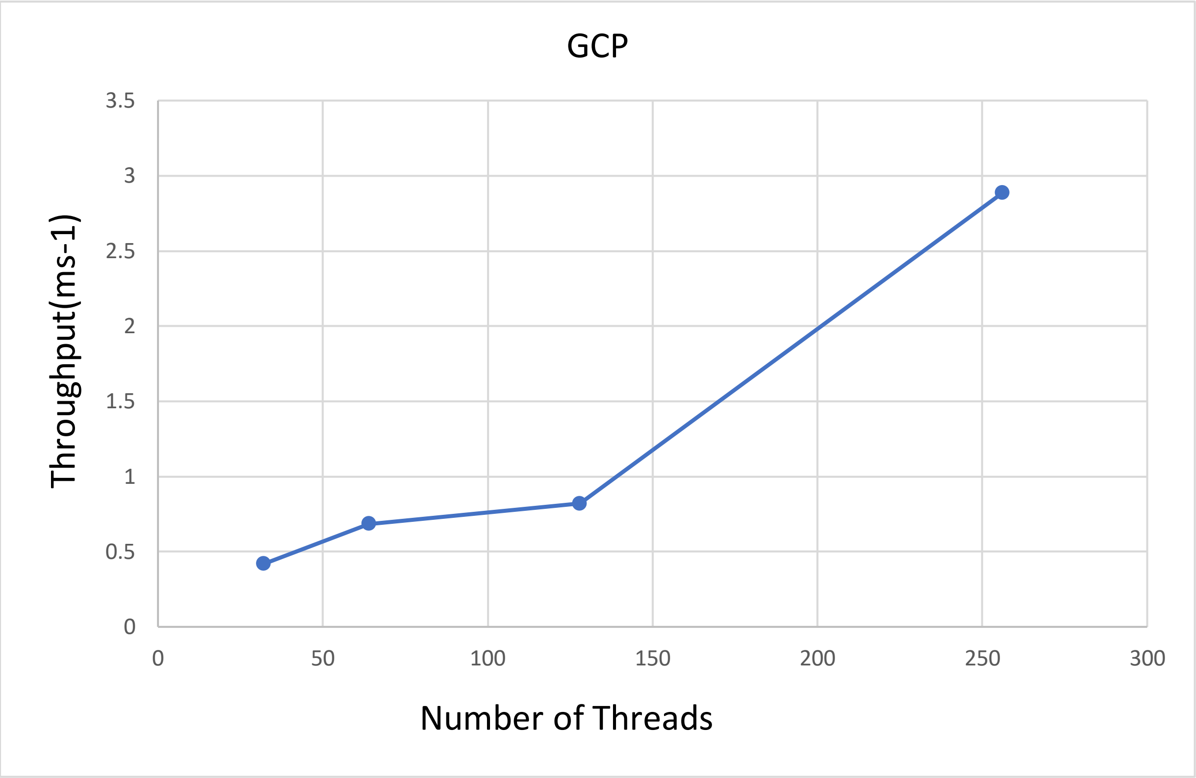
```
Total number of requests sent: 359488
Total number of successful responses: 359488
Total number of failed responses: 0
Wall time: 524724 milliseconds
The mean of all latencies: 66.0 milliseconds.
The median of all latencies: 52.0 milliseconds.
The throughput: 0.6850992140630122 milliseconds.
The 99th percentile of latencies: 305.0 milliseconds.
The max response time: 2897.0 milliseconds.
```

32 Thread



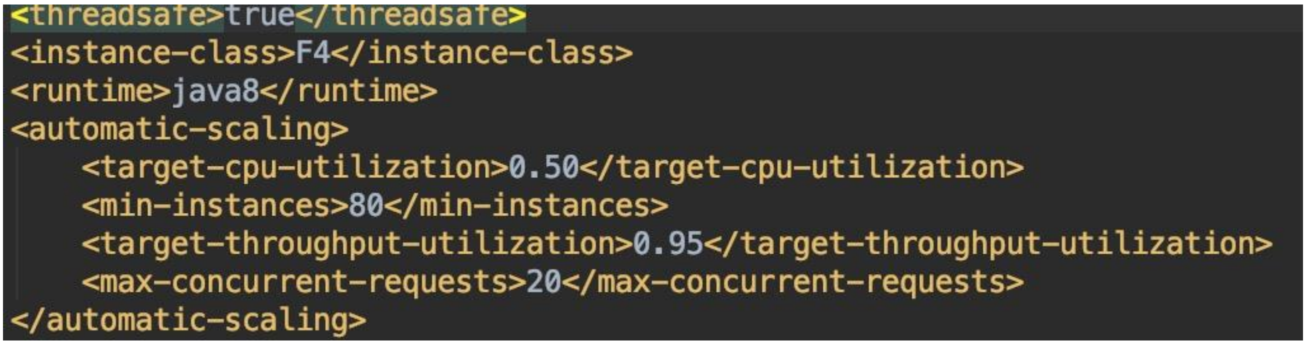
```
Total number of requests sent: 36000
Total number of successful responses: 36000
Total number of failed responses: 0
Wall time: 860468 milliseconds
The mean of all latencies: 55.0 milliseconds.
The median of all latencies: 47.0 milliseconds.
The throughput: 0.41837697625013365 milliseconds.
The 99th percentile of latencies: 279.0 milliseconds.
The max response time: 2471.0 milliseconds.
```





As the pictures show above, both throughput and mean response time are increasing when the threads increase.

The configuration of GCP:



```
<threadsafe>true</threadsafe>
<instance-class>F4</instance-class>
<runtime>java8</runtime>
<automatic-scaling>
    <target-cpu-utilization>0.50</target-cpu-utilization>
    <min-instances>80</min-instances>
    <target-throughput-utilization>0.95</target-throughput-utilization>
    <max-concurrent-requests>20</max-concurrent-requests>
</automatic-scaling>
```

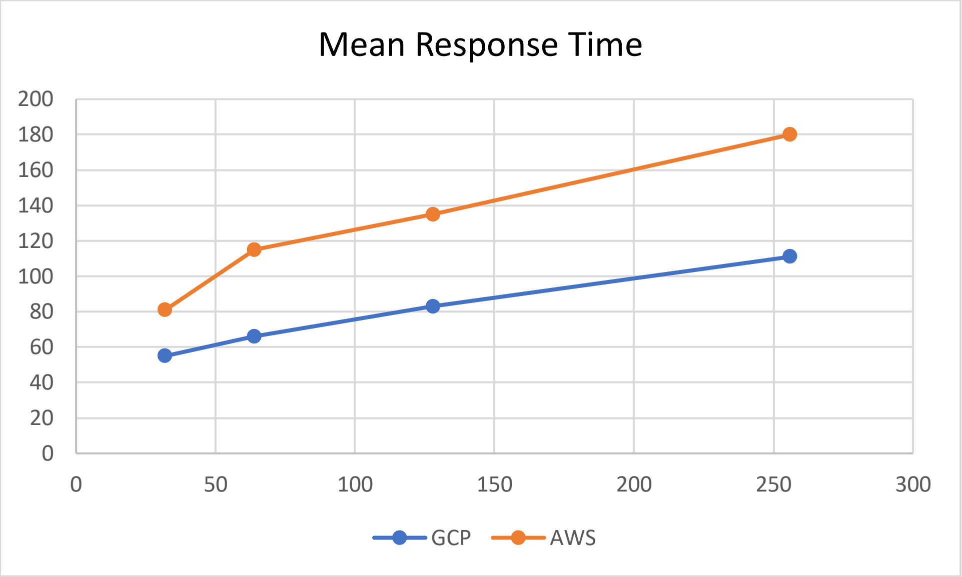
3. AWS VS Google Cloud

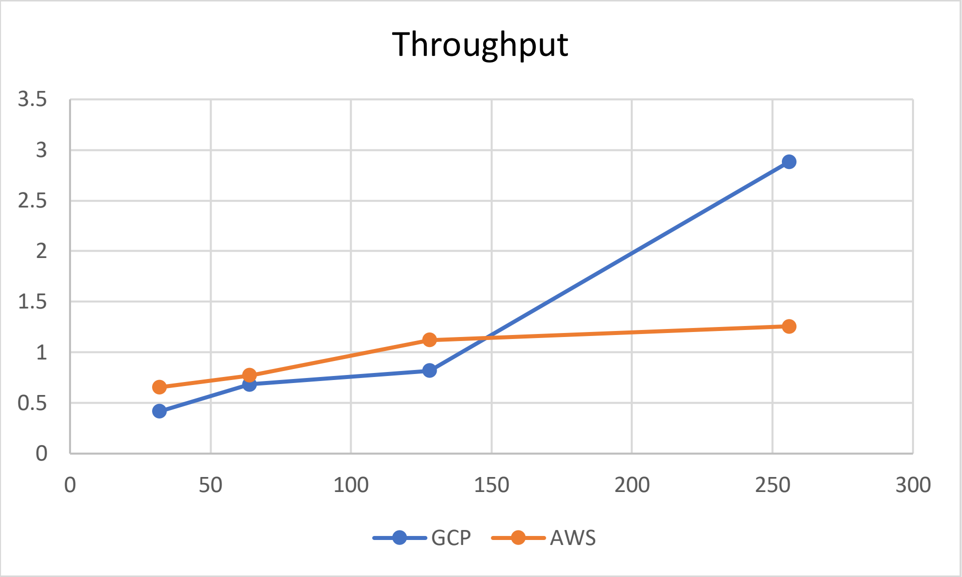
(1) AWS/GCP parameters

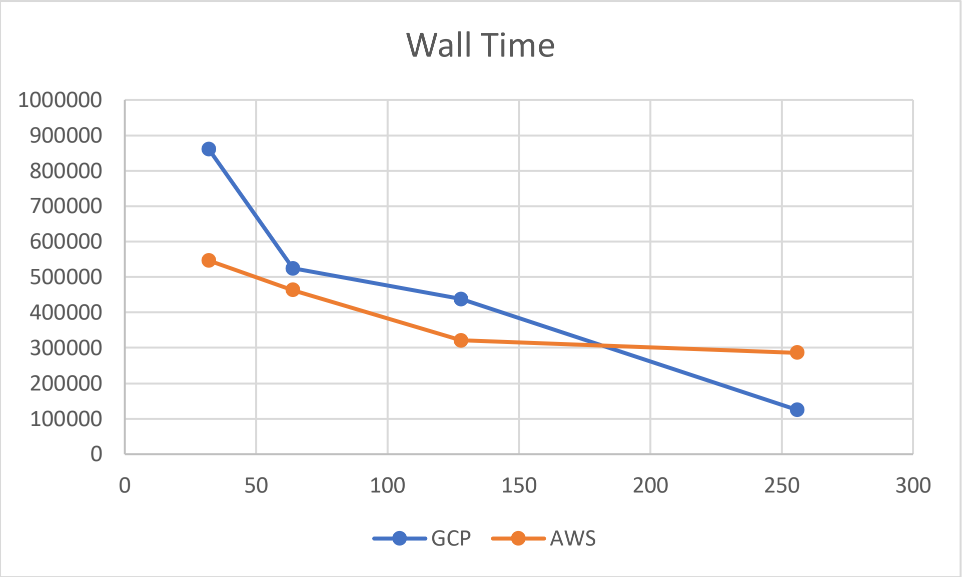
Cloud Platform	Auto-scaling	Database Hardware	Server Type
AWS	No	4 CPU/15GB RAM	Micro
GCP	Yes	46 CPU/416GB RAM	F4

Originally the GCP works not well, its slower than AWS. Then I upgrade the database instance to high performance and upgrade the server to F4. Finally the performance becomes much better.

(2) AWS/GCP results







From the pictures above, AWS is faster when threads are 32, 64 and 128, but GCP is faster when thread is 256. So generally, AWS is faster.

As for the cost, I spent little on AWS, but about 70 dollars on GCP to upgrade the database and server. The GCP use auto scale, so that I don’t need to build load balancers one by one. But the thing is that, the more instances GCP uses, the more I need to pay.

As for the building and deploy, I think both of them are complex. For me, GCP is easier to understand to deploy and run. But the authentication IAM is too difficult to set and the bugs during setting up are hard to solve.