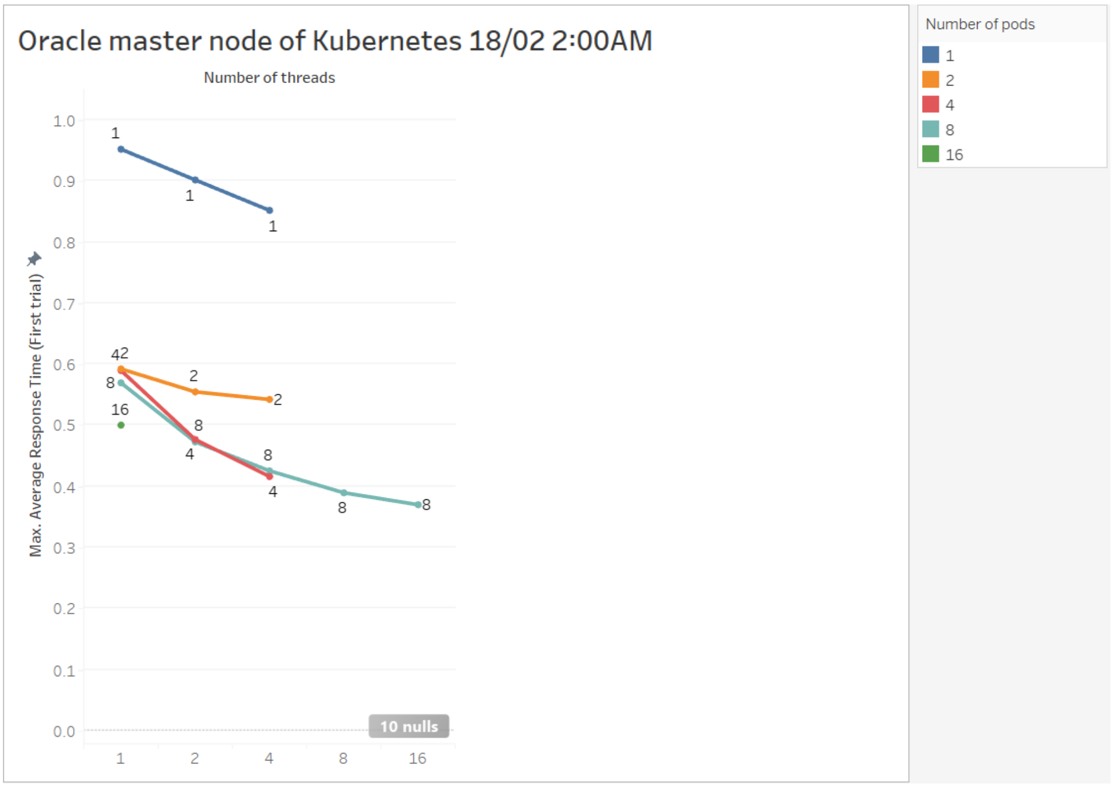
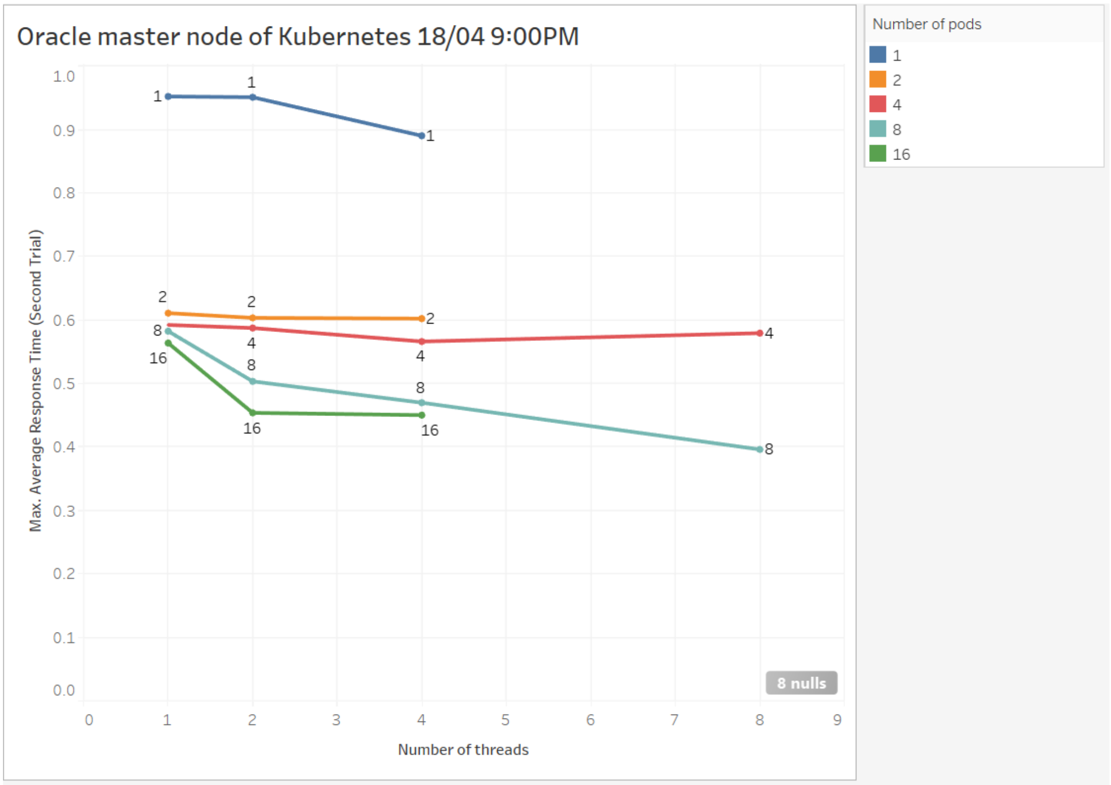
FIT5225 Assignment1

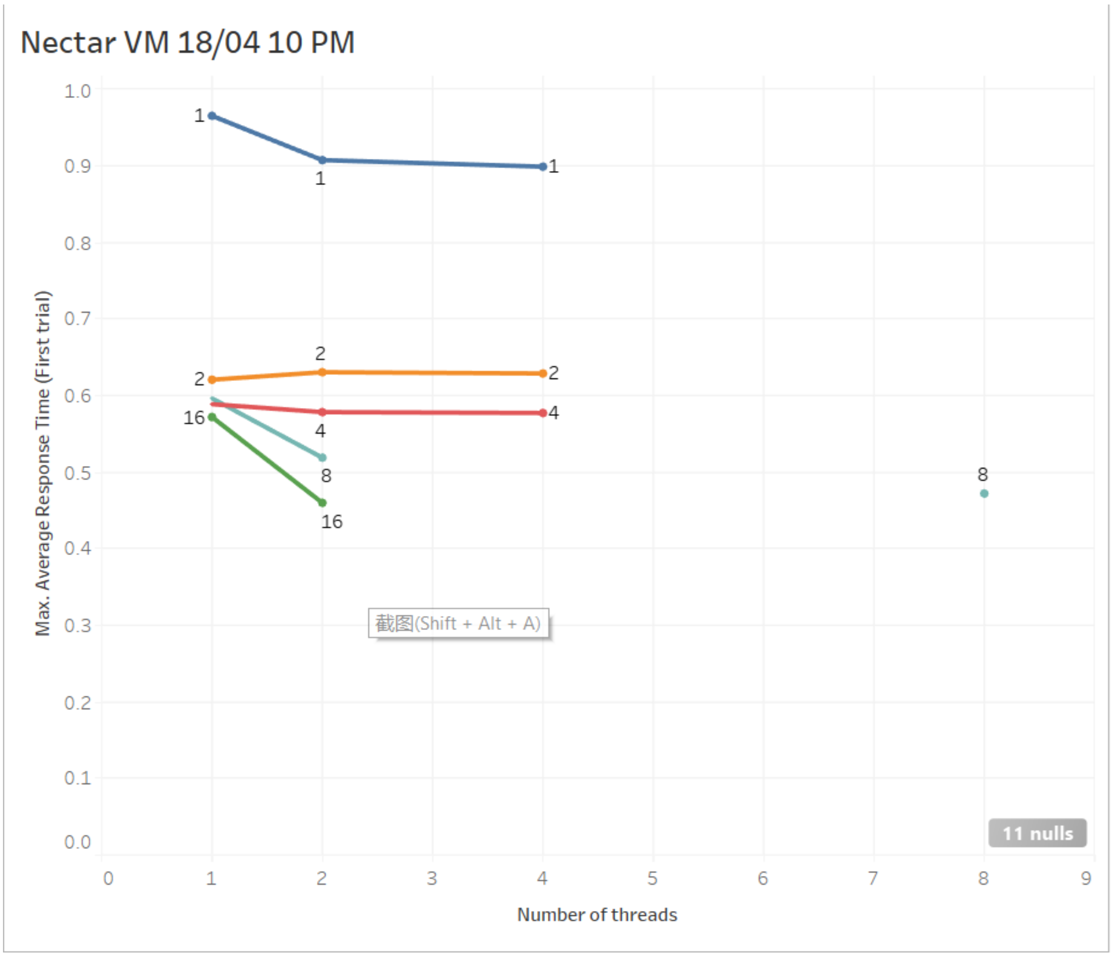
Mengzhen Li

29654378

Experiment of client executed locally on oracle master node.







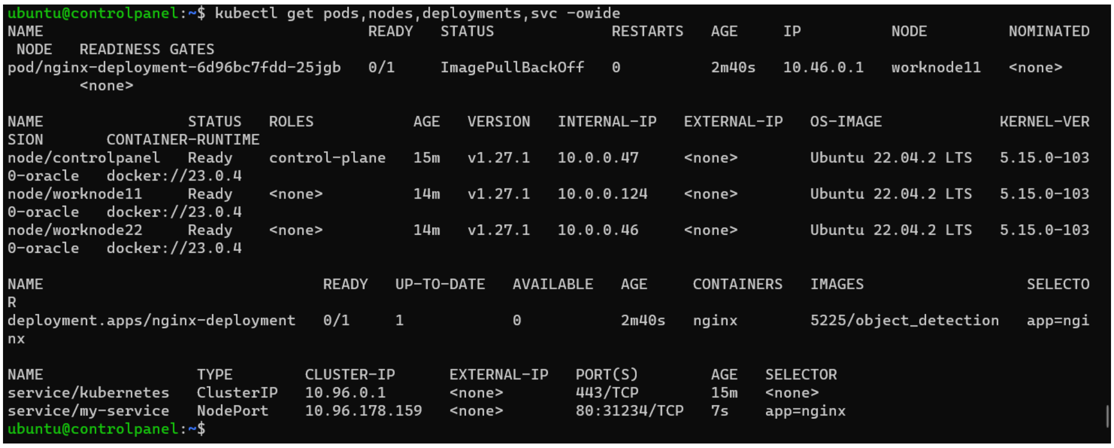
Based on the data recorded in my experiments, when the number of threads increases, the average response time decreases. For given number of threads, more pods can also decrease the response time. The combination of a higher number of pods and threads can reduce the response time maximumly. Some cases may not exactly follow the trend above. Like in my experiments, 2 pods in 1 thread and 4 pods in 1 thread spend similar time to response. 4 pods in 2 threads somehow even have less response time than 8 pods in 2 threads. This may be caused by pods competing for the same resources, then adding more pods may not improve performance. In fact, it may even slow down the overall response time if the resources become overburdened. Similarly, if too many threads are used, then they may compete for the same resources and slow down the overall response time. The age of the pods may also affect the results. I tested each combination 3 times and, in some cases, the third time's result is much quicker than the first two. This may also lead to the problem that the service might require some time to fully deploy after changing the "deployment. yaml".

The second complete experiment shows a trend that is not exactly similar to the first one. The second experiment uses different instances with new images created and up dated files. In the second experiment, I fixed the issue that when multiple pods are developed, some of them are not in running status, except 16 pods. In the first program when curl 30007 outside the node, it takes long time to receive the response, but the second spend much less time. However, the result of the second test is really confusing that when more threads are deployed the time somehow does not decrease that much. This may have been caused by gunicorn which allowed me to run more threads but did not really fix the issue. It may also be influenced by network status because 8 PM to 12 PM is a time period that everyone is relaxing online.

The nectar experiment is really different than I expected. My thought was executing on VM should be much slower than local, considering the experience in tutorial. In the most stable situation 1 pods with 1 thread, the nectar VM only takes 0.01 seconds more than local. In other situation there are few times is even faster than local. Comparing with my own experience using Monash VM back to China in covid-19 period. I can draw the conclusion that when VM in an enough network speed and close to local server the experience can be as good as local. The thing that most affects VM experience should be the latency caused by long distance to the local server.

However, there are few problems in my experiments. Obviously, none of my experiments can successfully run on 16 threads except 8 pods, and when the number of pods increased to 16, the client file can only execute successfully with 1 thread otherwise there will be a lot of time out error.

For find out the problems. Firstly, I checked the status of my pods, nodes, and deployments.

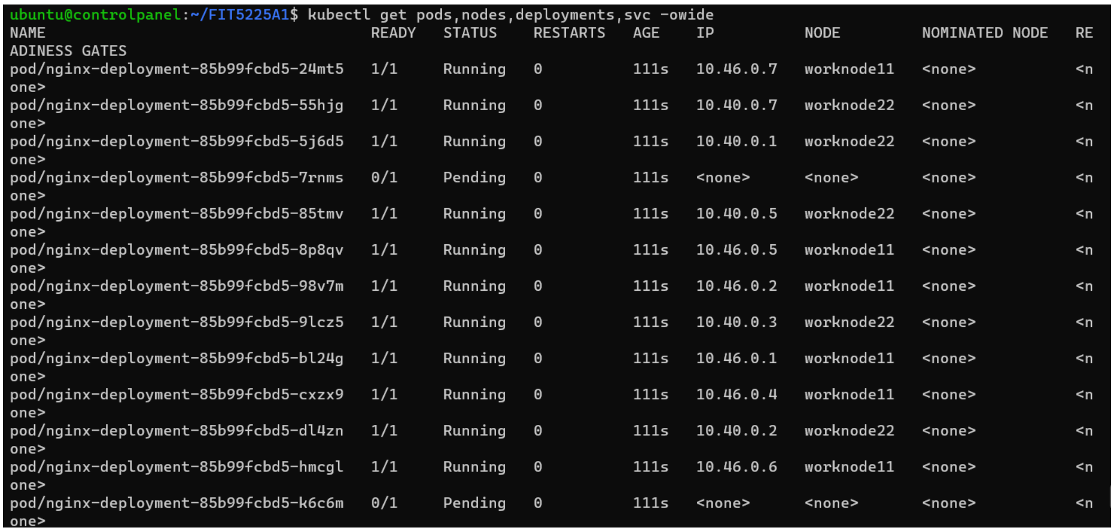


```
ubuntu@controlpanel:~$ kubectl get pods,nodes,deployments,svc -owide
NAME                                     READY   STATUS             RESTARTS   AGE     IP          NODE         NOMINATED
 NODE   READINESS GATES
pod/nginx-deployment-6d96bc7fdd-25jgb   0/1     ImagePullBackOff   0          2m40s   10.46.0.1   worknode11   <none>
        <none>

NAME                STATUS   ROLES           AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VER
SION       CONTAINER-RUNTIME
node/controlpanel   Ready    control-plane   15m   v1.27.1   10.0.0.47     <none>        Ubuntu 22.04.2 LTS   5.15.0-103
0-oracle   docker://23.0.4
node/worknode11     Ready    <none>          14m   v1.27.1   10.0.0.124    <none>        Ubuntu 22.04.2 LTS   5.15.0-103
0-oracle   docker://23.0.4
node/worknode22     Ready    <none>          14m   v1.27.1   10.0.0.46     <none>        Ubuntu 22.04.2 LTS   5.15.0-103
0-oracle   docker://23.0.4

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES                   SELECTO
R
deployment.apps/nginx-deployment   0/1     1            0           2m40s   nginx        5225/object_detection   app=ngi
nx

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        15m   <none>
service/my-service   NodePort    10.96.178.159   <none>        80:31234/TCP   7s    app=nginx
ubuntu@controlpanel:~$
```



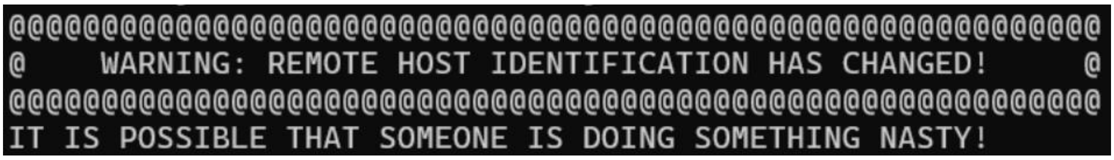
```
ubuntu@controlpanel:~/FIT5225A1$ kubectl get pods,nodes,deployments,svc -owide
NAME                                    READY   STATUS    RESTARTS   AGE    IP          NODE         NOMINATED NODE   RE
ADINESS GATES
pod/nginx-deployment-85b99fcbd5-24mt5   1/1     Running   0          111s   10.46.0.7   worknode11   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-55hjg   1/1     Running   0          111s   10.40.0.7   worknode22   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-5j6d5   1/1     Running   0          111s   10.40.0.1   worknode22   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-7rnms   0/1     Pending   0          111s   <none>      <none>       <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-85tmv   1/1     Running   0          111s   10.40.0.5   worknode22   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-8p8qv   1/1     Running   0          111s   10.46.0.5   worknode11   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-98v7m   1/1     Running   0          111s   10.46.0.2   worknode11   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-9lcz5   1/1     Running   0          111s   10.40.0.3   worknode22   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-bl24g   1/1     Running   0          111s   10.46.0.1   worknode11   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-cxzx9   1/1     Running   0          111s   10.46.0.4   worknode11   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-dl4zn   1/1     Running   0          111s   10.40.0.2   worknode22   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-hmcgl   1/1     Running   0          111s   10.46.0.6   worknode11   <none>           <n
one>
pod/nginx-deployment-85b99fcbd5-k6c6m   0/1     Pending   0          111s   <none>      <none>       <none>           <n
one>
```

I just couldn’t figure out why this is happening. I tried a few steps. resetting the cluster and starting over. Rebuilding my dock image. Also tried to add memory to 2048

Mib but never worked out. Although I didn’t fix this problem, during my research, I think the major problem is on port connection. Working with service is not stable in my project. Sometimes I can get results by “curl ip:80” quickly sometimes it shows no connection. Sometimes I can curl from outside by using”:30007” but I cannot curl “:80” from nodes.

Distributed systems challenges

For distributed systems challenges, I must discuss security first. The authentications on the windows system were really annoying in the first few weeks. I completed all tasks in week1, but I cannot connect to nectar in week2. After consulting with my tutor, she helped me set a file directly in terminal after class but somehow it won’t work in week3. Since I also have an old mac book I successfully connect to nectar before tutorial by watching the recordings and the recording is using mac OS. However, in the tutorial we start to use oracle and the pub file cannot be opened in my old mac OS and I cannot use the public key. It is until week 4 I finally figured out how to connect to my instances. When doing the assignment, I set my disk E’s security and removed all inherited permissions to this whole disk. Also created a special folder just for all the files of this assignment. I saved a pub file and key file just for oracle and a pem file just for nectar in case any problems occur. Although I still got this problem one time, I just deleted this work node instance and created a new one and never got similar problems. After all the practice, now I know it is important to have a backup plan in case of technical issues or unexpected problems. Saving multiple keys and pem files saved to ensure they could still access their resources even if one file did not work. It is also important to properly secure and manage access to resources, such as setting specific permissions and creating separate folders for specific assignments or tasks. Finally, when encountering problems, it is important to troubleshoot and try different solutions, such as deleting and recreating instances, in order to find a solution.



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

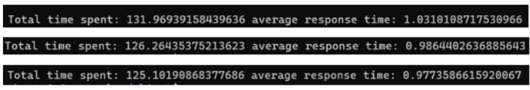
The second challenge is scalability. In tutorial 5, we created k8s master with 2 core and 8 Memory. In my first try of this assignment, I also created my control panel with 2 cores then I found it requires 4 in assignment instructions. Apparently, the scalability for this is very low which could not be changed. For controlling the performance loss, the burstable option is really useful for providing a baseline level of CPU performance with the ability to a higher level to support occasional increase of usage. Also, by analyzing the experiment results, the number of threads should be lower than the number of pods to get maximum efficiency and avoid performance bottlenecks. As a result, building a scalable and reliable distributed system like Kubernetes comes with several challenges. These challenges include ensuring fault tolerance, load balancing, network communication, and security. Kubernetes tackles these challenges by providing built-in mechanisms for managing application instances and resources, as well as automated scaling and self-healing capabilities. Additionally, the use of containers allows for a more portable and flexible deployment model, while the use of microservices promotes modularity and decoupling. However, despite these benefits, there are still potential issues that need to be addressed, such as resource constraints, data consistency, and managing dependencies between services. Overall, designing and managing a distributed system requires careful planning, testing, and monitoring to ensure optimal performance and reliability.

Failure handling is always happening during the experiments. It is not 100% that I can have a record response time for 100% success rate. The failure can be detected and reported on terminal so developers can find the problems. Also, the client file includes redundancy so the project can continue execute after one or two timeouts. If this project is already for different users, I believe the masking is needed for the program.

Response time results test before experiment 1

-Pods:1

Threads:1



```
Total time spent: 131.96939158439636 average response time: 1.0310108717530966
Total time spent: 126.26435375213623 average response time: 0.9864402636885643
Total time spent: 125.10190868377686 average response time: 0.9773586615920067
```

Time: 1.031

Threads:2

Time: NA

Threads:4

Time: NA

Threads:8

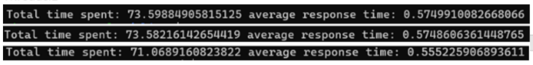
Time: NA

Threads:16

Time: NA

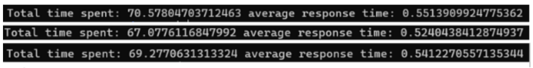
Pods:2

Threads:1



```
Total time spent: 73.59884905815125 average response time: 0.5749910082668066
Total time spent: 73.58216142654419 average response time: 0.5748606361448765
Total time spent: 71.0689160823822 average response time: 0.555225906893611
```

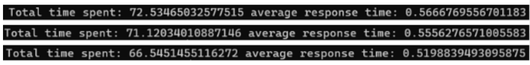
Threads:2



```
Total time spent: 70.57804703712463 average response time: 0.5513909924775362
Total time spent: 67.0776116847992 average response time: 0.5240438412874937
Total time spent: 69.2770631313324 average response time: 0.5412270557135344
```

Pods:4

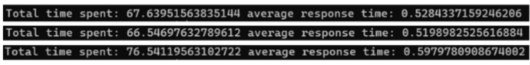
Threads:1



```
Total time spent: 72.53465032577515 average response time: 0.5666769556701183
Total time spent: 71.12034010887146 average response time: 0.5556276571005583
Total time spent: 66.5451455116272 average response time: 0.5198839493095875
```

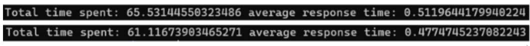
Pods:8

Threads:1



```
Total time spent: 67.63951563835144 average response time: 0.5284337159246206
Total time spent: 66.54697632789612 average response time: 0.5198982525616884
Total time spent: 76.54119563102722 average response time: 0.5979780908674002
```

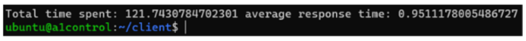
Threads:2



```
Total time spent: 65.53144550323486 average response time: 0.5119644179940224
Total time spent: 61.11673903465271 average response time: 0.4774745237082243
```

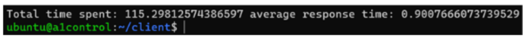
Pods:1

Threads:1



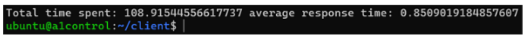
```
Total time spent: 121.7430784702301 average response time: 0.9511178005486727
ubuntu@a1control:~/client$ |
```

Threads:2



```
Total time spent: 115.29812574386597 average response time: 0.9007666073739529
ubuntu@a1control:~/client$ |
```

Threads:4



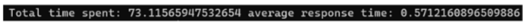
```
Total time spent: 108.91544556617737 average response time: 0.8509019184857607
ubuntu@a1control:~/client$ |
```

Threads:8

Failure

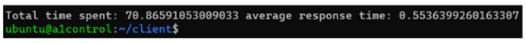
Pods:2

Threads:1



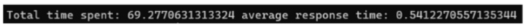
```
Total time spent: 73.11565947532654 average response time: 0.5712160896509886
```

Threads:2



```
Total time spent: 70.86591053009033 average response time: 0.5536399260163307
ubuntu@a1control:~/client$
```

Threads:4



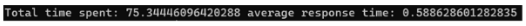
```
Total time spent: 69.2770631313324 average response time: 0.5412270557135344
```

Threads:8

Failure

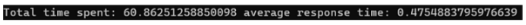
Pods:4

Threads:1



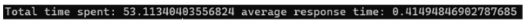
```
Total time spent: 75.34446096420288 average response time: 0.588628601282835
```

Threads:2



```
Total time spent: 60.86251258850098 average response time: 0.4754883795976639
```

Threads:4



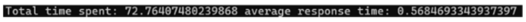
```
Total time spent: 53.11340403556824 average response time: 0.41494846902787685
```

Threads:8

Failure

Pods:8

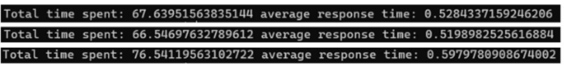
Threads:1



```
Total time spent: 72.76407480239868 average response time: 0.5684693343937397
```

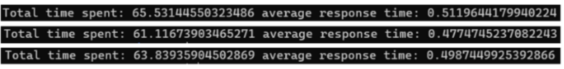
Pods:8

Threads:1



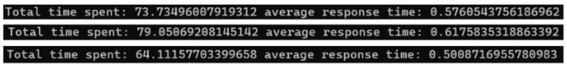
```
Total time spent: 67.63951563835144 average response time: 0.5284337159246206
Total time spent: 66.54697632789612 average response time: 0.5198982525616884
Total time spent: 76.54119563102722 average response time: 0.5979780908674002
```

Threads:2



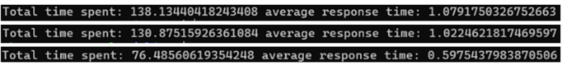
```
Total time spent: 65.53144550323486 average response time: 0.5119644179940224
Total time spent: 61.11673903465271 average response time: 0.4774745237082243
Total time spent: 63.83935904502869 average response time: 0.4987449925392866
```

Threads:4



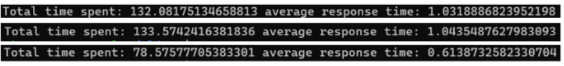
```
Total time spent: 73.73496007919312 average response time: 0.5760543756186962
Total time spent: 79.05069208145142 average response time: 0.6175835318863392
Total time spent: 64.11157703399658 average response time: 0.5008716955780983
```

Threads:8



```
Total time spent: 138.13440418243408 average response time: 1.0791750326752663
Total time spent: 130.87515926361084 average response time: 1.0224621817469597
Total time spent: 76.48560619354248 average response time: 0.5975437983870506
```

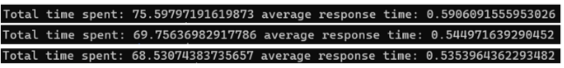
Threads:16



```
Total time spent: 132.08175134658813 average response time: 1.0318886823952198
Total time spent: 133.5742416381836 average response time: 1.0435487627983093
Total time spent: 78.57577705383301 average response time: 0.6138732582330704
```

Pods:16

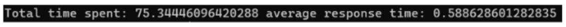
Threads:1



```
Total time spent: 75.59797191619873 average response time: 0.5906091555953026
Total time spent: 69.75636982917786 average response time: 0.544971639290452
Total time spent: 68.53074383735657 average response time: 0.5353964362293482
```

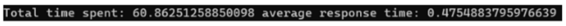
Pods:4

Threads:1



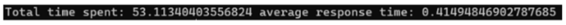
```
Total time spent: 75.34446096420288 average response time: 0.588628601282835
```

Threads:2



```
Total time spent: 60.86251258850098 average response time: 0.4754883795976639
```

Threads:4



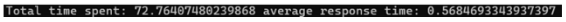
```
Total time spent: 53.11340403556824 average response time: 0.41494846902787685
```

Threads:8

Failure

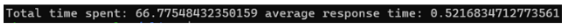
Pods:8

Threads:1



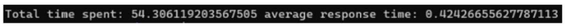
```
Total time spent: 72.76407480239868 average response time: 0.5684693343937397
```

Threads:2



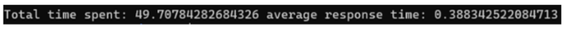
```
Total time spent: 66.77548432350159 average response time: 0.5216834712773561
```

Threads:4



```
Total time spent: 54.306119203567505 average response time: 0.42426655627787113
```

Threads:8



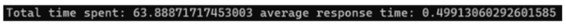
```
Total time spent: 49.70784282684326 average response time: 0.388342522084713
```

Threads:16

Failure

Pods:16

Threads:1



```
Total time spent: 63.88871717453003 average response time: 0.49913060292601585
```

Threads:2