FIT5225 ASS1 Report

Name: Jiawei Ren

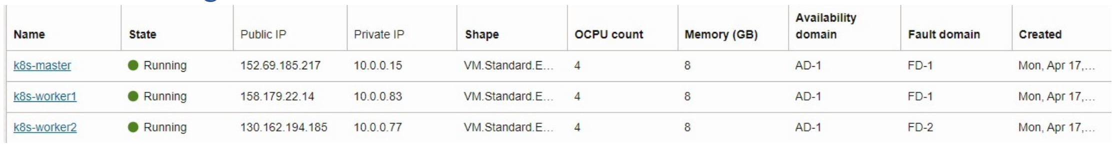
Student Number:32073119

Tutor name: Jay Zhao, Negin Akbari

Contents

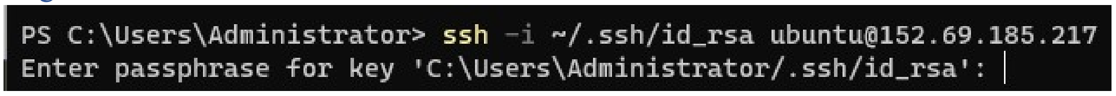
1. Local Machine:

Instances Configuration:



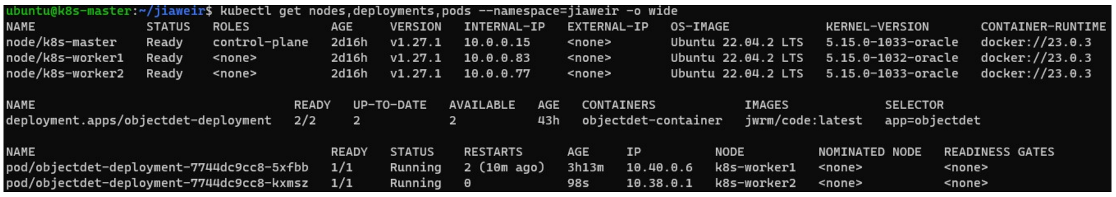
Name	State	Public IP	Private IP	Shape	OCPU count	Memory (GB)	Availability domain	Fault domain	Created
k8s-master	● Running	152.69.185.217	10.0.0.15	VM.Standard.E...	4	8	AD-1	FD-1	Mon, Apr 17,...
k8s-worker1	● Running	158.179.22.14	10.0.0.83	VM.Standard.E...	4	8	AD-1	FD-1	Mon, Apr 17,...
k8s-worker2	● Running	130.162.194.185	10.0.0.77	VM.Standard.E...	4	8	AD-1	FD-2	Mon, Apr 17,...

Log Into Instance:



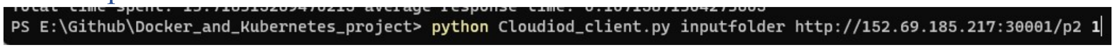
```
PS C:\Users\Administrator> ssh -i ~/.ssh/id_rsa ubuntu@152.69.185.217
Enter passphrase for key 'C:\Users\Administrator/.ssh/id_rsa':
```

Overview of Instances:



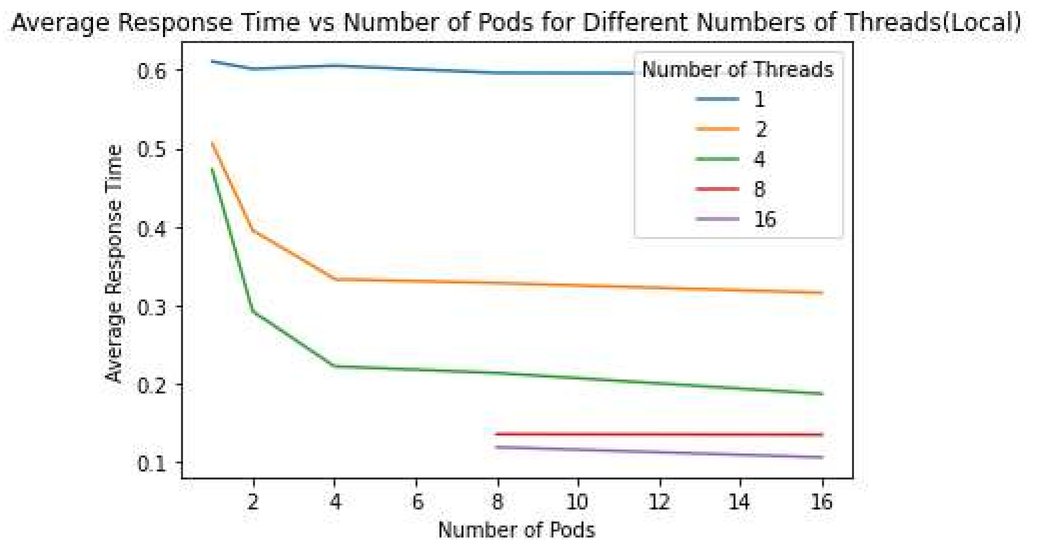
```
ubuntu@k8s-master:~/jiaweir$ kubectl get nodes,deployments,pods --namespace=jiaweir -o wide
NAME                STATUS   ROLES           AGE    VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION        CONTAINER-RUNTIME
node/k8s-master     Ready    control-plane   2d16h  v1.27.1   10.0.0.15     <none>        Ubuntu 22.04.2 LTS   5.15.0-1033-oracle    docker://23.0.3
node/k8s-worker1    Ready    <none>          2d16h  v1.27.1   10.0.0.83     <none>        Ubuntu 22.04.2 LTS   5.15.0-1032-oracle    docker://23.0.3
node/k8s-worker2    Ready    <none>          2d16h  v1.27.1   10.0.0.77     <none>        Ubuntu 22.04.2 LTS   5.15.0-1033-oracle    docker://23.0.3

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS            IMAGES              SELECTOR
deployment.apps/objectdet-deployment   2/2     2            2           43h   objectdet-container   jwrm/code:latest    app=objectdet

NAME                                         READY   STATUS    RESTARTS       AGE     IP           NODE          NOMINATED NODE   READINESS GATES
pod/objectdet-deployment-7744dc9cc8-5xfbb    1/1     Running   2 (10m ago)    3h13m   10.40.0.6    k8s-worker1   <none>           <none>
pod/objectdet-deployment-7744dc9cc8-kxmsz    1/1     Running   0              98s     10.38.0.1    k8s-worker2   <none>           <none>
```

Start Experiment:



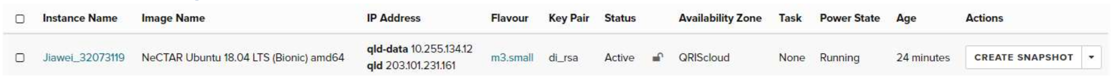
```
Total time spent: [illegible] average response time: [illegible]
PS E:\Github\Docker_and_Kubernetes_project> python Cloudiod_client.py inputfolder http://152.69.185.217:30001/p2 1
```

Plot



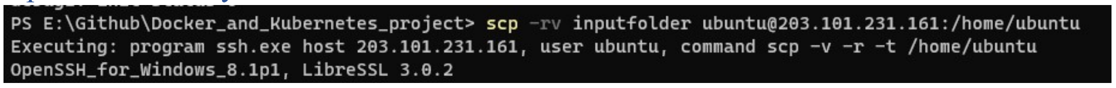
2. Nectar client:

Instance Configuration:

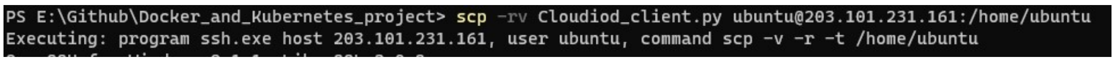


	Instance Name	Image Name	IP Address	Flavour	Key Pair	Status		Availability Zone	Task	Power State	Age	Actions
▢	Jiawei_32073119	NeCTAR Ubuntu 18.04 LTS (Bionic) amd64	**qld-data** 10.255.134.12 **qld** 203.101.231.161	m3.small	di_rsa	Active	🔓	QRIScloud	None	Running	24 minutes	CREATE SNAPSHOT ▾

Upload Necessary Files into Nectar Instance:

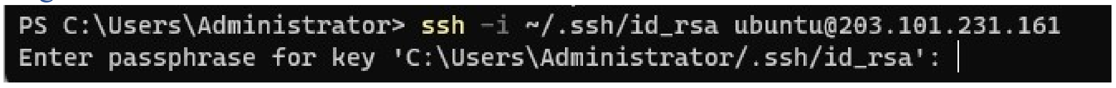


```
PS E:\Github\Docker_and_Kubernetes_project> scp -rv inputfolder ubuntu@203.101.231.161:/home/ubuntu
Executing: program ssh.exe host 203.101.231.161, user ubuntu, command scp -v -r -t /home/ubuntu
OpenSSH_for_Windows_8.1p1, LibreSSL 3.0.2
```



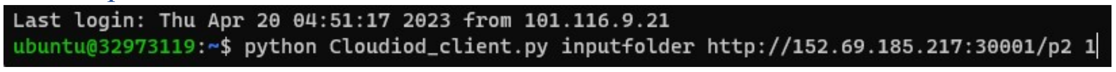
```
PS E:\Github\Docker_and_Kubernetes_project> scp -rv Cloudiod_client.py ubuntu@203.101.231.161:/home/ubuntu
Executing: program ssh.exe host 203.101.231.161, user ubuntu, command scp -v -r -t /home/ubuntu
```

Log Into Instance:



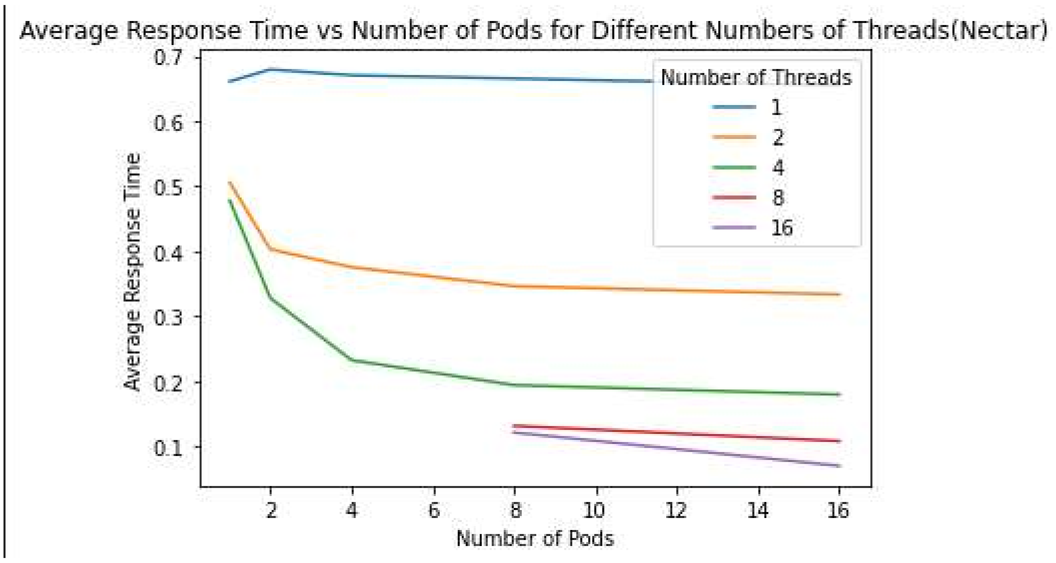
```
PS C:\Users\Administrator> ssh -i ~/.ssh/id_rsa ubuntu@203.101.231.161
Enter passphrase for key 'C:\Users\Administrator/.ssh/id_rsa':
```

Start Experiment:



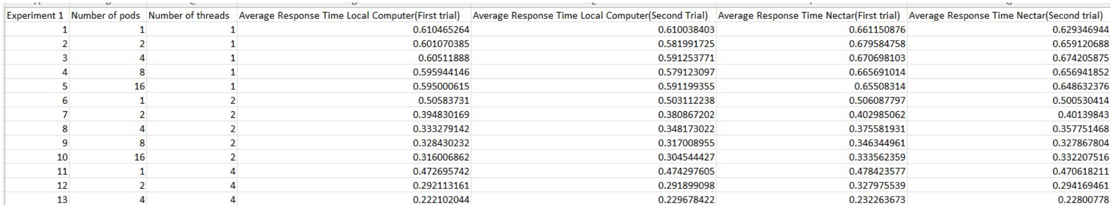
```
Last login: Thu Apr 20 04:51:17 2023 from 101.116.9.21
ubuntu@32973119:~$ python Cloudiod_client.py inputfolder http://152.69.185.217:30001/p2 1
```

Plot:



3. Plots Observation:

Snapshot of experiment data:



Experiment 1	Number of pods	Number of threads	Average Response Time Local Computer(First trial)	Average Response Time Local Computer(Second Trial)	Average Response Time Nectar(First trial)	Average Response Time Nectar(Second trial)
1	1	1	0.610465264	0.610038403	0.661150876	0.629346944
2	2	1	0.601070385	0.581991725	0.679584758	0.659120688
3	4	1	0.60511888	0.591253771	0.670698103	0.674205875
4	8	1	0.595944146	0.579123097	0.665691014	0.656941852
5	16	1	0.595000615	0.591199355	0.65508314	0.648632376
6	1	2	0.50583731	0.503112238	0.506087797	0.500530414
7	2	2	0.394830169	0.380867202	0.402985062	0.40139843
8	4	2	0.333279142	0.348173022	0.375581931	0.357751468
9	8	2	0.328430232	0.317008955	0.346344961	0.327867804
10	16	2	0.316006862	0.304544427	0.333562359	0.332207516
11	1	4	0.472695742	0.474297605	0.478423577	0.470618211
12	2	4	0.292113161	0.291899098	0.327975539	0.294169461
13	4	4	0.222102044	0.229678422	0.232263673	0.22800778

Observation and Explain Results:

1. **Impact of Threads:** for both testing environment, the average response time decreases as the number of threads increases. This observation explains the fact that more threads have better concurrency and allows server to handle more works simultaneously. On the other hand, there is a limit to how much performance I can gain by increasing the number of threads. Too many threads can cause problem if there are not enough resources such as CPU and RAM.
2. **Impact of Pods:** Overall, as the number of pods increases, the performance of the server is also better, which is reflected in the number of threads we can use. For example, only the configuration of 8 pods and 16 pods can increase the number of threads to 16. In the most of cases, average response time is reduced as more pods were added, which means that the system could benefit from the additional resources, except when the number of threads is 1, the performance is relatively constant. Furthermore, the impact between the number of pods and response time is not consistent across different number of threads. Potential reason for this can be the efficiency of load balancing system and the nature of workload
3. **Local VS Nectar:** the average of response time on the local computer is generally lower than on Nectar. Factors could be network latency, speed, different hardware, software and internet protocol.

4. Three Challenges

Heterogeneity:

In distributed systems, different hardware and software may have different communication protocols between clients and servers. These varieties can make the process of building and maintain the system more complicate. To meet this challenge, I have to make sure my web service adhere to defined standards and protocol for data exchange such as JSON and encode data, communication protocols such as HTTP and TCP. In the assignment, the image data is encoded using base64 before being sent through the client API. So, on the server side, I have to use base64 to decode the image data so that my function can process the data and store it in a predefined format. In the cloud, I have to open the supported protocols and ports for docker so that I can deploy docker on my instance. Consider supporting multiple platforms, we have to install weave net, which is a networking model that manage the communication between containers across multiple hosts.

Openness:

In a distributed system, openness refers to its function that can interact with other services and systems, and able to extend the system in different ways by adding software or hardware resources. In my assignment, I have to accurately define the RESTful API endpoints, that load image data, understand expected input, output format and any expected parameters. Adhere to required methods, in my case, it is ‘POST’ and enable thread. In the cloud, I have to clearly define the NodePort and public address, so my cluster can communicate with my instance, and extend the number of pods, so that my service can handle higher workload.

Concurrency:

In a distributed system, concurrency means my server can interact with multiple clients simultaneously, but there are some challenges such as race condition and inconsistent data. In client script, we used ThreadPoolExecutor, it helps manage thread pools and provide a way to achieve concurrency in the code. In the cloud instance, I created a service, which is also a cluster, that let my pods work together to perform tasks, it helps concurrency by distributing workload and improving the overall performance. Therefore, It has load balancing function, that distribute incoming tasks among multiple pods, this load balancing helps to utilize the pods and process requests concurrently.