Tung-I Chen 33609886; Pranav Jain 33982032

Lab3
Evaluation Document

How to deploy our application on AWS

1. First we created an `m5a.large` EC2 instance in the `us-east-1` region on AWS

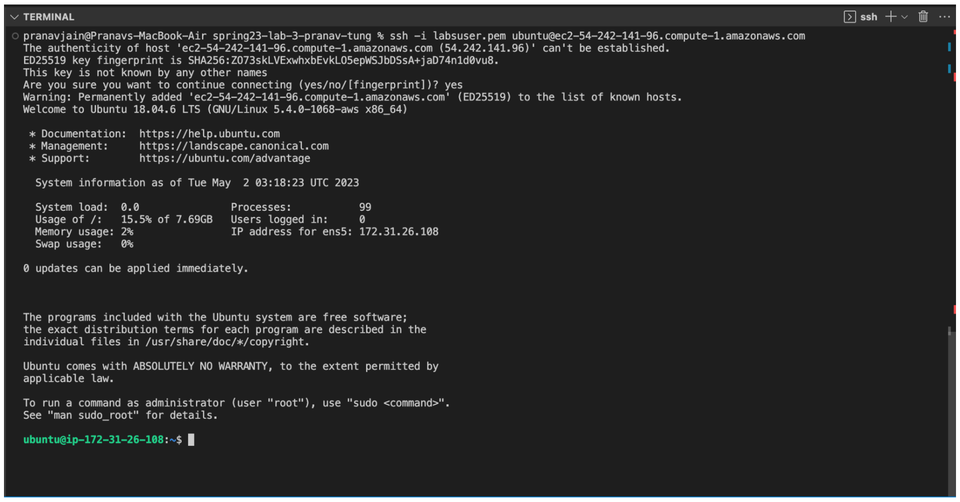
- $ aws ec2 run-instances --image-id ami-0d73480446600f555 --instance-type m5a.large --key-name vockey > instance.json

2. Checking status of our instance to find the public DNS name

- aws ec2 describe-instances --instance-id {instance_ID}
- For instance, "PublicDnsName" : ec2-54-242-141-96.compute-1.amazonaws.com

3. Access the instance via ssh

- chmod 400 labuser.pem
- aws ec2 authorize-security-group-ingress --group-name default --protocol tcp --port 22 --cidr 0.0.0.0/0
- aws ec2 authorize-security-group-ingress --group-name default --protocol tcp --port 8080 --cidr 0.0.0.0/0 (for our frontend service, which uses port=8080)
- ssh -i labsuser.pem ubuntu@ec2-54-242-141-96.compute-1.amazonaws.com



```
pranavjain@Pranavs-MacBook-Air spring23-lab-3-pranav-tung % ssh -i labsuser.pem ubuntu@ec2-54-242-141-96.compute-1.amazonaws.com
The authenticity of host 'ec2-54-242-141-96.compute-1.amazonaws.com (54.242.141.96)' can't be established.
ED25519 key fingerprint is SHA256:ZO73skLVExwhxbEvkLO5epWSJbDSsA+jaD74n1d0vu8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-242-141-96.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1068-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue May  2 03:18:23 UTC 2023

  System load:  0.0               Processes:           99
  Usage of /:   15.5% of 7.69GB   Users logged in:     0
  Memory usage: 2%                IP address for ens5: 172.31.26.108
  Swap usage:   0%

0 updates can be applied immediately.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-26-108:~$ 
```

4. Inside the instance, install the required packages:

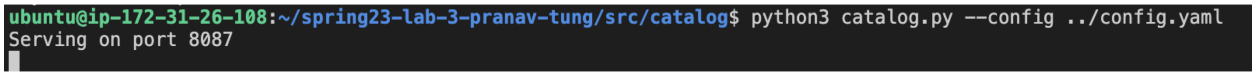
- $ sudo apt-get install software-properties-common
- $ sudo apt-add-repository universe
- $ sudo apt-get update
- $ sudo apt-get install python-pip
- $ pip install requests
- $ pip install pyyaml

5. Cloning our repo on the instance:

- $ git clone https://github.com/umass-cs677-current/spring23-lab-3-pranav-tung.git

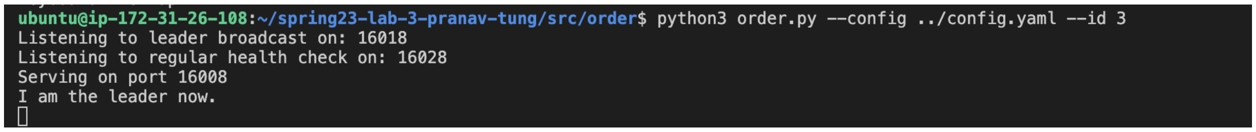
6. Running all the five services:

- python3 catalog.py --config ../config.yaml



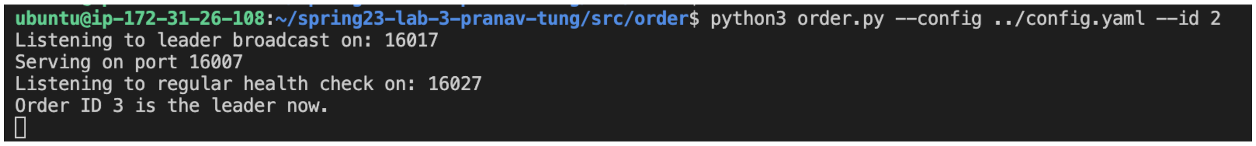
```
ubuntu@ip-172-31-26-108:~/spring23-lab-3-pranav-tung/src/catalog$ python3 catalog.py --config ../config.yaml
Serving on port 8087
```

- python3 order.py --config ../config.yaml --id 3



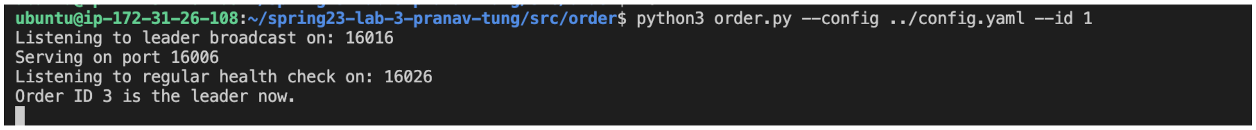
```
ubuntu@ip-172-31-26-108:~/spring23-lab-3-pranav-tung/src/order$ python3 order.py --config ../config.yaml --id 3
Listening to leader broadcast on: 16018
Listening to regular health check on: 16028
Serving on port 16008
I am the leader now.
```

- python3 order.py --config ../config.yaml --id 2



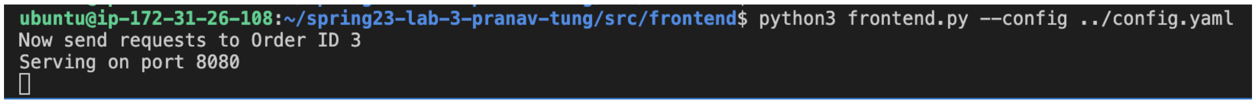
```
ubuntu@ip-172-31-26-108:~/spring23-lab-3-pranav-tung/src/order$ python3 order.py --config ../config.yaml --id 2
Listening to leader broadcast on: 16017
Serving on port 16007
Listening to regular health check on: 16027
Order ID 3 is the leader now.
```

- python3 order.py --config ../config.yaml --id 1



```
ubuntu@ip-172-31-26-108:~/spring23-lab-3-pranav-tung/src/order$ python3 order.py --config ../config.yaml --id 1
Listening to leader broadcast on: 16016
Serving on port 16006
Listening to regular health check on: 16026
Order ID 3 is the leader now.
```

- python3 frontend.py --config ../config.yaml



```
ubuntu@ip-172-31-26-108:~/spring23-lab-3-pranav-tung/src/frontend$ python3 frontend.py --config ../config.yaml
Now send requests to Order ID 3
Serving on port 8080
```

7. Now you can run concurrent clients locally:

- $ ./run_multi_client.sh {host_name} {frontend_port} {n_request} (probability)
- E.g.,
  $ ./run_multi_client.sh ec2-54-242-141-96.compute-1.amazonaws.com 8080 500 0.2

Evaluation results and plots

First, we compared the average latency of order and lookup requests over different follow-up-trade-request probability (from 0 to 80%).

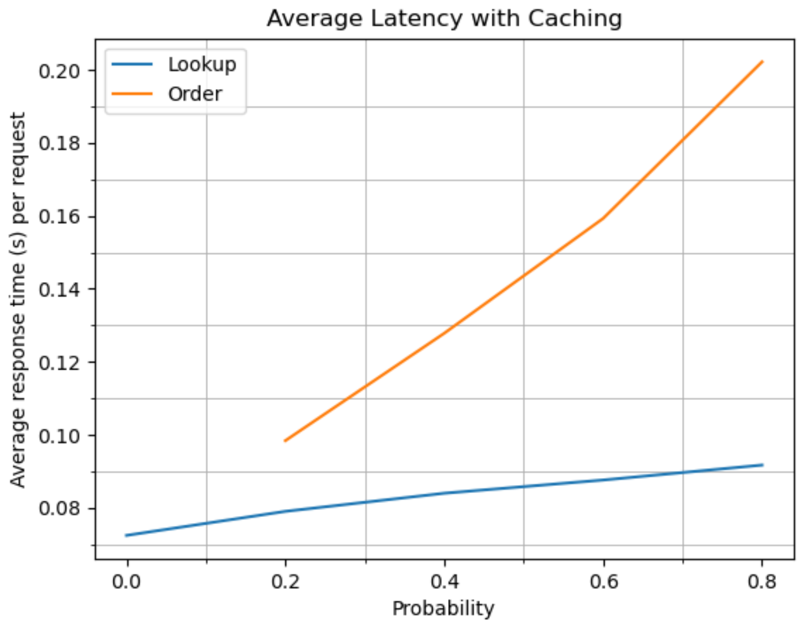


Fig1. Latency comparison between order and lookup requests with caching turning on

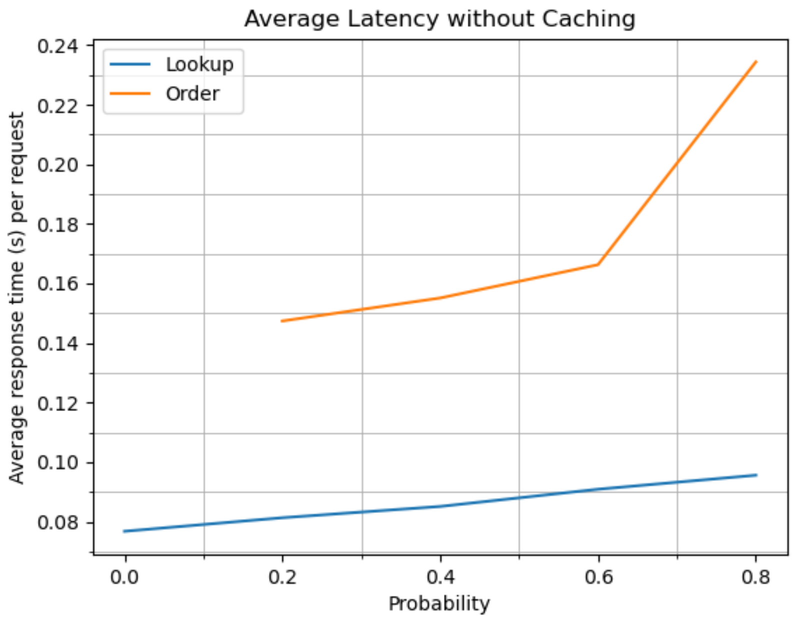


Fig2. Latency comparison between order and lookup requests with caching turning off

To begin with, upon analyzing the graphical representation presented in Figures 1 and 2, it is evident that the processing time required for fulfilling order requests is comparatively longer than that of lookup requests. This is attributed to the fact that order requests necessitate more utilization of system resources, resulting in a higher number of communications between the services involved. Additionally, it is noteworthy that an increase in the probability of follow-up trading requests leads to a substantial rise in the processing time required for both order and lookup requests.

We also tested the effectiveness of caching by comparing system performance with and without caching enabled. The following figures illustrate the results of our analysis.

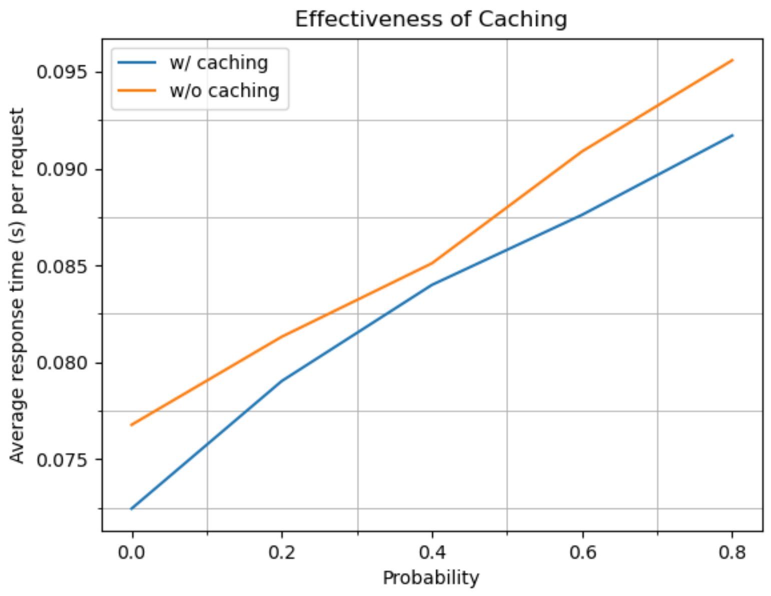


Fig3. Average lookup latency with and without caching enabled

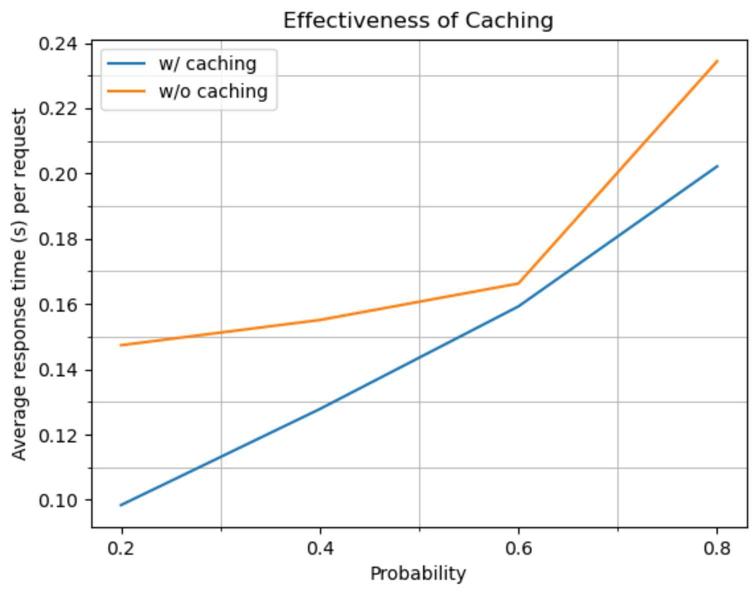


Fig4. Average order latency with and without caching enabled

The implementation of the front-end service caching mechanism has been shown to enhance the system's performance. Notably, both lookup and order requests have experienced performance improvements as a result of caching. With caching enabled, the catalog server's workload is alleviated, allowing for more efficient catalog database searching and writing.

In conclusion, our evaluation results show that order requests suffer a more significant latency than lookup requests, especially when the follow-up trading requests increase. Also, enabling the caching mechanism in the front-end service can be a practical approach to optimize the system's performance by reducing the workload on the catalog server and improving the processing speed of both lookup and order requests.