Lab 3 Evaluation Document
How to deploy our application on AWS

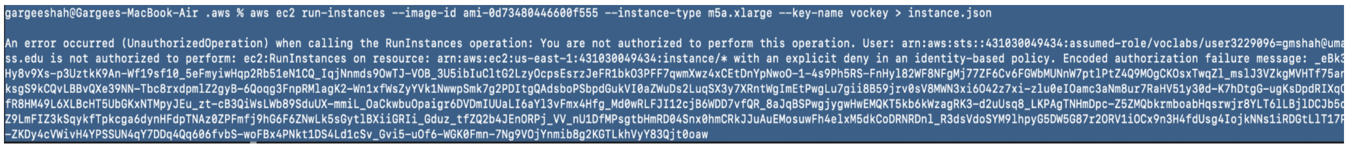
Steps taken for aws deployment:

Configuration:

- Started the lab and copied the credentials and labsuser.pem file in the aws directory.
- aws configure

Running Instances:

1**.** First we created an \`t2.micro\` EC2 instance in the \`us-east-1\` region on AWS
Due to facing issues of authorization for the m5a.xlarge instance, we used t2.micro instance as per piazza post @443.



```
gargeeshah@Gargees-MacBook-Air .aws % aws ec2 run-instances --image-id ami-0d73480446600f555 --instance-type m5a.xlarge --key-name vockey > instance.json

An error occurred (UnauthorizedOperation) when calling the RunInstances operation: You are not authorized to perform this operation. User: arn:aws:sts::431030049434:assumed-role/voclabs/user3229096=gmshah@uma
ss.edu is not authorized to perform: ec2:RunInstances on resource: arn:aws:ec2:us-east-1:431030049434:instance/* with an explicit deny in an identity-based policy. Encoded authorization failure message: _eBk3
Hy8v9Xs-p3UztkK9An-Wf19sf10_5eFmyiwHqp2Rb51eN1CQ_IqjNnmds9OwTJ-VOB_3U5ibIuCltG2LzyOcpsEsrzJeFR1bkO3PFF7qwmXwz4xCEtDnYpNwoO-1-4s9Ph5RS-FnHyl82WF8NFgMj77ZF6Cv6FGWbMUNnW7ptlPtZ4Q9MOgCKOsxTwqZl_mslJ3VZkgMVHTf75ar
ksgS9kCQvLBBvQXe39NN-Tbc8rxdpmlZ2gyB-6Qoqg3FnpRMlagK2-Wn1xfWsZyYVk1NwwpSmk7g2PDItgQAdsboPSbpdGukVI0aZWuDs2LuqSX3y7XRntWgImEtPwgLu7gii8B59jrv0sV8MWN3xi6O42z7xi-zlu0eIOamc3aNm8ur7RaHV51y30d-K7hDtgG-ugKsDpdRIXqC
fR8HM49L6XLBcHT5UbGKxNTMpyJEu_zt-cB3QiWsLWb89SduUX-mmiL_OaCkwbuOpaigr6DVDmIUUaLI6aYl3vFmx4Hfg_Md0wRLFJI12cjB6WDD7vfQR_8aJqBSPwgjygwHwEMQKT5kb6kWzagRK3-d2uUsq8_LKPAgTNHmDpc-Z5ZMQbkrmboabHqsrwjr8YLT6lLBjlDCJb5c
Z9LmFIZ3kSqykfTpkcga6dynHFdpTNAz0ZPFmfj9hG6F6ZNwLk5sGytlBXiiGRIi_Gduz_tfZQ2b4JEnORPj_VV_nU1DfMPsgtbHmRD04Snx0hmCRkJJuAuEMosuwFh4elxM5dkCoDRNRDnl_R3dsVdoSYM9lhpyG5DW5G87r2ORV1iOCx9n3H4fdUsg4IojkNNs1iRDGtLlT17F
-ZKDy4cVWivH4YPSSUN4qY7DDq4Qq606fvbS-woFBx4PNkt1DS4Ld1cSv_Gvi5-uOf6-WGK0Fmn-7Ng9VOjYnmib8g2KGTLkhVyY83Qjt0oaw
```

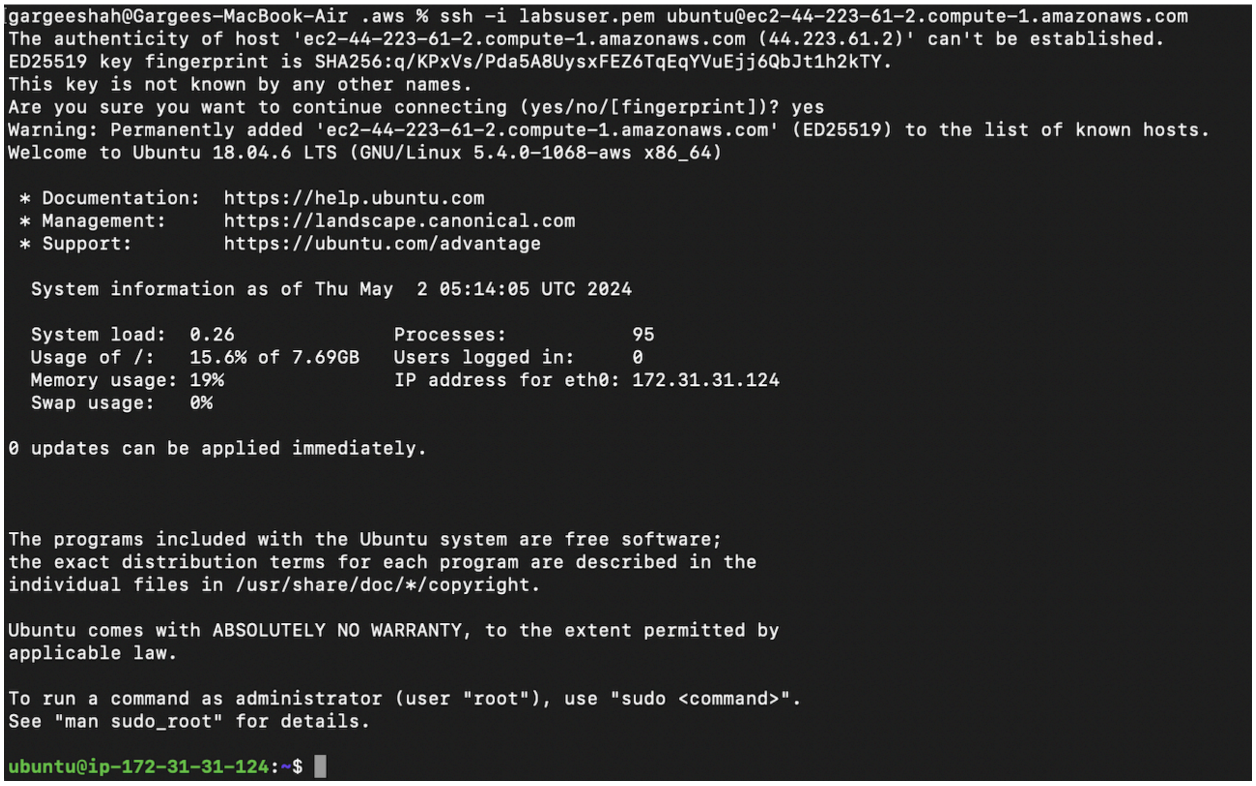
- $ aws ec2 run-instances --image-id ami-0d73480446600f555 --instance-type t2.micro --key-name vockey > instance.json

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

- aws ec2 describe-instances --instance-id {instance_ID}
  For instance, “PublicDnsName” : ec2-44-223-61-2.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 4773 --cidr 0.0.0.0/0 (for our frontend service, which uses port=4773)
- ssh -i labsuser.pem ubuntu@ec2-44-223-61-2.compute-1.amazonaws.com



```
gargeeshah@Gargees-MacBook-Air .aws % ssh -i labsuser.pem ubuntu@ec2-44-223-61-2.compute-1.amazonaws.com
The authenticity of host 'ec2-44-223-61-2.compute-1.amazonaws.com (44.223.61.2)' can't be established.
ED25519 key fingerprint is SHA256:q/KPxVs/Pda5A8UysxFEZ6TqEqYVuEjj6QbJt1h2kTY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-223-61-2.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 Thu May  2 05:14:05 UTC 2024

  System load:  0.26              Processes:           95
  Usage of /:   15.6% of 7.69GB   Users logged in:     0
  Memory usage: 19%               IP address for eth0: 172.31.31.124
  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-172-31-31-124:~$
```

4. Inside the instance, install the required packages:

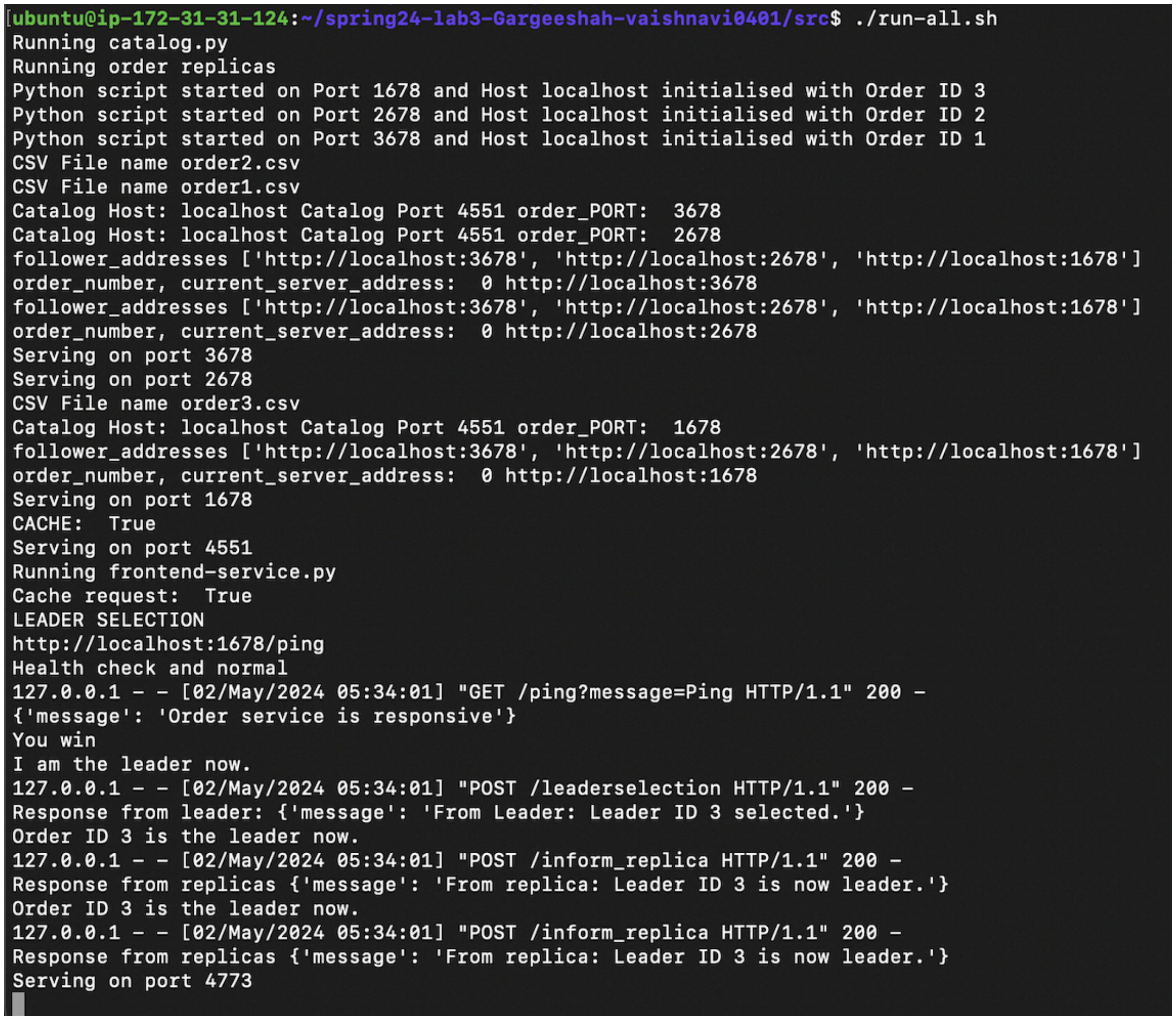
- sudo apt-get install software-properties-common
- sudo apt-add-repository universe
- sudo apt-get update
- sudo apt-get install python3-pip
- pip3 install requests
- pip3 install python-dotenv
- pip3 install pandas

5. Cloning our repo on the instance:

- Git clone https://github.com/umass-cs677-current/spring24-lab3-Gargeeshah-vaishnavi0401.git
- cd /spring24-lab3-Gargeeshah-vaishnavi0401/src

6. Running all the five services:

- chmod +x run-all.sh
- ./run-all.sh



```
ubuntu@ip-172-31-31-124:~/spring24-lab3-Gargeeshah-vaishnavi0401/src$ ./run-all.sh
Running catalog.py
Running order replicas
Python script started on Port 1678 and Host localhost initialised with Order ID 3
Python script started on Port 2678 and Host localhost initialised with Order ID 2
Python script started on Port 3678 and Host localhost initialised with Order ID 1
CSV File name order2.csv
CSV File name order1.csv
Catalog Host: localhost Catalog Port 4551 order_PORT:  3678
Catalog Host: localhost Catalog Port 4551 order_PORT:  2678
follower_addresses ['http://localhost:3678', 'http://localhost:2678', 'http://localhost:1678']
order_number, current_server_address:  0 http://localhost:3678
follower_addresses ['http://localhost:3678', 'http://localhost:2678', 'http://localhost:1678']
order_number, current_server_address:  0 http://localhost:2678
Serving on port 3678
Serving on port 2678
CSV File name order3.csv
Catalog Host: localhost Catalog Port 4551 order_PORT:  1678
follower_addresses ['http://localhost:3678', 'http://localhost:2678', 'http://localhost:1678']
order_number, current_server_address:  0 http://localhost:1678
Serving on port 1678
CACHE:  True
Serving on port 4551
Running frontend-service.py
Cache request:  True
LEADER SELECTION
http://localhost:1678/ping
Health check and normal
127.0.0.1 - - [02/May/2024 05:34:01] "GET /ping?message=Ping HTTP/1.1" 200 -
{'message': 'Order service is responsive'}
You win
I am the leader now.
127.0.0.1 - - [02/May/2024 05:34:01] "POST /leaderselection HTTP/1.1" 200 -
Response from leader: {'message': 'From Leader: Leader ID 3 selected.'}
Order ID 3 is the leader now.
127.0.0.1 - - [02/May/2024 05:34:01] "POST /inform_replica HTTP/1.1" 200 -
Response from replicas {'message': 'From replica: Leader ID 3 is now leader.'}
Order ID 3 is the leader now.
127.0.0.1 - - [02/May/2024 05:34:01] "POST /inform_replica HTTP/1.1" 200 -
Response from replicas {'message': 'From replica: Leader ID 3 is now leader.'}
Serving on port 4773
```

7. Now you can run concurrent clients locally:

- cd test/load_test
- Update *{host_name} {frontend_port}* information in .env
- chmod +x client5.sh
- ./client5.sh

Evaluation results and plots

First, we compared the average latency of order and lookup requests over different request probability (from 0 to 80%) for 5 clients for 500 requests running concurrently with caching turned on.

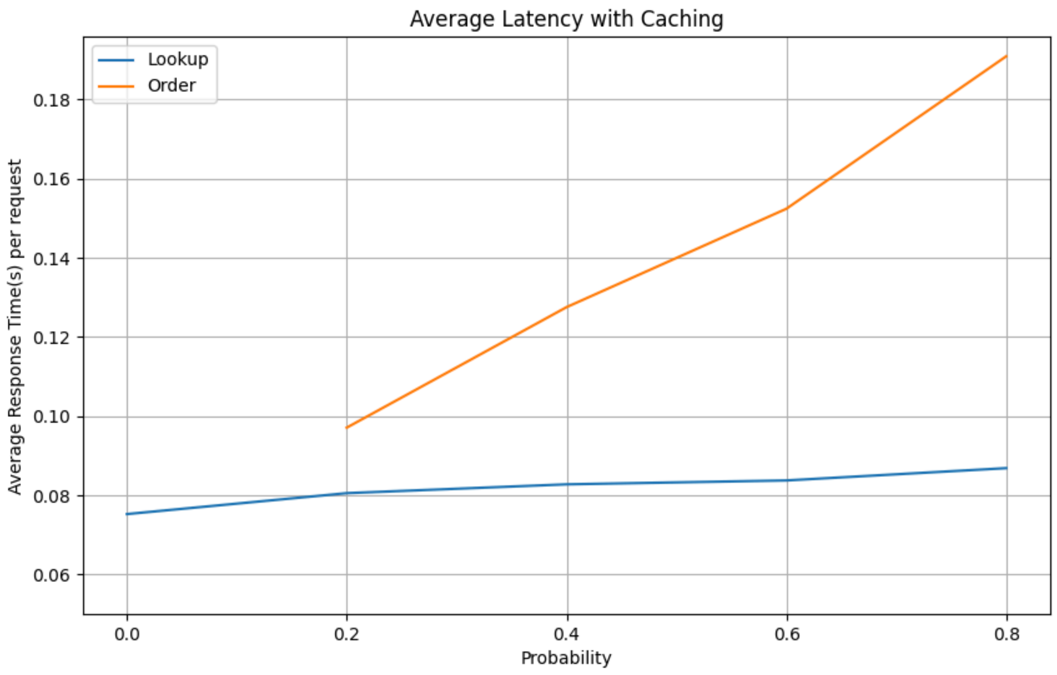


Fig1. Latency comparison between order and lookup requests with caching turning on

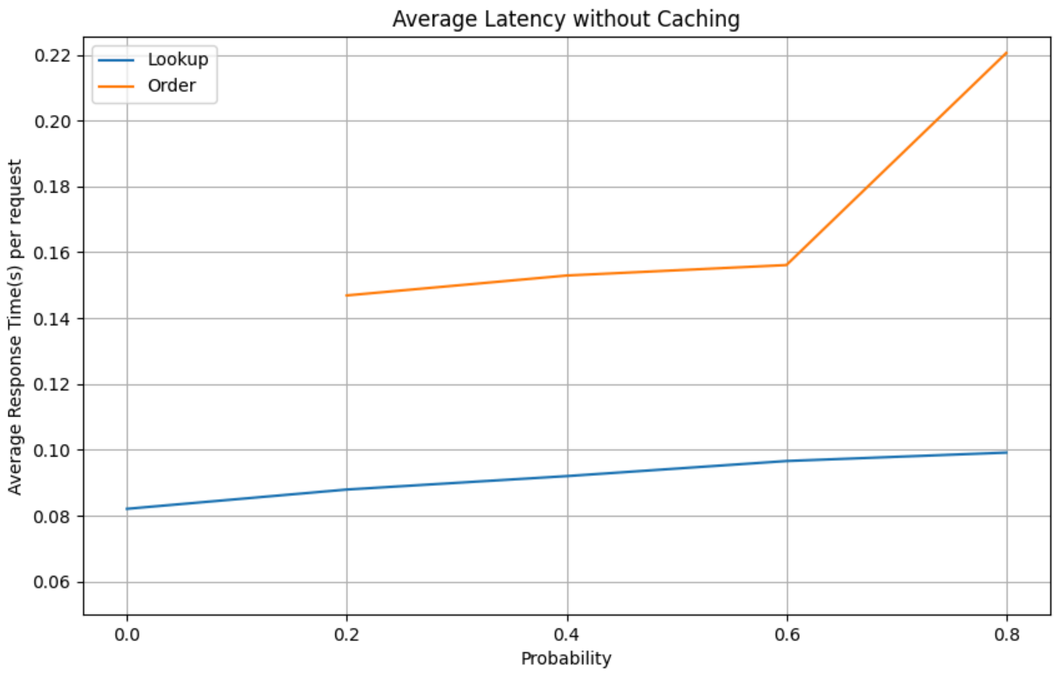


Fig2. Latency comparison between order and lookup requests with caching turning off

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 order 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.

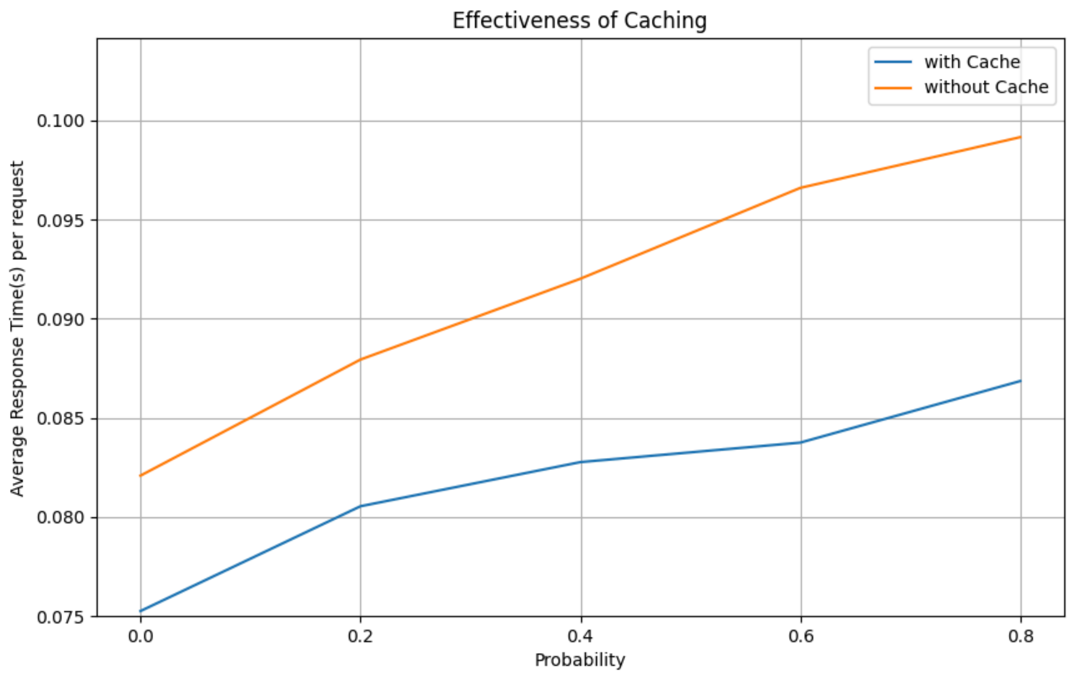


Fig3. Average lookup latency with and without caching enabled

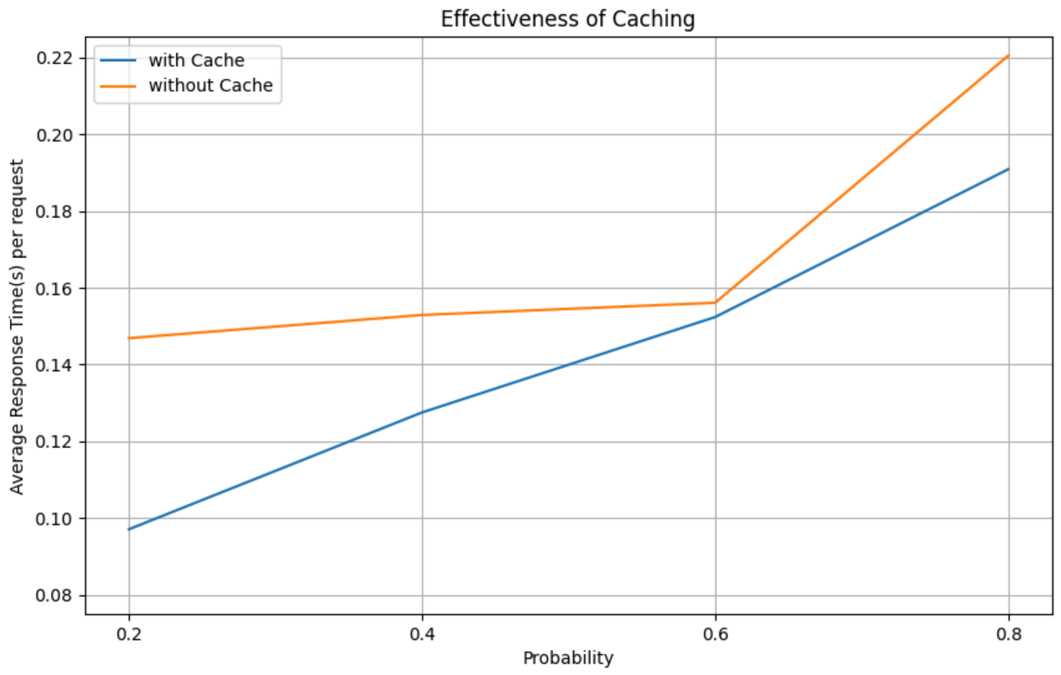


Fig4. Average order latency with and without caching enabled

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 order 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.