Evaluation Document

Deployment in AWS:

Instance creation	`aws ec2 run-instances --image-id ami-0d73480446600f555 --instance-type m5a.large --key-name vockey > instance.json`
Set permission for PEM key	`chmod 400 labsuser.pem`
SSH into created instance	`ssh -i labsuser.pem ubuntu@ec2-54-211-92-20.compute-1.amazonaws.com`
Make directory for copying source files	`mkdir lab3`
Install pip package to install other requirements	`sudo apt update` `sudo apt install python3-pip`

Copy required files into the created folder from local terminal using scp	`scp -i labsuser.pem "Documents/CS 677/Repositories/spring23-lab-3-lab3-vidhya-vaishnavi/requirements.txt" ubuntu@ec2-54-211-92-20.compute-1.amazonaws.com:lab3`
	
	`scp -i labsuser.pem -r "Documents/CS 677/Repositories/spring23-lab-3-lab3-vidhya-vaishnavi/src" ubuntu@ec2-54-211-92-20.compute-1.amazonaws.com:lab3`

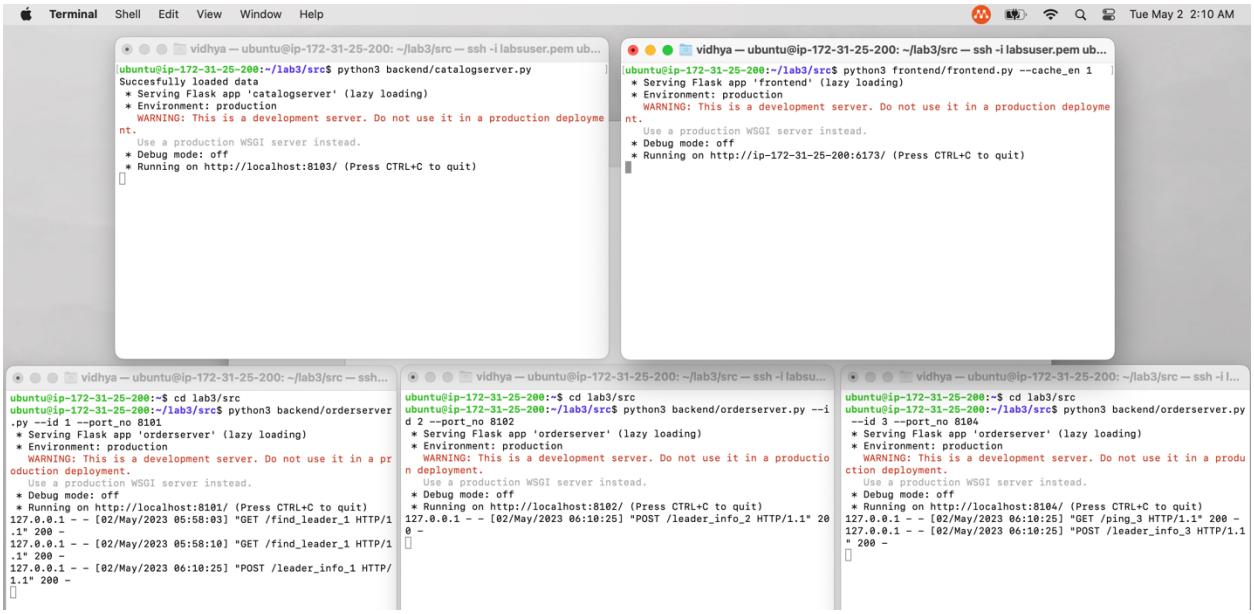
Install requirements from requirements.txt	`pip3 install -r requirements.txt`
Run servers by opening 5 terminals and ssh into the instance (Step 3) and run the servers.	`cd lab3/src` `python3 backend/catalogserver.py`
	`cd lab3/src` `python3 backend/orderserver.py --id 1 --port_no 8101`
	`cd lab3/src` `python3 backend/orderserver.py --id 2 --port_no 8102`
	`cd lab3/src` `python3 backend/orderserver.py --id 3 --port_no 8104`
	`cd lab3/src` `python3 frontend/frontendserver.py --cache_en 1`
Run 5 clients from local terminal opened at src folder of the project	`python3 client/client.py`
Stop and terminate instance	`aws ec2 terminate-instances --instance-ids i-097f3e4aa3877d9b4`

Note:

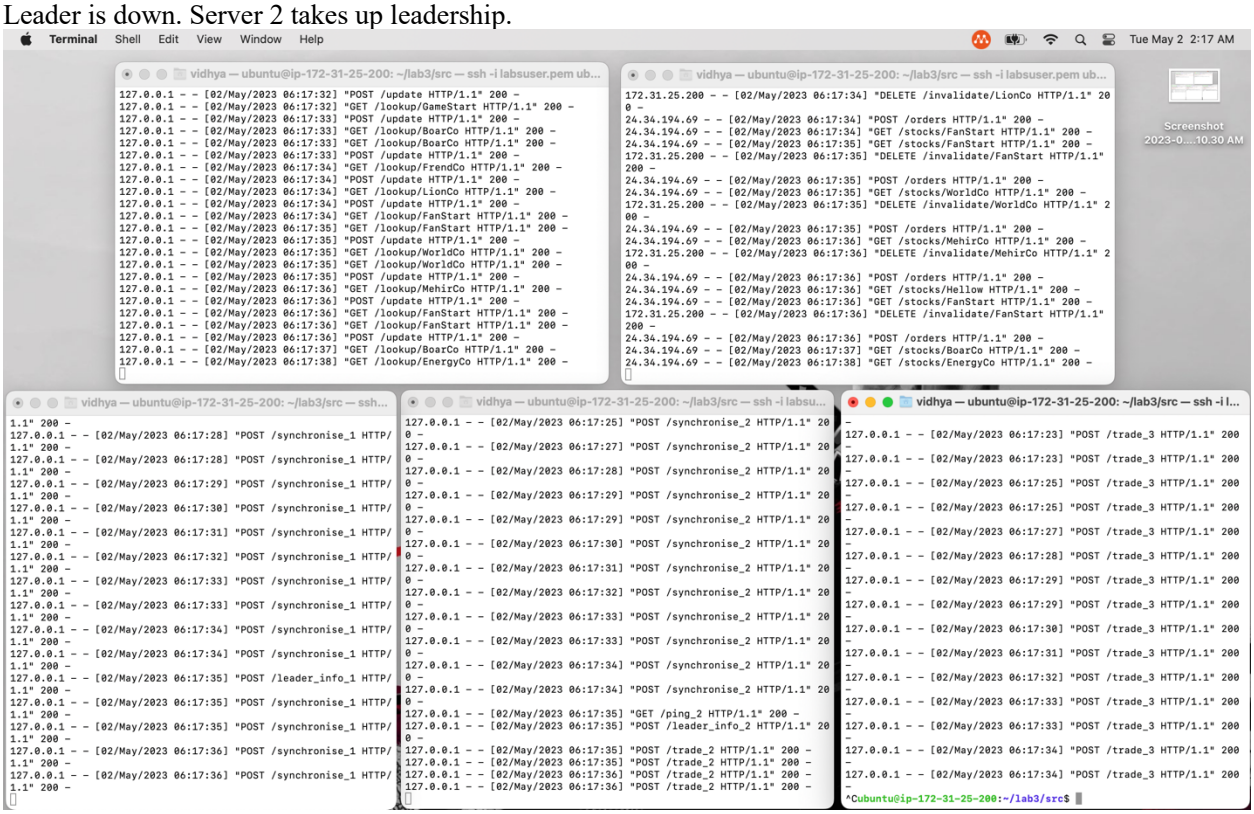
- Added a security rule to the ec2 instance to allow inbound traffic from local machine's ip.
- Added the public ip address of the ec2 instance in the local client to communicate with the servers hosted in the instance.

Working snapshots of application in AWS EC2 instance:

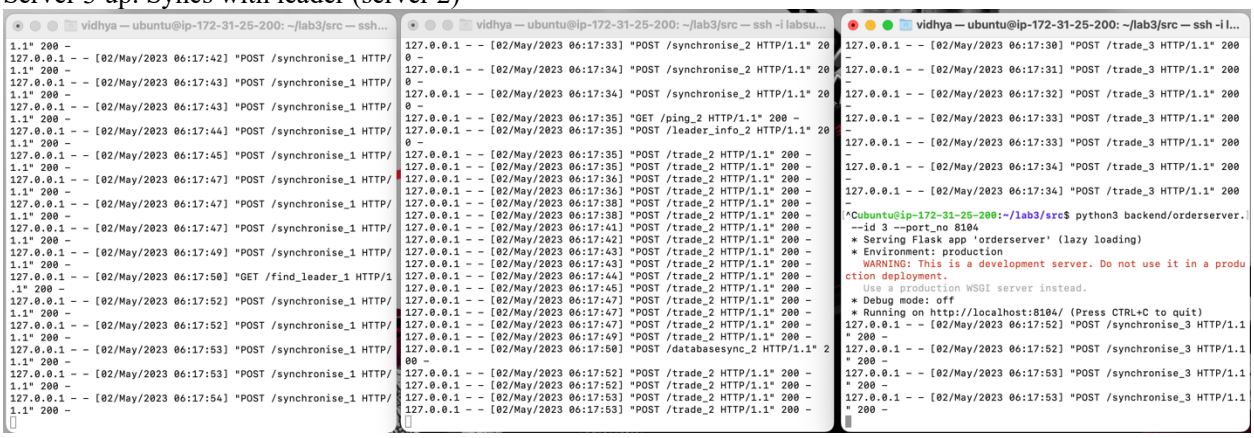
Servers started. Leader selected as 3.



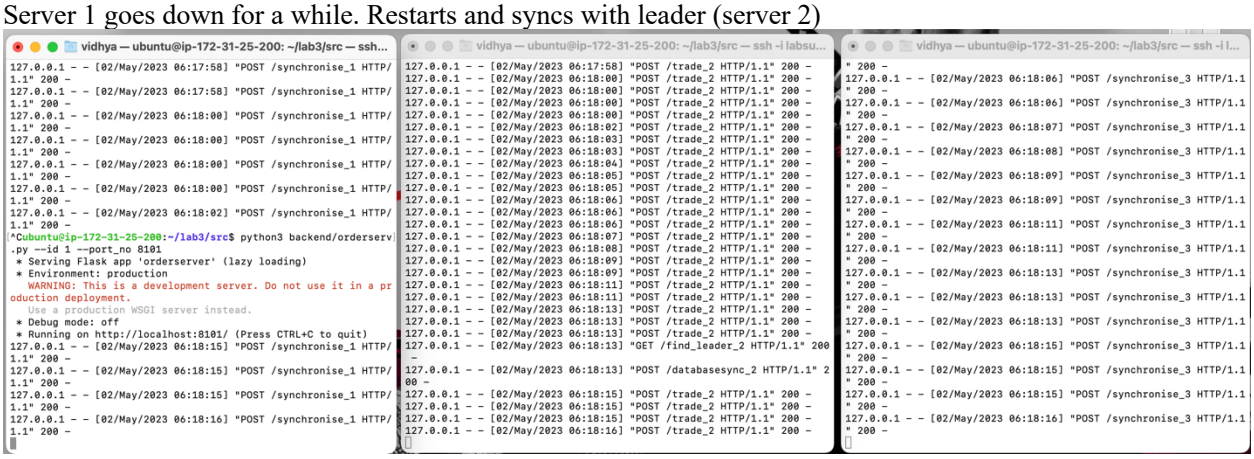
Leader is down. Server 2 takes up leadership.



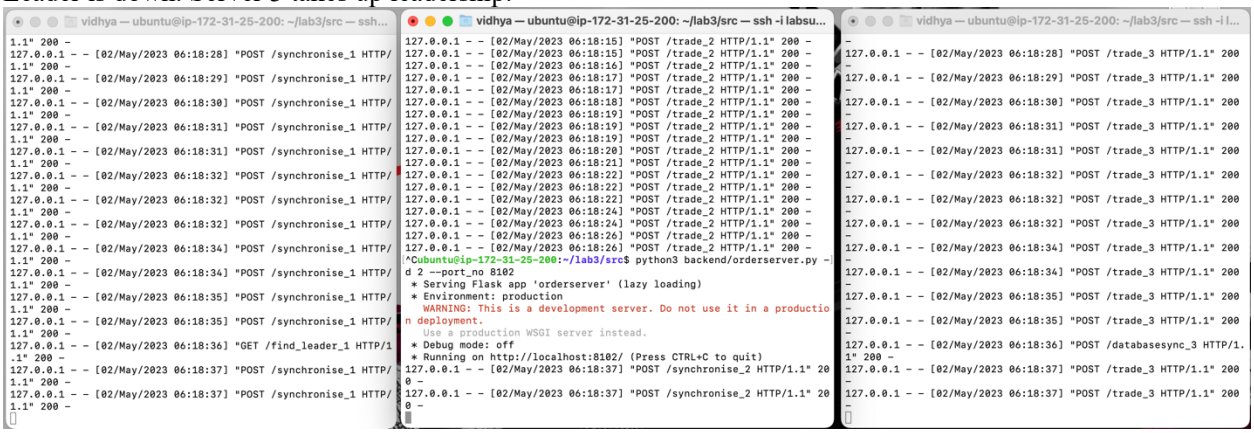
Server 3 up. Syncs with leader (server 2)



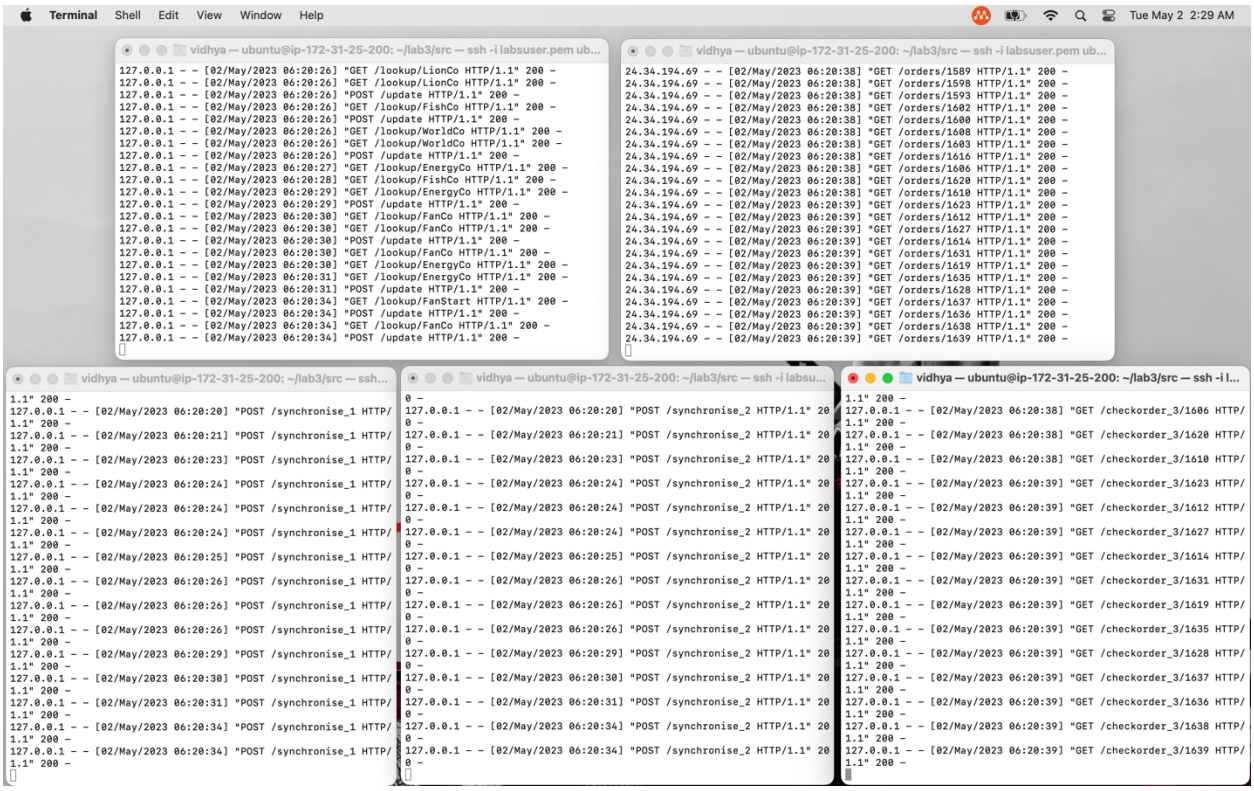
Server 1 goes down for a while. Restarts and syncs with leader (server 2)



Leader is down. Server 3 takes up leadership.



Finishes all transactions. Clients check transactions with the leader. Frontend reflects the checks.

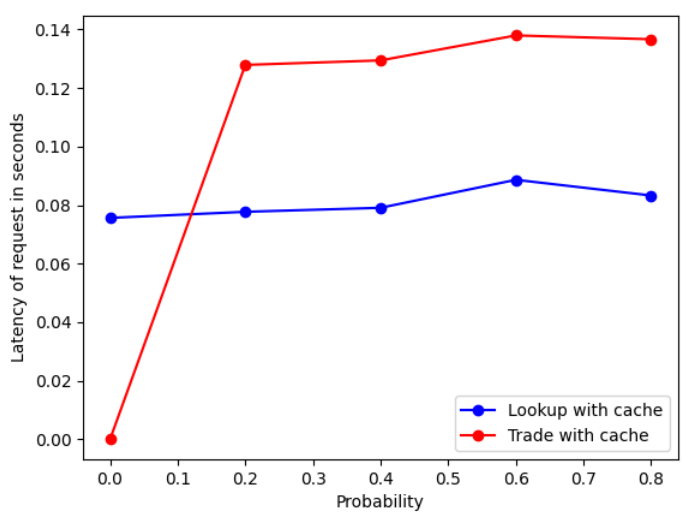


Measurement Results:

Measured the latency seen by each client for different types of requests. Changed the probability p of a follow up trade request from 0 to 80%, with an increment of 20%, and record the result for each p setting.
The files, Output_with_cache.txt and Output_without_cache.txt in docs folder has detailed output values of each client and the average latency values of different type of requests.

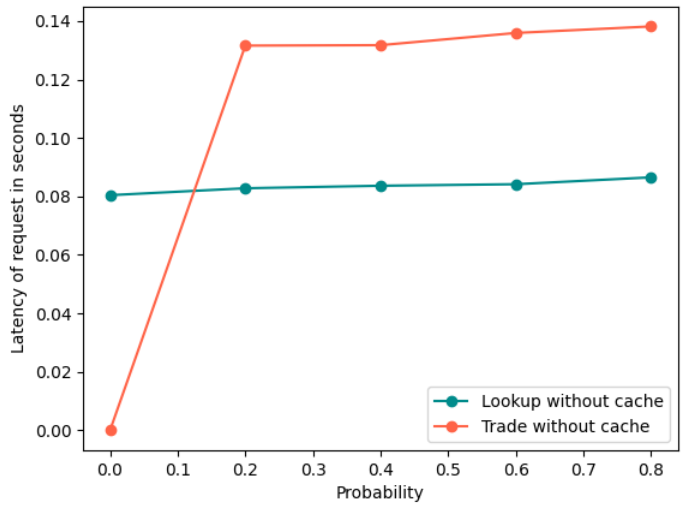
With Caching:

Below is the plot showing the values of p on the x-axis and the latency of different types of requests on the y-axis with caching turned on.



Without Caching:

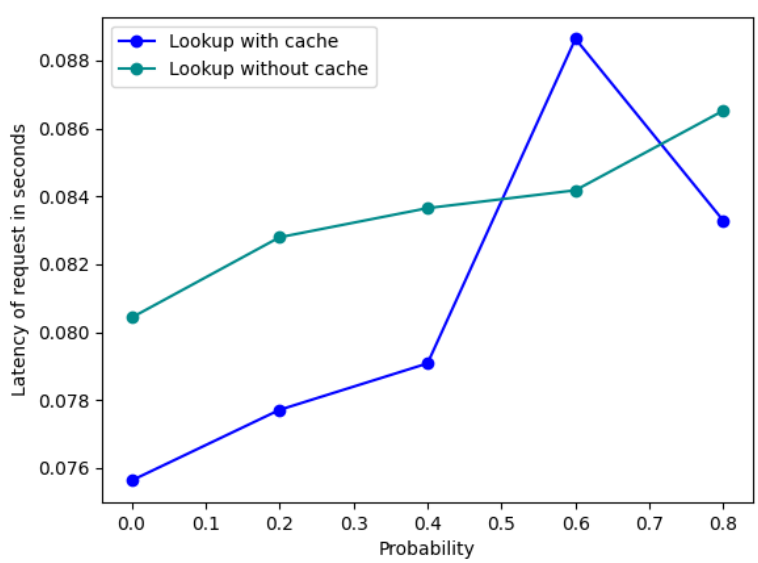
Below is the plot showing the values of p on the x-axis and the latency of different types of requests on the y-axis with caching turned off.



Lookup request with and without caching:

Below is the plot that clearly shows the difference in the latency of lookup request with and without cache. In majority of the cases, lookup requests with cache are taking lesser time in comparison with lookup requests without cache.

The difference between the latency of lookup request with and without cache decreases as the probability increases. It is because, as the number of trades triggered per lookup request is high, the number of invalidation requests sent to front end server is also high. This in-turn removes the corresponding stock from the cache. This results in multiple addition and deletion from cache which reduces the impact of cache as the probability increases.



Replication and Fault Tolerance testing:

Repeatedly simulated crash failures by killing both leader order service and other order service replicas and while the clients were running, and then bring it back online after some time as seen in the snapshots in the beginning of the document.

Can the clients notice the failures? (Either during order requests or the final order checking phase)

- No. The failures of order service replicas are not noticeable for the clients. The clients only prints out the time taken for all the transactions to complete. It is oblivious to the failures of the order servers.

Do all the order service replicas end up with the same database file?

- Yes. All the order service replicas ended up having the same database file as shown below.