

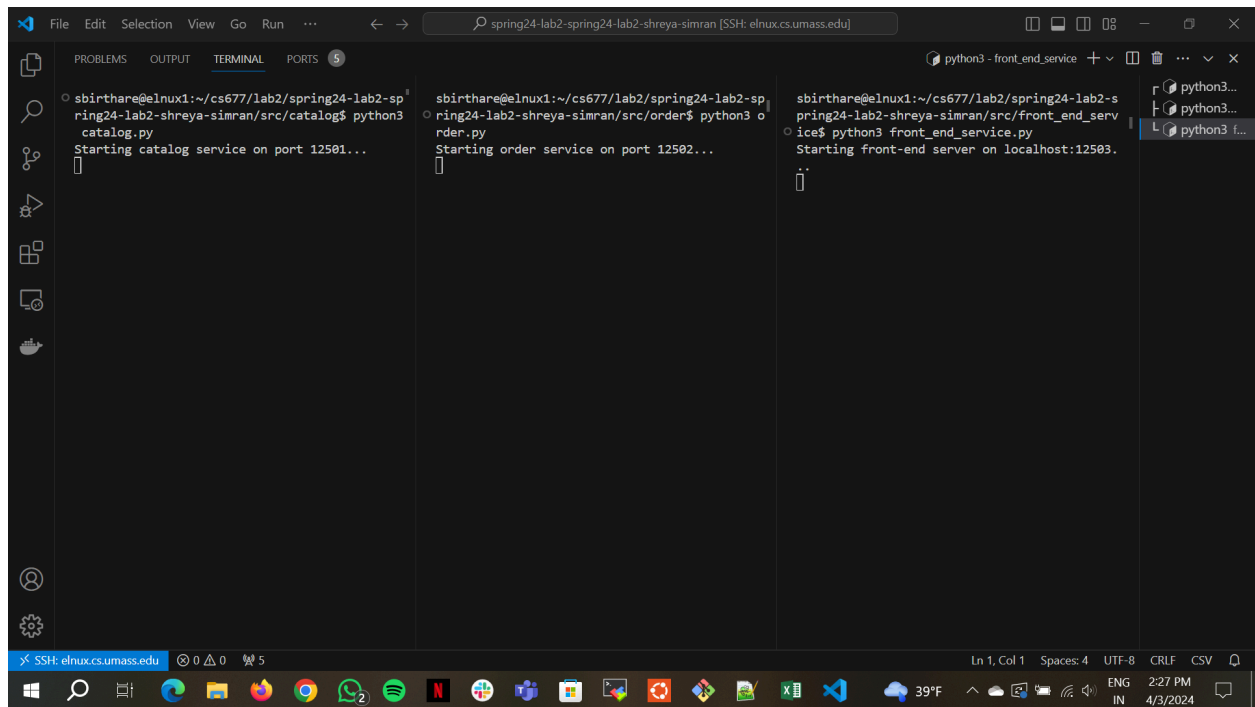
## LATENCY EVALUATION

For latency calculation we measure the start time just as the session starts and measure the end time once our queries are all handled, before closing the session. We do our calculations for two scenarios: (a) When we run microservices and clients on different setups without docker and (b) When we run microservices and clients on different setups using docker. We vary our clients from 1 to 5 for both the scenarios and take the average of the response time and plot avg latency vs number of clients graphs for calculations. See the latency evaluation document for our calculations.

### RUNNING APPLICATION WITHOUT DOCKER

We run our microservices on the edlab machine and client on local setup. While running the client, we pass the FRONTEND\_HOST environment variable as the ip address of the edlab setup.

Starting services on edlab:



```
spring24-lab2-spring24-lab2-shreya-simran [SSH: elnux.cs.umass.edu]

PROBLEMS OUTPUT TERMINAL PORTS 5

sbirthare@elnux1:~/cs677/lab2/spring24-lab2-sp
ring24-lab2-shreya-simran/src/catalog$ python3
catalog.py
Starting catalog service on port 12501...

sbirthare@elnux1:~/cs677/lab2/spring24-lab2-sp
ring24-lab2-shreya-simran/src/order$ python3 o
rder.py
Starting order service on port 12502...

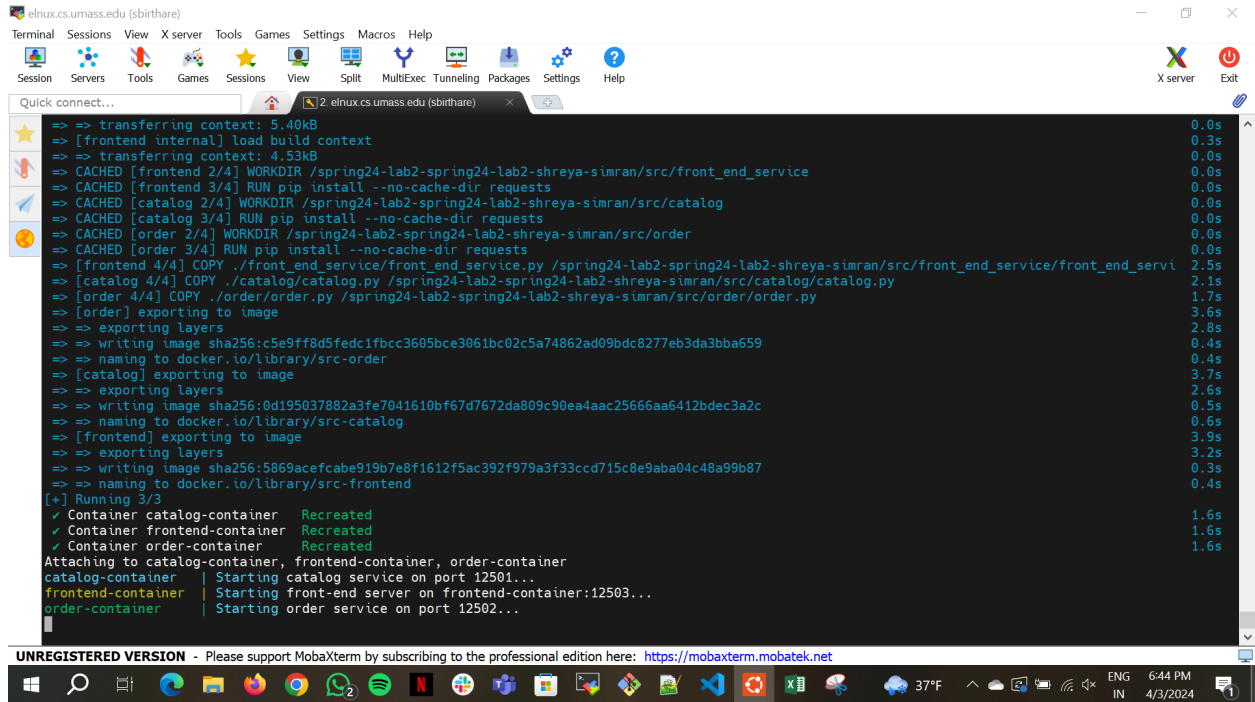
sbirthare@elnux1:~/cs677/lab2/spring24-lab2-s
pring24-lab2-shreya-simran/src/front_end_serv
ice$ python3 front_end_service.py
Starting front-end server on localhost:12503.

python3 - front_end_service + v
python3...
python3...
python3 f...
```

For output logs, pls see output/latencyEval/<file\_name\_no\_docker.txt>

### RUNNING APPLICATION WITH DOCKER

Starting docker on edlab using docker-compose.yml



For output logs, pls see output/latencyEval/<file\_name\_docker.txt>

### APPLICATION LATENCY:

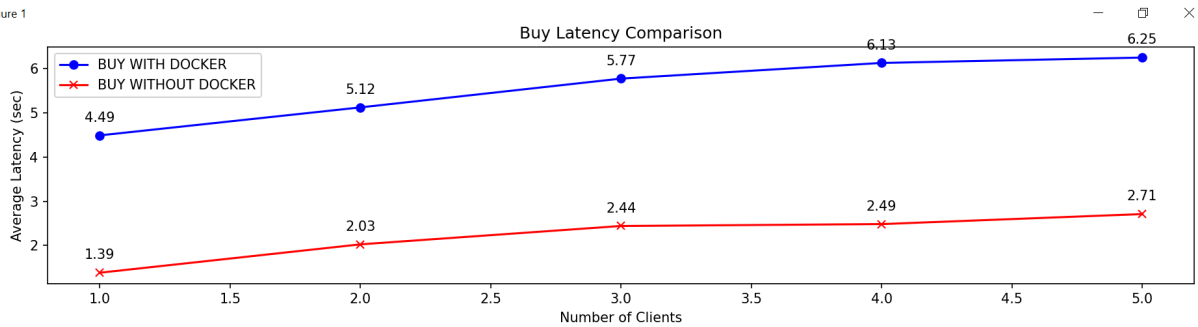
WITH DOCKER			WITHOUT DOCKER		
Number of Clients	Average Latency (sec)		Number of Clients	Average Latency (sec)	
1	4.490663767		1	1.385770082	
2	5.123154521		2	2.027086258	
3	5.774055322		3	2.442425807	
4	6.12833333		4	2.485036421	
5	6.24968853		5	2.711467683	

### QUERY LATENCY:

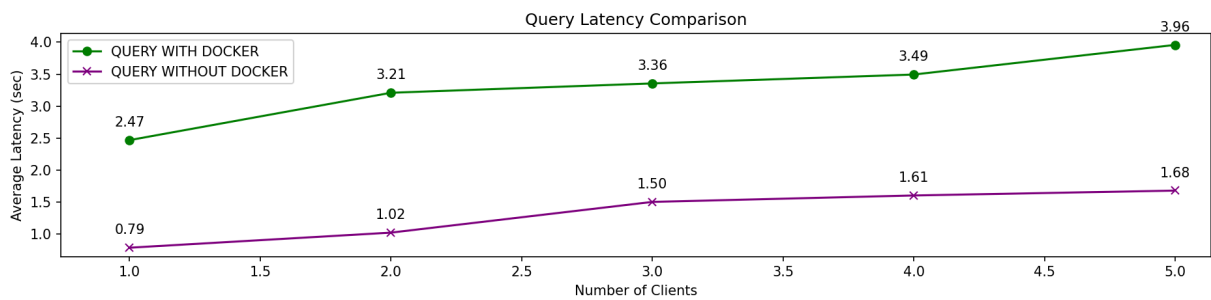
Here we comment out the code to place order in client.py and run only query code. See output/latencyEval for complete logs of the command prompt/terminal.

QUERY WITH DOCKER			QUERY WITHOUT DOCKER		
Number of Clients	Average Latency (sec)		Number of Clients	Average Latency (sec)	
1	2.469878435		1	0.788919687	
2	3.209934592		2	1.024381161	
3	3.355884155		3	1.504482269	
4	3.494788647		4	1.605131483	
5	3.957850981		5	1.681178451	

Figure 1



## BUY LATENCY:



## QUESTIONS:

- Does the latency of the application change with and without Docker containers? Did virtualization add any overheads?

Yes, latency of the application changes with and without Docker. With Docker, there is additional overhead and the latency increases. This can be seen in the table and the graphs above. This is expected due to additional virtualization done by docker.

- How does the latency of the query compare to buy? Since buy requests involve all three microservices, while query requests only involve two microservices, does it impact the observed latency?

The latency of the queries are less when compared to buy in both the scenarios: with and without running on docker and this is expected also because in case of query we only deal with 2 microservices: frontend and the catalog. The client sends the query to the front end which then sends the query to catalog to fetch the details of the product. Where as in the case of placing an order, client sends query to front end, front end calls the order microservice to place the order and the order microservice calls the catalog microservice to update the quantities therefore taking more time than buy

- How does the latency change as the number of clients changes? Does it change for different types of requests?

As we increase the number of clients the latency increases as more users make requests thereby increasing the load on our servers. It is also seen that the buy latencies are more than the query latencies and these also increase as the number of requests increases. Depending on the type of request GET/POST for buy or query. Updation takes more time than just fetching the results simply.