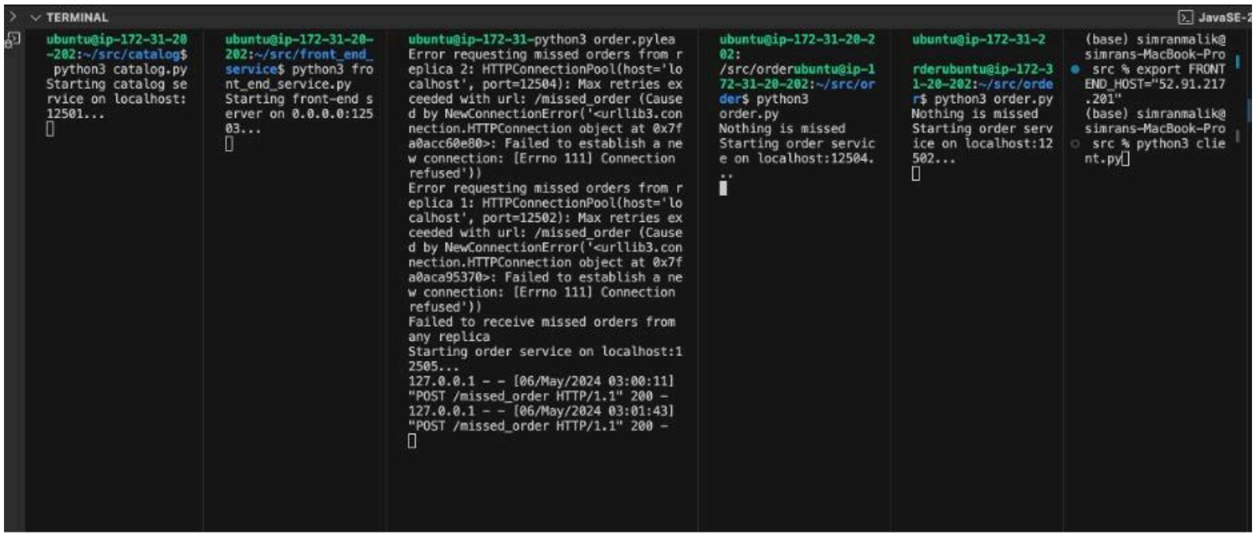
LATENCY EVALUATION DOCUMENT:

We run our 3 microservices and AWS and client on local. We run our client for ‘20’ iterations and vary the probability from 0 to 0.8 and measure latencies. We measure latency for 3 requests: product query, order buy and order get. We run 5 client instances concurrently on our local setup and the services on AWS to plot the graphs. We take the average of these latencies as latency is calculated for a single request so for 20 iterations and 5 clients we calculate the average via code (for 20 iterations) and manually (for 5 clients) and dump them in a table to plot the graphs.

Pls see spring24-lab3-spring24-lab3-shreya-simran\src\latencyCalc.ipynb for the code to plot graphs and calculate % improvement in latency using caching for our different requests for different values of p

Running services on AWS:



```
ubuntu@ip-172-31-20
-202:~/src/catalog$
 python3 catalog.py
Starting catalog se
rvice on localhost:
12501...
```

```
ubuntu@ip-172-31-20-
202:~/src/front_end_
service$ python3 fro
nt_end_service.py
Starting front-end s
erver on 0.0.0.0:125
03...
```

```
ubuntu@ip-172-31-python3 order.pylea
Error requesting missed orders from r
eplica 2: HTTPConnectionPool(host='lo
calhost', port=12504): Max retries ex
ceeded with url: /missed_order (Cause
d by NewConnectionError('<urllib3.con
nection.HTTPConnection object at 0x7f
a0acc60e80>: Failed to establish a ne
w connection: [Errno 111] Connection
refused'))
Error requesting missed orders from r
eplica 1: HTTPConnectionPool(host='lo
calhost', port=12502): Max retries ex
ceeded with url: /missed_order (Cause
d by NewConnectionError('<urllib3.con
nection.HTTPConnection object at 0x7f
a0aca95370>: Failed to establish a ne
w connection: [Errno 111] Connection
refused'))
Failed to receive missed orders from
any replica
Starting order service on localhost:1
2505...
127.0.0.1 - - [06/May/2024 03:00:11]
"POST /missed_order HTTP/1.1" 200 -
127.0.0.1 - - [06/May/2024 03:01:43]
"POST /missed_order HTTP/1.1" 200 -
```

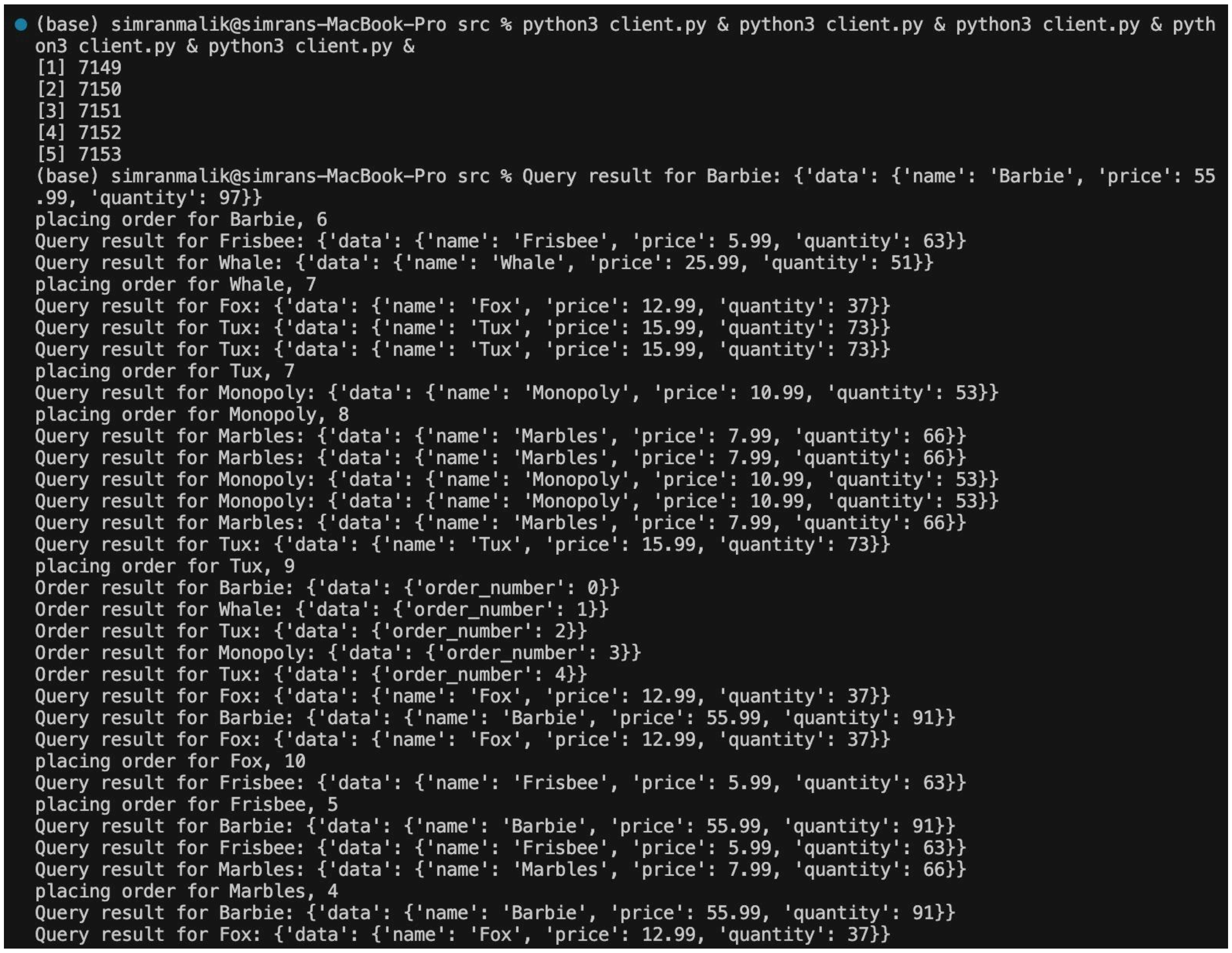
```
ubuntu@ip-172-31-20-2
02:
/src/orderubuntu@ip-1
72-31-20-202:~/src/or
der$ python3
order.py
Nothing is missed
Starting order servic
e on localhost:12504.
..
```

```
ubuntu@ip-172-31-2

rderubuntu@ip-172-3
1-20-202:~/src/orde
r$ python3 order.py
Nothing is missed
Starting order serv
ice on localhost:12
502...
```

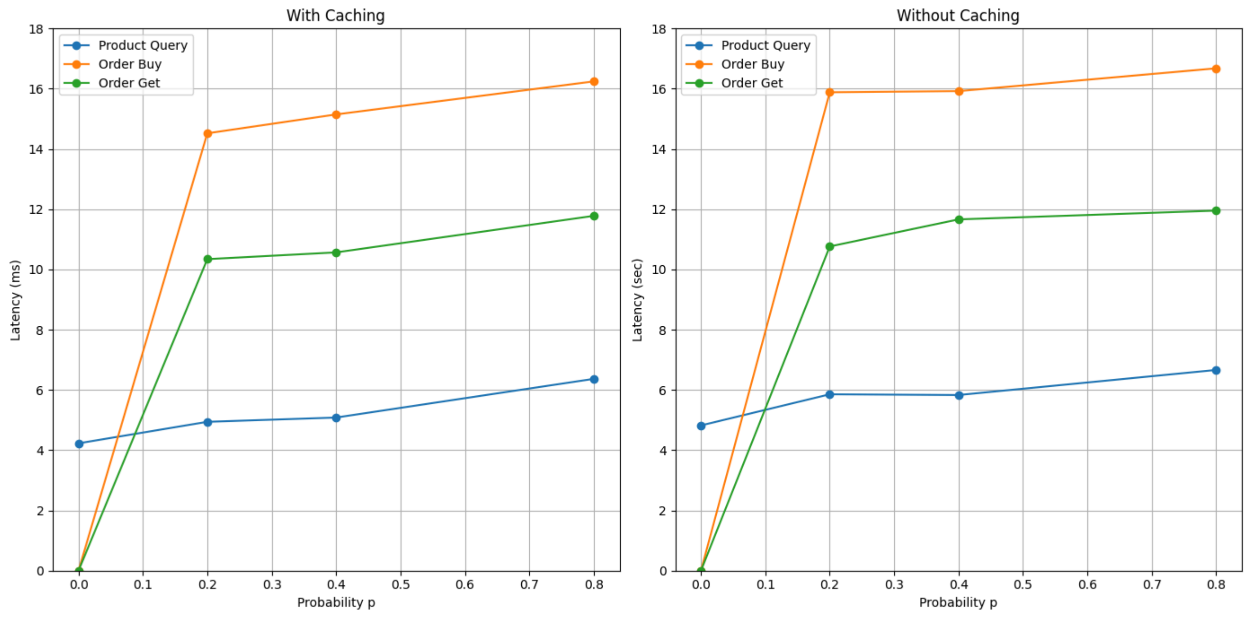
```
(base) simranmalik@
simrans-MacBook-Pro
 src % export FRONT
END_HOST="52.91.217
.201"
(base) simranmalik@
simrans-MacBook-Pro
 src % python3 clie
nt.py
```

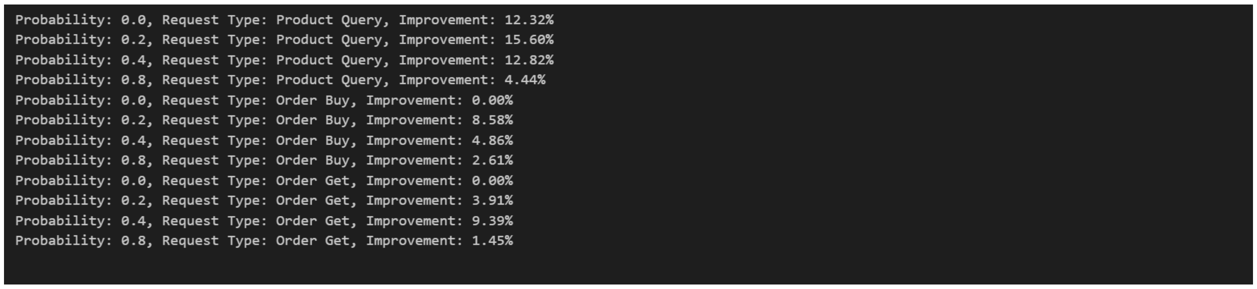
Running 5 clients on local:



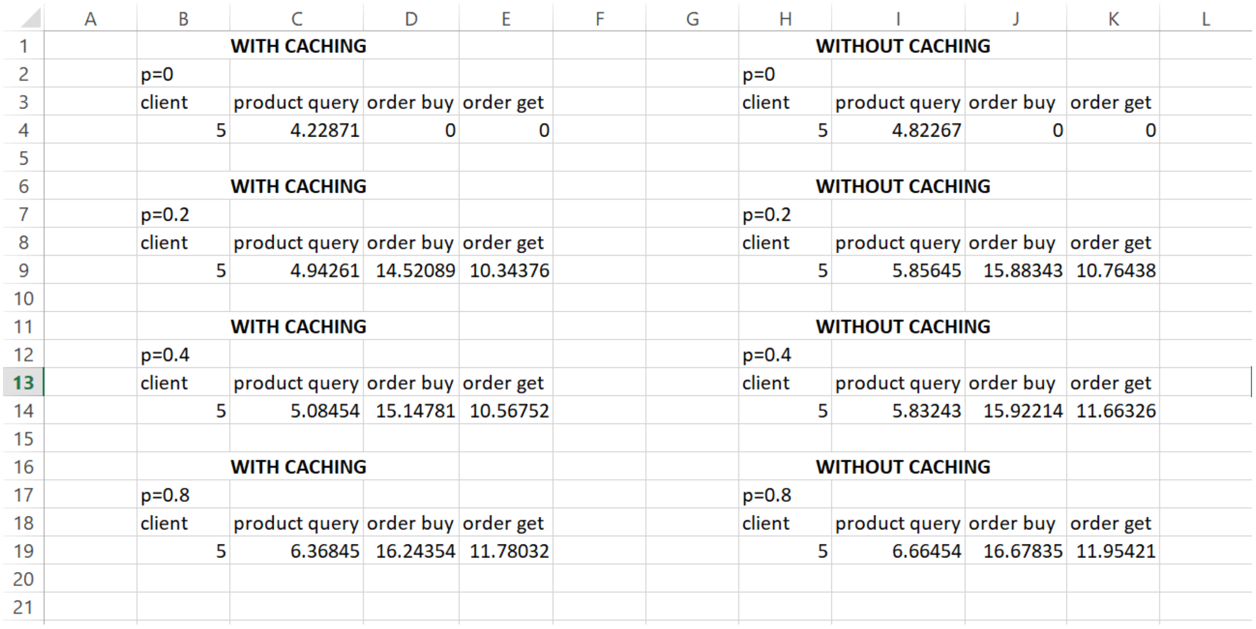
```
● (base) simranmalik@simrans-MacBook-Pro src % python3 client.py & python3 client.py & python3 client.py & pyth
  on3 client.py & python3 client.py &
  [1] 7149
  [2] 7150
  [3] 7151
  [4] 7152
  [5] 7153
  (base) simranmalik@simrans-MacBook-Pro src % Query result for Barbie: {'data': {'name': 'Barbie', 'price': 55
  .99, 'quantity': 97}}
  placing order for Barbie, 6
  Query result for Frisbee: {'data': {'name': 'Frisbee', 'price': 5.99, 'quantity': 63}}
  Query result for Whale: {'data': {'name': 'Whale', 'price': 25.99, 'quantity': 51}}
  placing order for Whale, 7
  Query result for Fox: {'data': {'name': 'Fox', 'price': 12.99, 'quantity': 37}}
  Query result for Tux: {'data': {'name': 'Tux', 'price': 15.99, 'quantity': 73}}
  Query result for Tux: {'data': {'name': 'Tux', 'price': 15.99, 'quantity': 73}}
  placing order for Tux, 7
  Query result for Monopoly: {'data': {'name': 'Monopoly', 'price': 10.99, 'quantity': 53}}
  placing order for Monopoly, 8
  Query result for Marbles: {'data': {'name': 'Marbles', 'price': 7.99, 'quantity': 66}}
  Query result for Marbles: {'data': {'name': 'Marbles', 'price': 7.99, 'quantity': 66}}
  Query result for Monopoly: {'data': {'name': 'Monopoly', 'price': 10.99, 'quantity': 53}}
  Query result for Monopoly: {'data': {'name': 'Monopoly', 'price': 10.99, 'quantity': 53}}
  Query result for Marbles: {'data': {'name': 'Marbles', 'price': 7.99, 'quantity': 66}}
  Query result for Tux: {'data': {'name': 'Tux', 'price': 15.99, 'quantity': 73}}
  placing order for Tux, 9
  Order result for Barbie: {'data': {'order_number': 0}}
  Order result for Whale: {'data': {'order_number': 1}}
  Order result for Tux: {'data': {'order_number': 2}}
  Order result for Monopoly: {'data': {'order_number': 3}}
  Order result for Tux: {'data': {'order_number': 4}}
  Query result for Fox: {'data': {'name': 'Fox', 'price': 12.99, 'quantity': 37}}
  Query result for Barbie: {'data': {'name': 'Barbie', 'price': 55.99, 'quantity': 91}}
  Query result for Fox: {'data': {'name': 'Fox', 'price': 12.99, 'quantity': 37}}
  placing order for Fox, 10
  Query result for Frisbee: {'data': {'name': 'Frisbee', 'price': 5.99, 'quantity': 63}}
  placing order for Frisbee, 5
  Query result for Barbie: {'data': {'name': 'Barbie', 'price': 55.99, 'quantity': 91}}
  Query result for Frisbee: {'data': {'name': 'Frisbee', 'price': 5.99, 'quantity': 63}}
  Query result for Marbles: {'data': {'name': 'Marbles', 'price': 7.99, 'quantity': 66}}
  placing order for Marbles, 4
  Query result for Barbie: {'data': {'name': 'Barbie', 'price': 55.99, 'quantity': 91}}
  Query result for Fox: {'data': {'name': 'Fox', 'price': 12.99, 'quantity': 37}}
```

Outputs:





```
Probability: 0.0, Request Type: Product Query, Improvement: 12.32%
Probability: 0.2, Request Type: Product Query, Improvement: 15.60%
Probability: 0.4, Request Type: Product Query, Improvement: 12.82%
Probability: 0.8, Request Type: Product Query, Improvement: 4.44%
Probability: 0.0, Request Type: Order Buy, Improvement: 0.00%
Probability: 0.2, Request Type: Order Buy, Improvement: 8.58%
Probability: 0.4, Request Type: Order Buy, Improvement: 4.86%
Probability: 0.8, Request Type: Order Buy, Improvement: 2.61%
Probability: 0.0, Request Type: Order Get, Improvement: 0.00%
Probability: 0.2, Request Type: Order Get, Improvement: 3.91%
Probability: 0.4, Request Type: Order Get, Improvement: 9.39%
Probability: 0.8, Request Type: Order Get, Improvement: 1.45%
```



	A	B	C	D	E	F	G	H	I	J	K	L
1			**WITH CACHING**						**WITHOUT CACHING**			
2		p=0						p=0				
3		client	product query	order buy	order get			client	product query	order buy	order get	
4		5	4.22871	0	0			5	4.82267	0	0	
5												
6			**WITH CACHING**						**WITHOUT CACHING**			
7		p=0.2						p=0.2				
8		client	product query	order buy	order get			client	product query	order buy	order get	
9		5	4.94261	14.52089	10.34376			5	5.85645	15.88343	10.76438	
10												
11			**WITH CACHING**						**WITHOUT CACHING**			
12		p=0.4						p=0.4				
13		client	product query	order buy	order get			client	product query	order buy	order get	
14		5	5.08454	15.14781	10.56752			5	5.83243	15.92214	11.66326	
15												
16			**WITH CACHING**						**WITHOUT CACHING**			
17		p=0.8						p=0.8				
18		client	product query	order buy	order get			client	product query	order buy	order get	
19		5	6.36845	16.24354	11.78032			5	6.66454	16.67835	11.95421	
20												
21												

Additionally, out of curiosity, we also evaluated the latencies when we vary the clients from 1-5 for different requests with and without caching. The results are as follows:

WITH CACHING

p=0

client	product query	order buy	order get
1	3.13569	0	0
2	3.14023	0	0
3	3.56713	0	0
4	4.06112	0	0
5	4.22871	0	0

WITHOUT CACHING

p=0

client	product query	order buy	order get
1	3.34228	0	0
2	3.39913	0	0
3	3.73211	0	0
4	4.13211	0	0
5	4.82267	0	0

WITH CACHING

p=0.2

client	product query	order buy	order get
1	3.13764	12.18887	8.11528
2	3.31179	12.64219	8.90023
3	3.78611	13.00988	9.43299
4	4.22965	13.82332	9.84374
5	4.94261	14.52089	10.34376

WITHOUT CACHING

p=0.2

client	product query	order buy	order get
1	3.15008	12.36871	9.00235
2	3.51129	13.17863	9.46234
3	4.06699	13.95622	10.34732
4	4.71143	14.67823	10.96432
5	5.85645	15.88343	12.76438

WITH CACHING

p=0.4

client	product query	order buy	order get
1	3.37658	12.34523	8.14795
2	3.84372	12.69634	8.90567
3	4.26743	13.44658	9.68347
4	4.77895	14.68523	10.16871
5	5.08454	15.14781	10.56752

WITHOUT CACHING

p=0.4

client	product query	order buy	order get
1	3.88621	12.78943	9.00142
2	3.99901	13.42431	9.73241
3	4.69743	13.99843	10.49523
4	5.24397	15.24314	10.74512
5	5.83243	15.92214	11.66326

WITH CACHING

p=0.8

client	product query	order buy	order get
1	3.94159	12.88343	8.28134
2	4.39763	13.45476	9.03264
3	5.16435	14.17854	9.76734
4	5.86524	14.95424	10.85324
5	6.36845	16.24354	11.78032

WITHOUT CACHING

p=0.8

client	product query	order buy	order get
1	4.14638	13.89432	8.68432
2	4.64323	14.04358	8.87583
3	5.24353	14.68932	9.22456
4	6.09343	15.57835	9.46343
5	6.66454	16.67835	9.95421