Performance Statistics and Experiments

1. We compute the average response time for queries and buys for 1000 sequential requests at the different points in the system. We compute them with and without the cache. We find that the cache speeds the average query response time. However, a cache hit for as particular query followed by cache miss for the same query sees a much bigger difference than the avg response time. This is likely due to a lot of cache missing when multiple clients make buys and invalidate the cache to preserve consistency.

1 Client with Cache:

Function	Client	Frontend	Catalog	Order
Search()	67.2ms	15.9ms	2.9 ms	
Lookup()	90.1ms	38.8 ms	1.8 ms	
Buy()	396.5ms	346 ms	2.5 ms	278.2ms

1 Client without Cache:

Function	Client	Frontend	Catalog	Order
Search()	110.2ms	55.3ms	2.6ms	
Lookup()	100.8ms	52.1ms	1.96ms	
Buy()	373.9ms	321ms	2.61ms	255.7ms

As we can see above using the cache does not really affect the latency of query requests within the catalog server. Also, it does not speed up buy requests as there is no lookup initiated by the frontend in the process of a buy. Therefore, it does not benefit from cache hits. The frontend and end-to-end client queries do see significant speed up because of cache hits however.

Here are the stats for multiple clients:

5 Clients with Cache:

Performance Statistics and Experiments

Function	Client	Frontend	Catalog	Order
Search()	92.9ms	22.4ms	3.0ms	
Lookup()	103.3ms	44.6ms	2.19ms	
Buy()	400.8ms	341ms	2.9ms	271.3ms

5 Clients without Cache:

Function	Client	Frontend	Catalog	Order
Search()	103.4ms	59.9ms	3.6ms	
Lookup()	120.3ms	60.9ms	2.2ms	
Buy()	404.1ms	325ms	2.6ms	260.4ms

10 Clients with Cache:

Function	Client	Frontend	Catalog	Order
Search()	70.7ms	14.5ms	4.2ms	
Lookup()	99.6ms	46.1ms	2.29ms	
Buy()	463.3ms	374ms	3.1ms	301.2ms

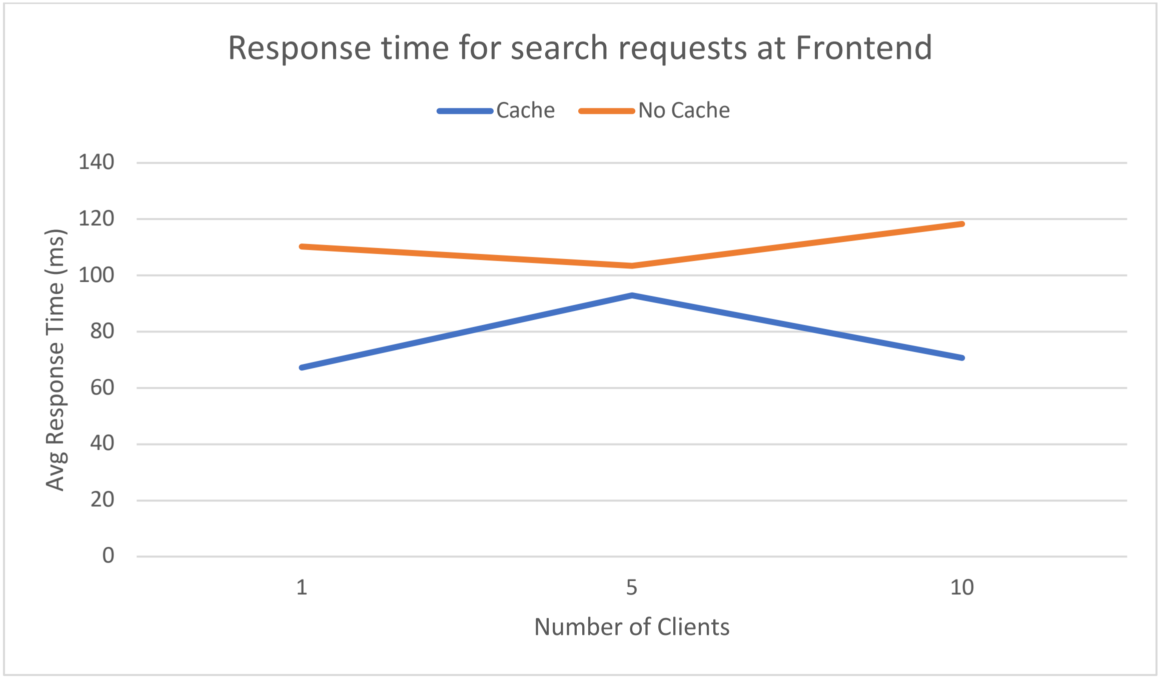
10 Clients without Cache:

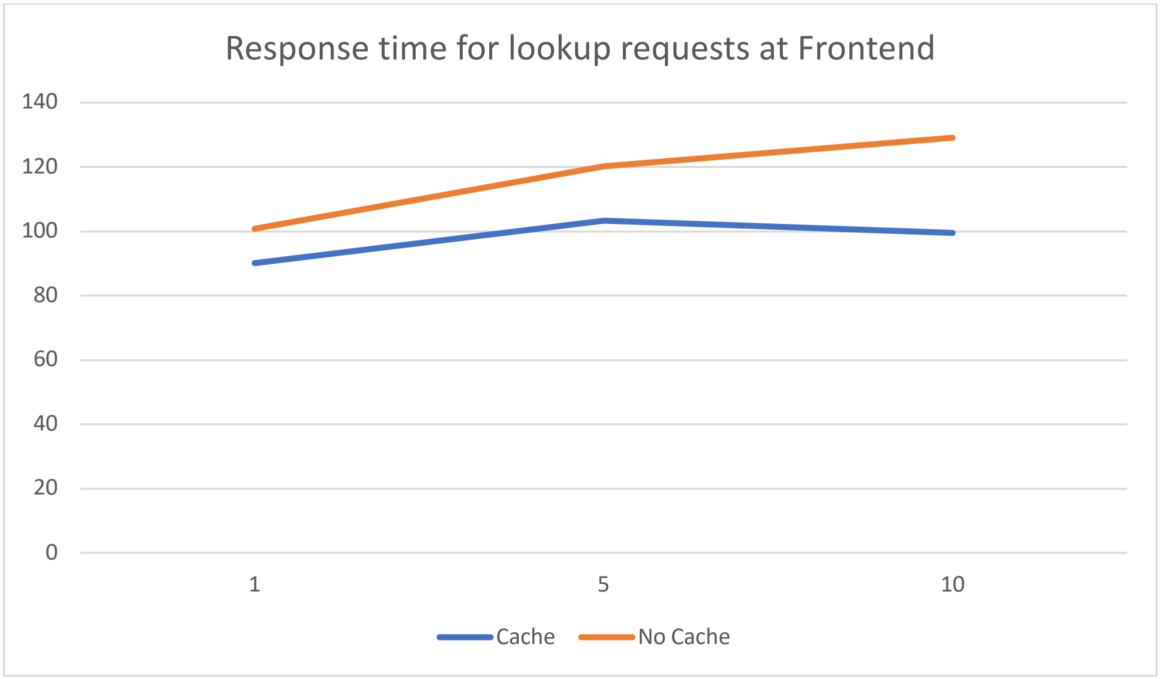
Function	Client	Frontend	Catalog	Order
Search()	118.3ms	66.2ms	4.4ms	
Lookup()	129.1ms	67.4ms	2.5ms	
Buy()	437ms	360ms	3.2ms	288.3ms

As can be seen from the tables above, queries do not slow down much with an increasing number of clients. This is probably because of multiple catalog servers and load balancing providing high availability. Once again, caches do not help with buy requests and queries within the catalog server, but speed up queries for the frontend and end-to-end client requests. Despite this, the difference in latency is

Performance Statistics and Experiments

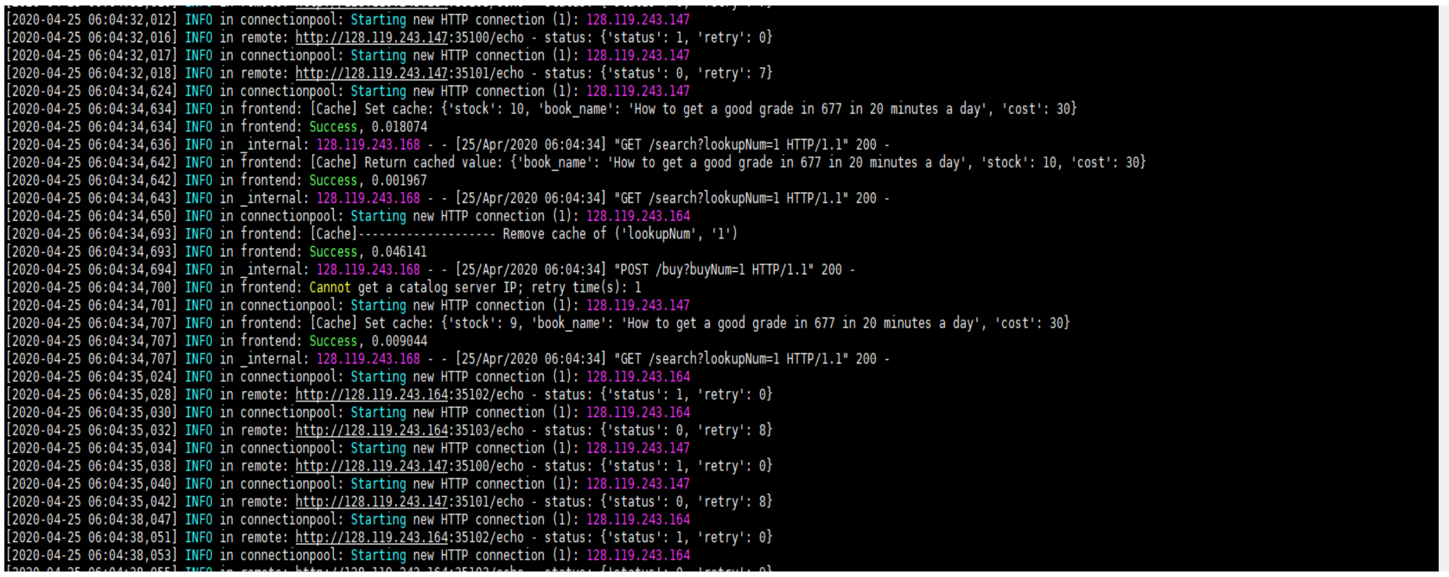
not as high as it can be because there are likely many caches misses as multiple clients could be making successful buys that invalidate cache entries. As in lab2, the latency of buy request go up as the number of clients increase as clients the system ensures consensus before proceeding with the next buy.





As the plots above show, the cache helps in reducing response time for queries at the frontend. The difference in the average response times for queries with a cache and without a cache are not as high as the difference for a particular query before and after it has been invalidated (as we will see below in 2.) as there might be many cache misses when the system is allowed to run.

2. We now compute the difference in response time for a query when it sees a cache hit followed immediately by a cache miss. Our experiment make a query request to put the item into the cache. We then query it again to see a cache hit. Then we buy a unit of the item invalidating the cache, and then query it to see a cache miss. The outputs on edlab are shown below. As we can see, a cache hit followed by an immediate cache miss on the same item results in a slow down of one order of magnitude. The cache hit sees a response time of 1.9 ms and the subsequent cache miss sees a response time of 9.04 ms.

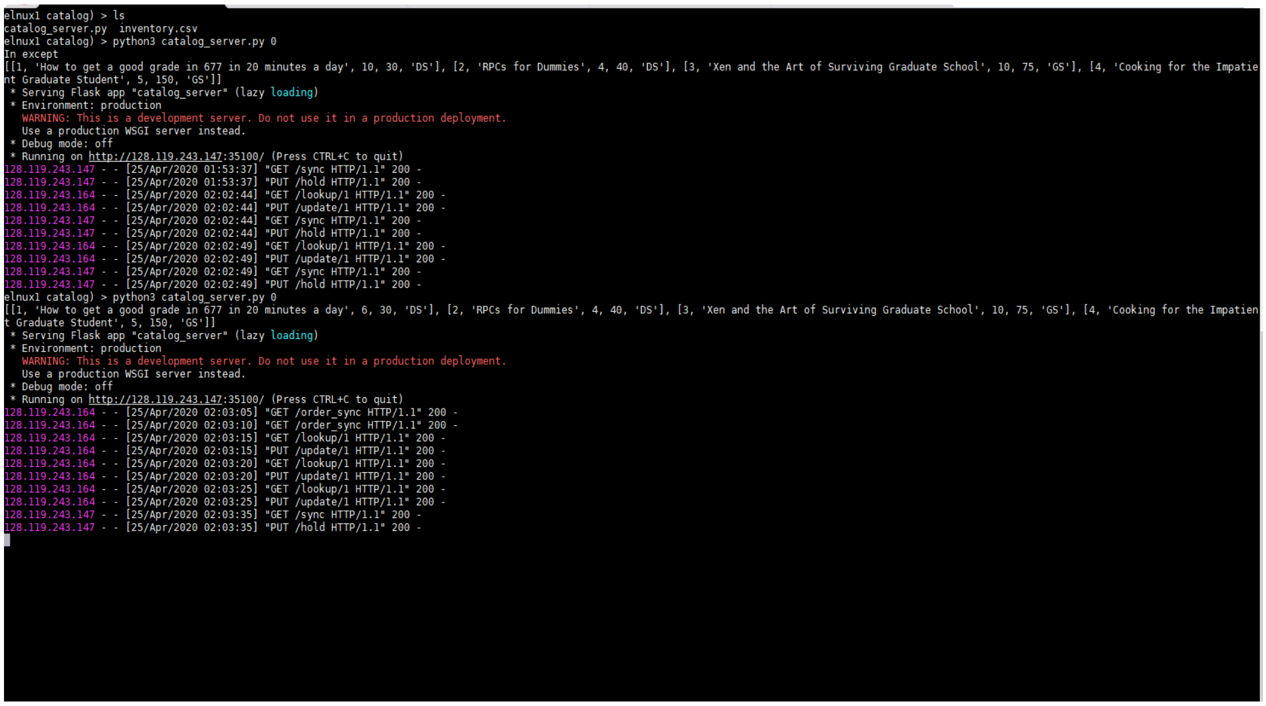


```
[2020-04-25 06:04:32,012] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.147
[2020-04-25 06:04:32,016] INFO in remote: http://128.119.243.147:35100/echo - status: {'status': 1, 'retry': 0}
[2020-04-25 06:04:32,017] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.147
[2020-04-25 06:04:32,018] INFO in remote: http://128.119.243.147:35101/echo - status: {'status': 0, 'retry': 7}
[2020-04-25 06:04:34,624] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.147
[2020-04-25 06:04:34,634] INFO in frontend: [Cache] Set cache: {'stock': 10, 'book_name': 'How to get a good grade in 677 in 20 minutes a day', 'cost': 30}
[2020-04-25 06:04:34,634] INFO in frontend: Success, 0.018074
[2020-04-25 06:04:34,636] INFO in _internal: 128.119.243.168 - - [25/Apr/2020 06:04:34] "GET /search?lookupNum=1 HTTP/1.1" 200 -
[2020-04-25 06:04:34,642] INFO in frontend: [Cache] Return cached value: {'book_name': 'How to get a good grade in 677 in 20 minutes a day', 'stock': 10, 'cost': 30}
[2020-04-25 06:04:34,642] INFO in frontend: Success, 0.001967
[2020-04-25 06:04:34,643] INFO in _internal: 128.119.243.168 - - [25/Apr/2020 06:04:34] "GET /search?lookupNum=1 HTTP/1.1" 200 -
[2020-04-25 06:04:34,650] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.164
[2020-04-25 06:04:34,693] INFO in frontend: [Cache]------------------- Remove cache of ('lookupNum', '1')
[2020-04-25 06:04:34,693] INFO in frontend: Success, 0.046141
[2020-04-25 06:04:34,694] INFO in _internal: 128.119.243.168 - - [25/Apr/2020 06:04:34] "POST /buy?buyNum=1 HTTP/1.1" 200 -
[2020-04-25 06:04:34,700] INFO in frontend: Cannot get a catalog server IP; retry time(s): 1
[2020-04-25 06:04:34,701] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.147
[2020-04-25 06:04:34,707] INFO in frontend: [Cache] Set cache: {'stock': 9, 'book_name': 'How to get a good grade in 677 in 20 minutes a day', 'cost': 30}
[2020-04-25 06:04:34,707] INFO in frontend: Success, 0.009044
[2020-04-25 06:04:34,707] INFO in _internal: 128.119.243.168 - - [25/Apr/2020 06:04:34] "GET /search?lookupNum=1 HTTP/1.1" 200 -
[2020-04-25 06:04:35,024] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.164
[2020-04-25 06:04:35,028] INFO in remote: http://128.119.243.164:35102/echo - status: {'status': 1, 'retry': 0}
[2020-04-25 06:04:35,030] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.164
[2020-04-25 06:04:35,032] INFO in remote: http://128.119.243.164:35103/echo - status: {'status': 0, 'retry': 8}
[2020-04-25 06:04:35,034] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.147
[2020-04-25 06:04:35,038] INFO in remote: http://128.119.243.147:35100/echo - status: {'status': 1, 'retry': 0}
[2020-04-25 06:04:35,040] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.147
[2020-04-25 06:04:35,042] INFO in remote: http://128.119.243.147:35101/echo - status: {'status': 0, 'retry': 8}
[2020-04-25 06:04:38,047] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.164
[2020-04-25 06:04:38,051] INFO in remote: http://128.119.243.164:35102/echo - status: {'status': 1, 'retry': 0}
[2020-04-25 06:04:38,053] INFO in connectionpool: Starting new HTTP connection (1): 128.119.243.164
```

3. The last experiment is to show that the system continues to run when a backend server crashes. We also show synchronization and consensus when a catalog server restarts from a crash. The test client makes 9 buys of a book number 1. A crash is induced in catalog server 1 in the middle of the buys. As the outputs below show, catalog 2 continues to process the buy requests as if there was no fault. Then catalog server 1 is restarted and catalog server 2 is killed. After the 9 buys, we restart catalog server 2. At this point, if the

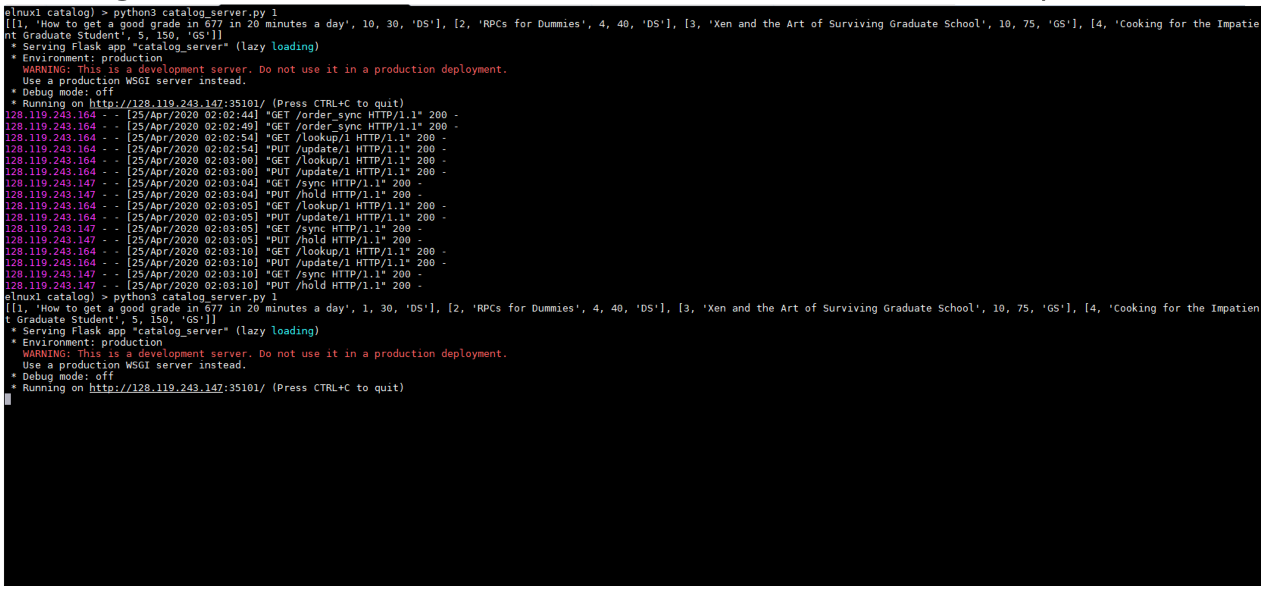
servers had been in sync during the time they were both alive, and if catalog server 2 correctly performs its function while catalog server 1 was down and vice versa, catalog server 2 should have a stock of 1 of the book on restarting and syncing back up with catalog server 1. For this simple experiment we only send buy requests direct to order server 1.

Catalog Server 1: Note it is killed and restarted after 4 successful buys



```
elnux1 catalog) > ls
catalog_server.py  inventory.csv
elnux1 catalog) > python3 catalog_server.py 0
In except
[[1, 'How to get a good grade in 677 in 20 minutes a day', 10, 30, 'DS'], [2, 'RPCs for Dummies', 4, 40, 'DS'], [3, 'Xen and the Art of Surviving Graduate School', 10, 75, 'GS'], [4, 'Cooking for the Impatie
nt Graduate Student', 5, 150, 'GS']]
 * Serving Flask app "catalog_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://128.119.243.147:35100/ (Press CTRL+C to quit)
128.119.243.147 - - [25/Apr/2020 01:53:37] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 01:53:37] "PUT /hold HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:44] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:44] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:02:44] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:02:44] "PUT /hold HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:49] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:49] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:02:49] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:02:49] "PUT /hold HTTP/1.1" 200 -
elnux1 catalog) > python3 catalog_server.py 0
[[1, 'How to get a good grade in 677 in 20 minutes a day', 6, 30, 'DS'], [2, 'RPCs for Dummies', 4, 40, 'DS'], [3, 'Xen and the Art of Surviving Graduate School', 10, 75, 'GS'], [4, 'Cooking for the Impatien
t Graduate Student', 5, 150, 'GS']]
 * Serving Flask app "catalog_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://128.119.243.147:35100/ (Press CTRL+C to quit)
128.119.243.164 - - [25/Apr/2020 02:03:05] "GET /order_sync HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:10] "GET /order_sync HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:15] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:15] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:20] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:20] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:25] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:25] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:35] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:35] "PUT /hold HTTP/1.1" 200 -
```

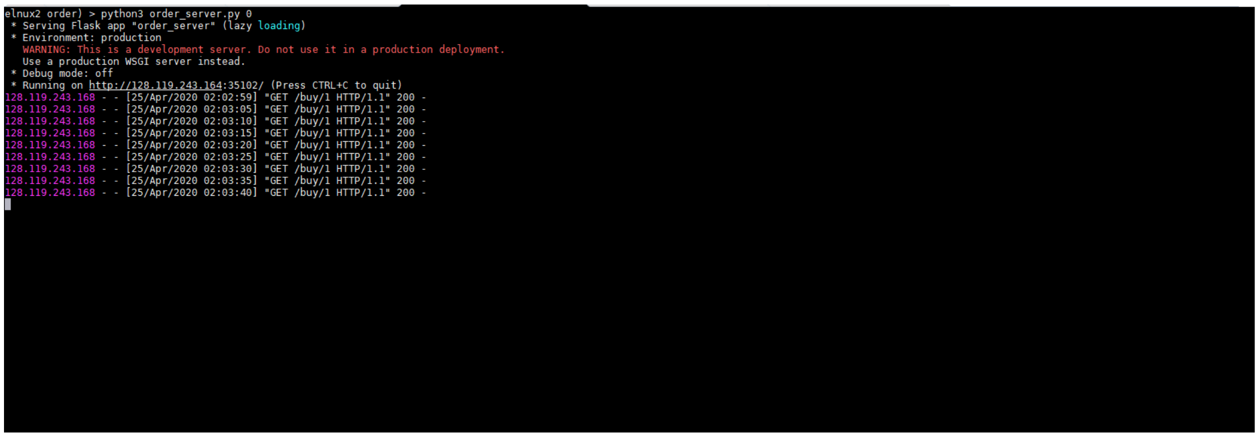
Catalog Server 2: Note how it is killed restarted after all the processes.



```
elnux1 catalog) > python3 catalog_server.py 1
[[1, 'How to get a good grade in 677 in 20 minutes a day', 10, 30, 'DS'], [2, 'RPCs for Dummies', 4, 40, 'DS'], [3, 'Xen and the Art of Surviving Graduate School', 10, 75, 'GS'], [4, 'Cooking for the Impatie
nt Graduate Student', 5, 150, 'GS']]
 * Serving Flask app "catalog_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://128.119.243.147:35101/ (Press CTRL+C to quit)
128.119.243.164 - - [25/Apr/2020 02:02:44] "GET /order_sync HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:49] "GET /order_sync HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:54] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:02:54] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:00] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:00] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:04] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:04] "PUT /hold HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:05] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:05] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:05] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:05] "PUT /hold HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:10] "GET /lookup/1 HTTP/1.1" 200 -
128.119.243.164 - - [25/Apr/2020 02:03:10] "PUT /update/1 HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:10] "GET /sync HTTP/1.1" 200 -
128.119.243.147 - - [25/Apr/2020 02:03:10] "PUT /hold HTTP/1.1" 200 -
elnux1 catalog) > python3 catalog_server.py 1
[[1, 'How to get a good grade in 677 in 20 minutes a day', 1, 30, 'DS'], [2, 'RPCs for Dummies', 4, 40, 'DS'], [3, 'Xen and the Art of Surviving Graduate School', 10, 75, 'GS'], [4, 'Cooking for the Impatien
t Graduate Student', 5, 150, 'GS']]
 * Serving Flask app "catalog_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://128.119.243.147:35101/ (Press CTRL+C to quit)
```

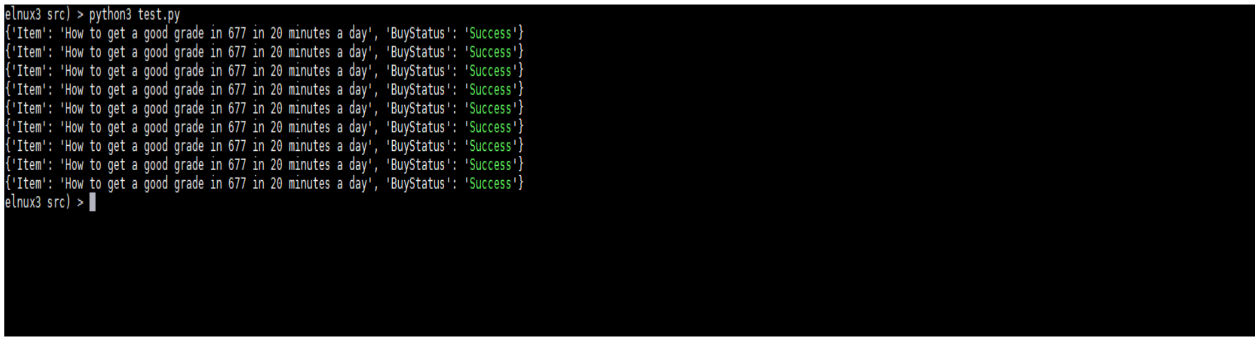
Performance Statistics and Experiments

Order Server 1: Order Server showing 9 GET requests. If you add the number of UPDATE requests in both catalog servers, you would see a total of 9.



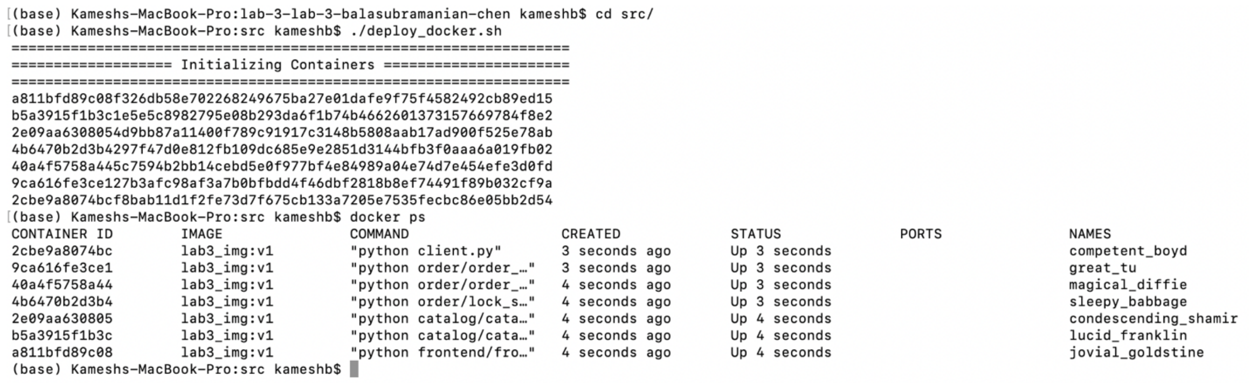
```
elnux2 order) > python3 order_server.py 0
 * Serving Flask app "order_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://128.119.243.164:35102/ (Press CTRL+C to quit)
128.119.243.168 - - [25/Apr/2020 02:02:59] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:05] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:10] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:15] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:20] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:25] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:30] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:35] "GET /buy/1 HTTP/1.1" 200 -
128.119.243.168 - - [25/Apr/2020 02:03:40] "GET /buy/1 HTTP/1.1" 200 -
```

Test Client: Shows 9 successful buys despite crashing and restarting both servers.



```
elnux3 src) > python3 test.py
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
{'Item': 'How to get a good grade in 677 in 20 minutes a day', 'BuyStatus': 'Success'}
elnux3 src) >
```

4. A dockerized version of our system also runs on our local systems.



```
[(base) Kameshs-MacBook-Pro:lab-3-lab-3-balasubramanian-chen kameshb$ cd src/
[(base) Kameshs-MacBook-Pro:src kameshb$ ./deploy_docker.sh
 ==================================================================
 =================== Initializing Containers ======================
 ==================================================================
 a811bfd89c08f326db58e702268249675ba27e01dafe9f75f4582492cb89ed15
 b5a3915f1b3c1e5e5c8982795e08b293da6f1b74b4662601373157669784f8e2
 2e09aa6308054d9bb87a11400f789c91917c3148b5808aab17ad900f525e78ab
 4b6470b2d3b4297f47d0e812fb109dc685e9e2851d3144bfb3f0aaa6a019fb02
 40a4f5758a445c7594b2bb14cebd5e0f977bf4e84989a04e74d7e454efe3d0fd
 9ca616fe3ce127b3afc98af3a7b0bfbdd4f46dbf2818b8ef74491f89b032cf9a
 2cbe9a8074bcf8bab11d1f2fe73d7f675cb133a7205e7535fecbc86e05bb2d54
[(base) Kameshs-MacBook-Pro:src kameshb$ docker ps
 CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
 2cbe9a8074bc        lab3_img:v1         "python client.py"       3 seconds ago       Up 3 seconds                            competent_boyd
 9ca616fe3ce1        lab3_img:v1         "python order/order_…"   3 seconds ago       Up 3 seconds                            great_tu
 40a4f5758a44        lab3_img:v1         "python order/order_…"   4 seconds ago       Up 3 seconds                            magical_diffie
 4b6470b2d3b4        lab3_img:v1         "python order/lock_s…"   4 seconds ago       Up 3 seconds                            sleepy_babbage
 2e09aa630805        lab3_img:v1         "python catalog/cata…"   4 seconds ago       Up 4 seconds                            condescending_shamir
 b5a3915f1b3c        lab3_img:v1         "python catalog/cata…"   4 seconds ago       Up 4 seconds                            lucid_franklin
 a811bfd89c08        lab3_img:v1         "python frontend/fro…"   4 seconds ago       Up 4 seconds                            jovial_goldstine
 (base) Kameshs-MacBook-Pro:src kameshb$
```