CS677 Lab 3

Bharath Narasimhan, Ronak Zala

Turning the Pygmy into an Amazon: Replication, Caching and Consistency

April 18, 2019

1 Readme

1.1 Environment Setup

There is one config file *sv_info.py* that has server information in a dictionary format *Type of Server, IP Address, Port*. Modify the config files as required to setup the docker environment.
There are four docker files *dockerfile.catalog*, *dockerfile.order*, *dockerfile.frontend* and *dockerfile.client* - they represent the catalog server, order server, frontend server and the client respectively.

1. Build the docker images for each of the components using **docker build -f dockerfile.server_name -t server_name**, eg. *docker build -f dockerfile.catalog -t catalog .* Do this for catalog, frontend, order and client.

2. Create a subnet using **docker network create - -subnet=172.18.0.0/16 mynet123**

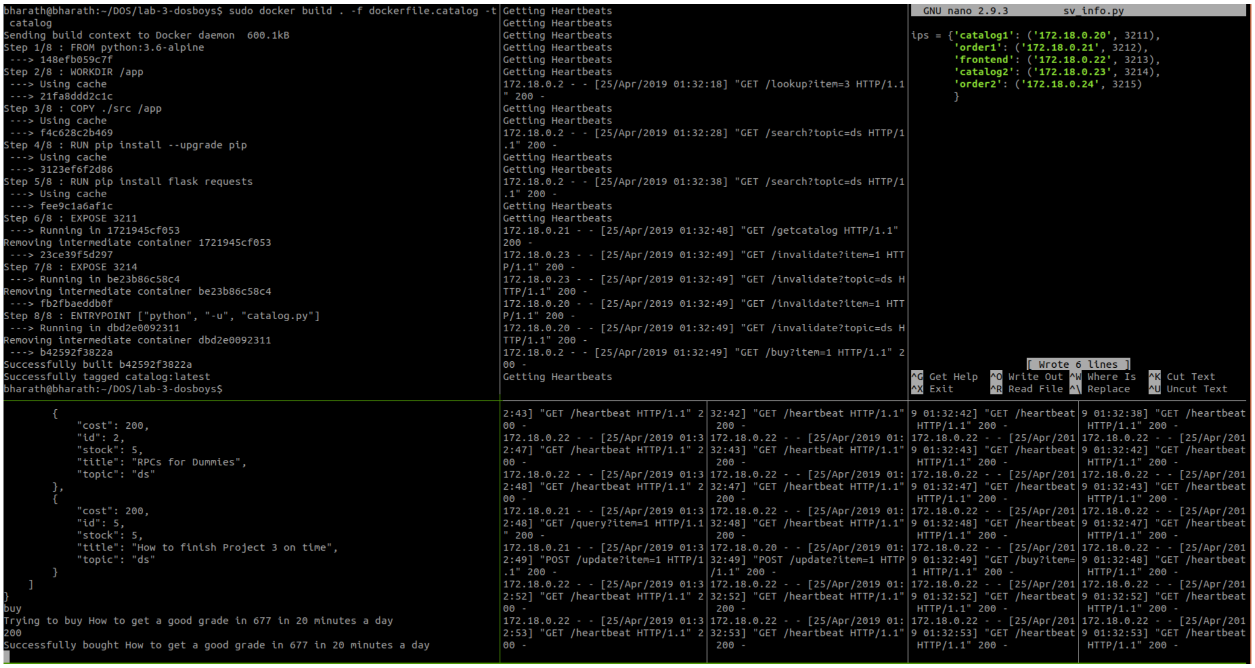
3. Run the docker containers using **docker run - -net mynet123 - -ip ip_here server_name server_id first_start**. ip_here refers to the IP in sv_info.py . server_id is for the case of replicating servers (catalog or order) and is zero-indexed. first_start is for the case of catalog server, where it is initialized with a default catalog if first_start=1, or resynced if first_start=0, Ex - *docker run - -net mynet123 - -ip 172.18.0.20 catalog 0 1* ***(for first start)*** and *docker run - -net mynet123 - -ip 172.18.0.20 catalog 0 0* ***(for subsequent starts)***

4. The default IPs are 172.18.0.x where $20 \leq x \leq 25$ and the default ports are y where $3211 \leq y \leq 3215$

5. We chose this way of configuration as setting up static ips using a created subnet involves minimal conifguration once the docker images have been built. There is no need to do multiple *docker ps-a*, *docker inspect container_name* and *docker cp source destination* with this method.

6. If you encounter a *maximum number of running instances reached* error, try increasing the *max_instances* parameter of *scheduler.add_job* function in *frontend.py*

Dockerization was done using a python3.6-Alpine base image, similar to the Alpine Linux base image which occupies minimal space (5MB). The dockerfiles copy the contents of the *src* directory, install the necessary libraries - flask, requests and apscheduler - and expose the appropriate ports. The output after dockerization (running 6 containers - 2 catalog, 2 order, 1 frontend, 1 client) is shown in Figure 1. We only show the results for containers on the same machine as discussed on Piazza.



```
bharath@bharath:~/DOS/lab-3-dosboys$ sudo docker build . -f dockerfile.catalog -t
 catalog
Sending build context to Docker daemon  600.1kB
Step 1/8 : FROM python:3.6-alpine
 ---> 148efb059c7f
Step 2/8 : WORKDIR /app
 ---> Using cache
 ---> 21fa8ddd2c1c
Step 3/8 : COPY ./src /app
 ---> Using cache
 ---> f4c628c2b469
Step 4/8 : RUN pip install --upgrade pip
 ---> Using cache
 ---> 3123ef6f2d86
Step 5/8 : RUN pip install flask requests
 ---> Using cache
 ---> fee9c1a6af1c
Step 6/8 : EXPOSE 3211
 ---> Running in 1721945cf053
Removing intermediate container 1721945cf053
 ---> 23ce39f5d297
Step 7/8 : EXPOSE 3214
 ---> Running in be23b86c58c4
Removing intermediate container be23b86c58c4
 ---> fb2fbaeddb0f
Step 8/8 : ENTRYPOINT ["python", "-u", "catalog.py"]
 ---> Running in dbd2e0092311
Removing intermediate container dbd2e0092311
 ---> b42592f3822a
Successfully built b42592f3822a
Successfully tagged catalog:latest
bharath@bharath:~/DOS/lab-3-dosboys$
```

```
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
172.18.0.2 - - [25/Apr/2019 01:32:18] "GET /lookup?item=3 HTTP/1.1
" 200 -
Getting Heartbeats
Getting Heartbeats
172.18.0.2 - - [25/Apr/2019 01:32:28] "GET /search?topic=ds HTTP/1
.1" 200 -
Getting Heartbeats
Getting Heartbeats
172.18.0.2 - - [25/Apr/2019 01:32:38] "GET /search?topic=ds HTTP/1
.1" 200 -
Getting Heartbeats
Getting Heartbeats
172.18.0.21 - - [25/Apr/2019 01:32:48] "GET /getcatalog HTTP/1.1"
200 -
172.18.0.23 - - [25/Apr/2019 01:32:49] "GET /invalidate?item=1 HTT
P/1.1" 200 -
172.18.0.23 - - [25/Apr/2019 01:32:49] "GET /invalidate?topic=ds H
TTP/1.1" 200 -
172.18.0.20 - - [25/Apr/2019 01:32:49] "GET /invalidate?item=1 HTT
P/1.1" 200 -
172.18.0.20 - - [25/Apr/2019 01:32:49] "GET /invalidate?topic=ds H
TTP/1.1" 200 -
172.18.0.2 - - [25/Apr/2019 01:32:49] "GET /buy?item=1 HTTP/1.1" 2
00 -
Getting Heartbeats
```

```
  GNU nano 2.9.3          sv_info.py

ips = {'catalog1': ('172.18.0.20', 3211),
       'order1': ('172.18.0.21', 3212),
       'frontend': ('172.18.0.22', 3213),
       'catalog2': ('172.18.0.23', 3214),
       'order2': ('172.18.0.24', 3215)
       }

                    [ Wrote 6 lines ]
^G Get Help   ^O Write Out ^W Where Is   ^K Cut Text
^X Exit       ^R Read File ^\ Replace    ^U Uncut Text
```

```
        {
            "cost": 200,
            "id": 2,
            "stock": 5,
            "title": "RPCs for Dummies",
            "topic": "ds"
        },
        {
            "cost": 200,
            "id": 5,
            "stock": 5,
            "title": "How to finish Project 3 on time",
            "topic": "ds"
        }
    ]
}
buy
Trying to buy How to get a good grade in 677 in 20 minutes a day
200
Successfully bought How to get a good grade in 677 in 20 minutes a day
```

```
2:43] "GET /heartbeat HTTP/1.1" 2
00 -
172.18.0.22 - - [25/Apr/2019 01:3
2:47] "GET /heartbeat HTTP/1.1" 2
00 -
172.18.0.22 - - [25/Apr/2019 01:3
2:48] "GET /heartbeat HTTP/1.1" 2
00 -
172.18.0.21 - - [25/Apr/2019 01:3
2:48] "GET /query?item=1 HTTP/1.1
" 200 -
172.18.0.21 - - [25/Apr/2019 01:3
2:49] "POST /update?item=1 HTTP/1
.1" 200 -
172.18.0.22 - - [25/Apr/2019 01:3
2:52] "GET /heartbeat HTTP/1.1" 2
00 -
172.18.0.22 - - [25/Apr/2019 01:3
2:53] "GET /heartbeat HTTP/1.1" 2
00 -
```

```
32:42] "GET /heartbeat HTTP/1.1"
 200 -
172.18.0.22 - - [25/Apr/2019 01:
32:43] "GET /heartbeat HTTP/1.1"
 200 -
172.18.0.22 - - [25/Apr/2019 01:
32:47] "GET /heartbeat HTTP/1.1"
 200 -
172.18.0.22 - - [25/Apr/2019 01:
32:48] "GET /heartbeat HTTP/1.1"
 200 -
172.18.0.20 - - [25/Apr/2019 01:
32:49] "POST /update?item=1 HTTP
/1.1" 200 -
172.18.0.22 - - [25/Apr/2019 01:
32:52] "GET /heartbeat HTTP/1.1"
 200 -
172.18.0.22 - - [25/Apr/2019 01:
32:53] "GET /heartbeat HTTP/1.1"
 200 -
```

```
9 01:32:42] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:43] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:47] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:48] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:49] "GET /buy?item=
1 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:52] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:53] "GET /heartbeat
 HTTP/1.1" 200 -
```

```
9 01:32:38] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:42] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:43] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:47] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:48] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:52] "GET /heartbeat
 HTTP/1.1" 200 -
172.18.0.22 - - [25/Apr/201
9 01:32:53] "GET /heartbeat
 HTTP/1.1" 200 -
```

Figure 1: Dockerization of web applications

2 Program Design

Framework used - Flask[1] is a micro web framework written in Python. The Flask version used is 1.0.2
Additional Libraries used - requests (2.9.1), apscheduler (3.6.0)

2.1 File Outline

Note - Items in bold refer to additions after Lab 2

2.1.1 Catalog Server

The file catalog.py represents the implementation of the catalog server in question. Book details are stored in a persistent manner in *catalog.json*. It has the following methods:

1. *get_books* (Query) [GET]: Method to return the results of a client-side *search* or *lookup*. If the query parameter passed is *topic* followed by *gs*(Graduate School) or

[1] http://flask.pocoo.org/

Item ID	Book Name
1	How to get a good grade in 677 in 20 minutes a day
2	RPCs for Dummies
3	Xen and the Art of Surviving Graduate School
4	Cooking for the Impatient Graduate Student
5	How to finish Project 3 on time
6	Why theory classes are so hard
7	Spring in the Pioneer Valley

Table 1: Book names and their corresponding IDs

ds(Distributed Systems), it returns all entries belonging to the topic category specified. If the query parameter passed is *item* followed by the item number (Refer Table 2.1.1, it returns details such as number of items in stock, cost.

2. *update_books* (Update) [POST]: Method to update the stock or cost of specified item. It takes in a query parameter *item* and increments the *stock* of the corresponding book by *delta*. It can also update the *cost* of the item specified. *cost* and *delta* are passed as a JSON with the POST request. Note that negative delta corresponds to a decrement in stock (happens with each buy request).
   There is also copy of the POST call forwarded to the other catalog replica to ensure write consistency. An infinite loop of POST calls is prevented by passing a bool 'order' in the JSON of the POST call, which is an indicator of the origin of the call (Order server or not).
   Every update invalidates the cache for the respective book and its corresponding topic in the frontend server using a GET call.

3. *heartbeat* [GET]: **Method periodically polled by the frontend server. It returns a positive response if the catalog server is up and running.**

4. *resync* [GET]: **A REST endpoint that returns the catalog of the current replica. It is called by other catalog servers during resynchronization after a crash failure.**

5. *main*: **The catalog server is started with 2 arguments - server ID (0/1) and first start (0/1). If first start is 1, a default catalog is loaded, else resynchronization is attempted by calling the *resync* endpoint of its replica. The zero-indexed server ID is used in numerous places to ensure less hardcoding and easy extensibility to ≥ 2 replicas.**

2.1.2 Order Server

The file order.py represents the implementation of the order server in question. Transaction details are stored in a persistent manner in *order_log.txt*. It has the following methods:

1. *buy_order* [GET]: Method to return the results of a client-side *buy* request. If the query parameter passed is *item* followed by the item number (Refer Table 2.1.1, it **queries**

the catalog server to to check if the item being requested is in stock. If yes, it **updates** the stock of the given item by a *delta* of -1. If not, it does nothing and prints an 'Out of Stock' message.
The order server calls the *get_catalog* endpoint of the frontend server which returns the id of the next available catalog server in a round-robin fashion. This makes sure that cases like Frontend - Order1 - Catalog2 and Frontend - Order2 - Catalog1 are handled well. (There is no need to assume a one-to-one mapping of order to catalog with this design.)

2. *heartbeat* [GET]: **Method periodically polled by the frontend server. It returns a positive response if the order server is up and running.**

2.1.3 Frontend Server

The file frontend.py represents the implementation of the frontend server in question. It functions as an abstraction layer between the client and the servers. It has the following methods:

1. *search* [GET]: Method to return the results of a client-side *search* request. The frontend server just forwards the search request as a **query** to the catalog server. **Calls to *get_catalog_server_id* decide the load balancing.**

2. *lookup* [GET]: Method to return the results of a client-side *lookup* request. The frontend server just forwards the lookup request as a **query** to the catalog server. **Calls to *get_catalog_server_id* decide the load balancing.**

3. *buy* [GET]: Method to return the results of a client-side *buy* request. The frontend server just forwards the buy request to the order server. **Calls to *get_order_server_id* decide the load balancing.**

4. *format_now*: A simple datetime formatter

5. *invalidate* [GET]: **Method to invalidate the cache for a particular item and its corresponding topic. Called from order server after every update operation.**

6. *get_crashed* [GET]: **Returns True if the catalog server with given ID has crashed in the past. Used in resynchronization.**

7. *get_catalog_server_id* [GET]: **The *next* operator is used to iterate over catalog servers, where their current states are observed as the result of the latest *heartbeat*. If the current catalog server is down, the function calls itself recursively, effectively iterating to the next replica using the *next* operator. In this way, this function eventually returns the id of the *next available catalog server*.**

8. *get_order_server_id* [GET]: **Similar functionality as the above method, applied to order server replicas.**

Note that this design evaluates to *round-robin per-request* if all servers are up

9. *heartbeat* [GET] : **Gets 4 states (Catalog1, Catalog2, Order1, Order2) - stored as global variables - every 5 seconds. Also sets *crashed*(in the past) variable for catalog servers.**

2.1.4 Client

The file client.py represents the implementation of the client in question. It has the following methods:

1. *main*: The functionality of the code is tested by randomly calling one of *search*, *lookup* or *buy* every few seconds (default - 5) with a request to the frontend server. Stock is updated by calling *update_stock* every few seconds.

2. *test_response_times*: A utility function to perform *num_req* number of sequential client requests and measure the average response time. Call with *num_req* and *mode* (*search, lookup or buy)* to get per-tier response times written to *times* directory.

3. *update_stock*: A function to periodically increment the stock of a random book by 2 (default). This is done by directly making a POST call to the catalog server. **This method has been commented out at times to ensure relevant debug statements are visible.**

4. *pp_json*: A utility function to pretty-print a given JSON file.

2.1.5 Time Parser

The file time_parser.py is used to get the average response times by taking the mean of the times written to the files *(server)_(method)_time.txt* in *times* directory. Run *python3 time_parser.py* after running *test_response_times* with *mode = 'search', 'lookup', 'buy'* in *client.py* to see the ARTs (in seconds). All files in the *times* directory must be deleted before running *test_response_times* to ensure proper ARTs are being recorded.

2.2 Design Features

2.2.1 Replication and Caching

Replication - The frontend server uses a *round-robin, per-request* load balancer to handle requests to the replicas. For the sake of demonstration, the number of replicas of the order and catalog servers are 2 each, but this can be easily extended to many more. We use a partial state (as all round-robin load balancers do) using global state variables o_{state} and c_{state}.

Caching - When a new query request comes in, the front end server checks the cache first before it forwards the request to the catalog server. Note that caching is only useful for read requests (queries to catalog); write requests, which are basically orders or update requests, to the catalog must be processed by the order or catalog servers rather than the cache. This is an *in-memory cache* integrated into the front-end server process - internal function calls are used to get and put items into the cache.

Cache consistency is also implemented wherein backend replicas send invalidate requests to the in-memory cache prior to making any writes to their database files. The invalidate request causes the data for that item to be removed from the cache.

Write Synchronization - The replicas also use an internal protocol to ensure that any writes to their database are also performed at the other replica to keep them in sync with one another.

2.2.2 Dockerization

As discussed above, dockerization has several advantages:

1. You can deploy different components on different machines.
2. Easy scalability according to load

2.2.3 Fault Tolerance

Discussed in 4.4 below.

2.2.4 Other design features

1. Minimal configuration (Only docker server host and port) required.
2. Concurrency built-in in Flask.
3. Edge cases like wrong product/missing product handled well
4. Clear print messages for readability at client. Debug messages at all 3 servers
5. End-to-end testing as well as Unit testing done.

3 Github

The source code can be found at https://github.com/umass-cs677-spring19/lab-3-dosboys. It is divided into 3 folders *docs*, *src* for documentation and code respectively.

4 Evaluation and Measurement

4.1 Evaluation on EdLab machines

The catalog, order and frontend servers were setup on *elnux1, elnux3*, *elnux3, elnux7* and *elnux7* respectively. (These are the default values and can be changed in *sv_info.csv*). The client was started from *elnux2*.

4.2 Average Response time

To properly compare the average response times with and without caching, we conduct the experiment in a controlled setting instead of a random simulation. We analyze the effect of caching on each method separately. This could give information about the upper bound of performance increase from caching. *update_stock()* was suppressed whenever necessary to ensure required print statements were visible.

4.2.1 Average Response time per client request without caching

The results of running 1000 sequential requests to search(), lookup() and buy() are shown in Table 4.2.1 and visualized in Figure 2(a). We see that the *search* and *lookup* calls take the same amount of time as expected because they are similar GET requests. *buy* takes more time than the two as the order server has to first query the catalog server with a GET request, and then update the catalog server with a POST request. We observe that time spent in the catalog tier is the least because it is just calculations with no network involved. The time spent increases from Frontend to Client largely due to network latency. The difference between two successive bars would give us twice the latency between the two machines.

Method\Response time (ms)	Catalog tier	Order tier	Frontend tier	Client tier
search()	0.974	-	8.715	15.91
lookup()	0.929	-	8.633	15.78
buy()	15.623	32.284	41.205	48.42

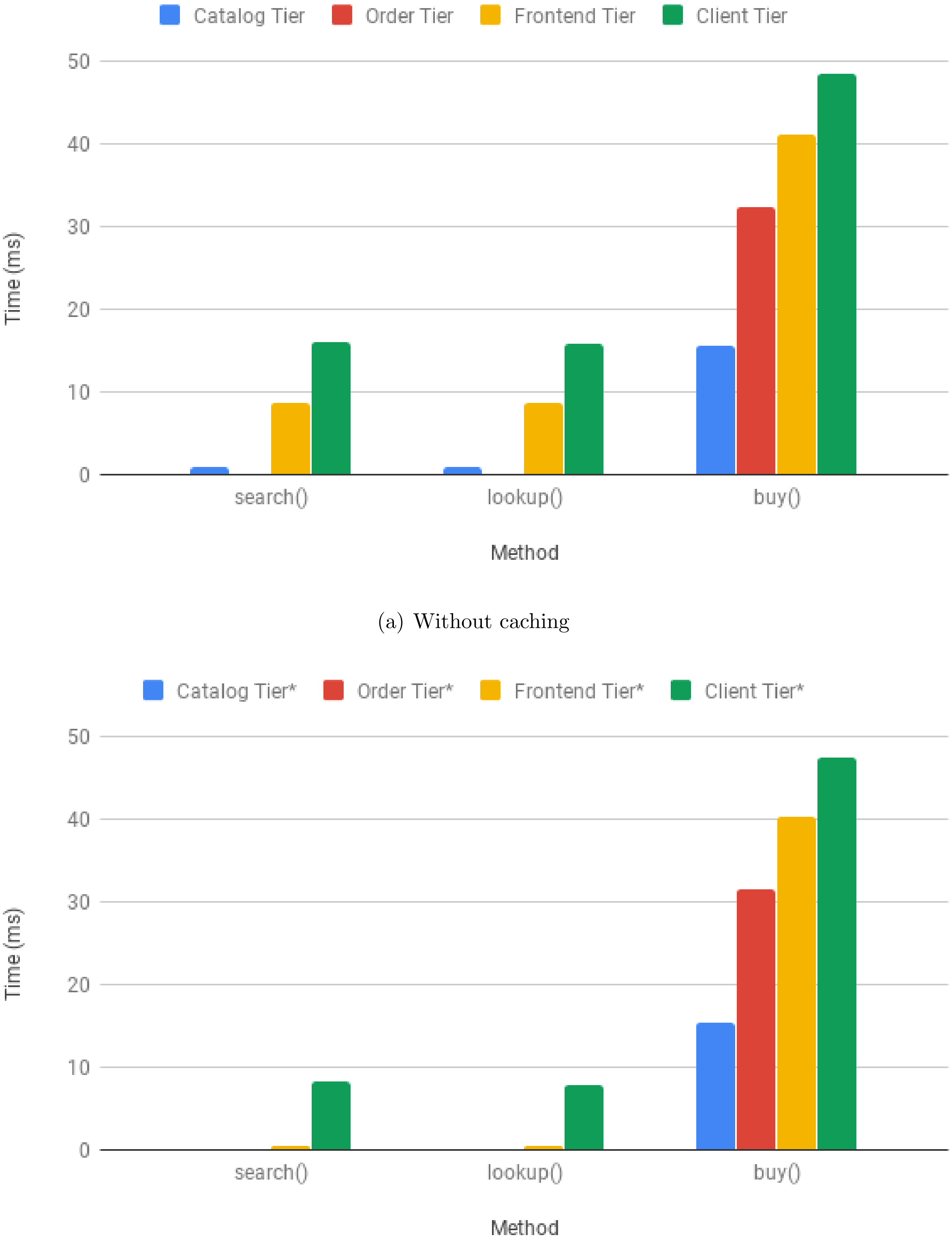
Table 2: Per-tier response time for query and buy requests (averaged for 1000 sequential requests each) in milliseconds **without caching**

4.2.2 Average Response time per client request with caching

The results of running 1000 sequential requests to search(), lookup() and buy() are shown in Table 4.2.2 and visualized in Figure 2(b). We observe that caching dramatically improves the ART for *search()* and *lookup()*. Decreases of **48.44**%, **49.96**% were observed for *search()* and *lookup()* respectively. The ARTs for the *buy()* were almost the same as before (**0.02**% difference) as there is no caching involved in this case.

Method\Response time (ms)	Catalog tier*	Order tier*	Frontend tier*	Client tier*
search()	0.012	-	0.531	8.202
lookup()	0.005	-	0.6	7.896
buy()	15.338	31.505	40.23	47.408

Table 3: Per-tier response time for query and buy requests (averaged for 1000 sequential requests each) in milliseconds **with caching**



(a) Without caching

(b) With caching

Figure 2: Per-tier response time for query and buy requests (averaged for 1000 sequential requests each) in milliseconds with and without caching

4.3 Cache Invalidation

As stated in the design features, caches are invalidated with every update call by calling an *invalidate* REST endpoint at the frontend server. To calculate the overhead of ensuring cache consistency, a controlled experiment was carried out with only *buy()* calls. 1000 sequential buy() calls were carried out with and without cache consistency and the results are shown in Table 4.3.

We see that there is an overhead of $4 - 18\%$ across all tiers, with the most effect being in the catalog tier where the *invalidate* call originates from. Comparing this to the 50% decreases observed for *search()* and *lookup()* with the use of caching in the previous section, we can say that this is a reasonable tradeoff to make.

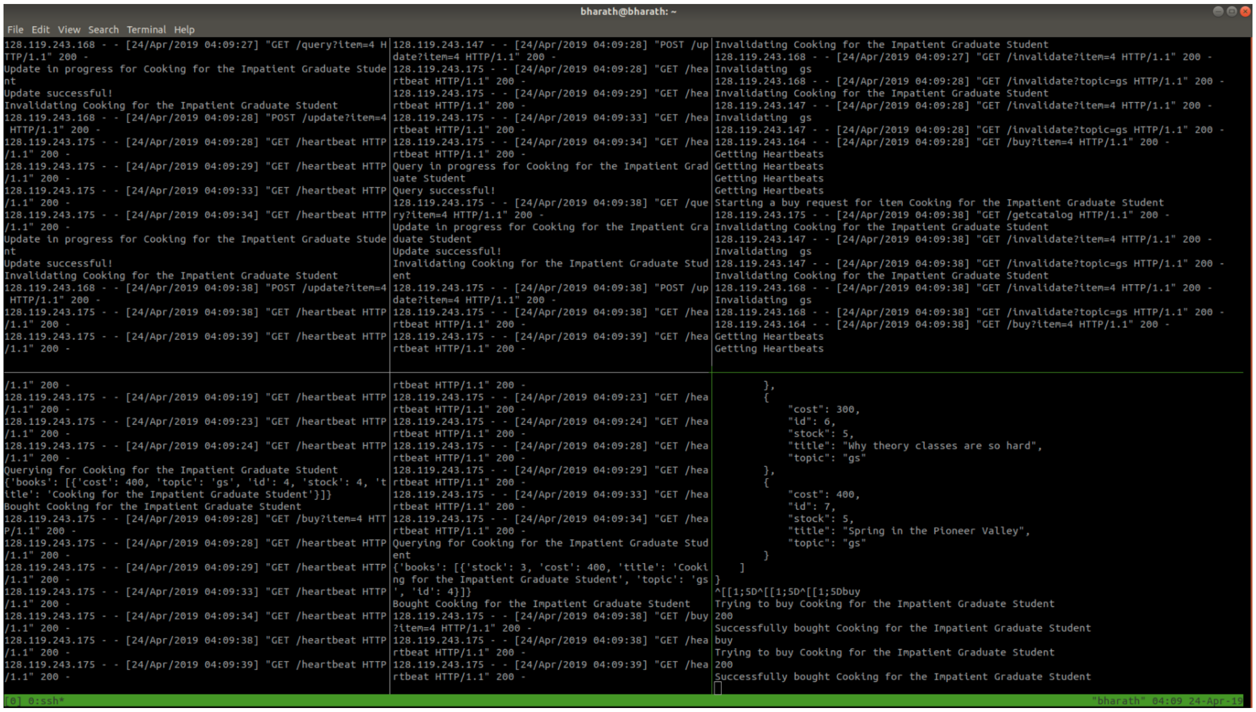
The latency of a request after invalidation (**cache miss**) is **15.84 ms** if it is a *search()* call, **15.87 ms** if it is a *lookup()* call, and **48.2 ms** if it is a *buy()* call. This was evaluated using a controlled experiment wherein a *buy()* was called first, followed by a query for the same item, or an update for any item (No caching involved in updates).

Method\Response time (ms)	Catalog tier	Order tier	Frontend tier	Client tier
Without invalidation	9.749	26.131	35.014	42.236
With invalidation	11.55	27.891	36.735	43.99
Overhead	**18.473 %**	**6.735 %**	**4.915 %**	**4.152 %**

Table 4: Overheads for cache invalidation

4.3.1 Write Replication

POST calls are made from one catalog replica to the other during an update to ensure write consistency. To prevent an infinite back-and-forth loop of this update process, a JSON is passed with the key *'order'* set to 1 if the request needs to be further propogated to the catalog replica. If the key *'order'* is set to 0, no further propogation occurs. Figure 3 shows the stdout of the 6 servers (Catalog1, Catalog2, Frontend, Order1, Order2, Client in order, left to right, top to bottom). The top-left 2 panes show the 2 catalog servers in *elnux1* and *elnux3*. We observe from stdout that the update calls are being synchronized, thus ensuring consistency. The line 'Query successful' in the second pane (absent in the first pane) shows that the query was originally directed to Catalog2, and then replicated on Catalog1.



```
128.119.243.168 - - [24/Apr/2019 04:09:27] "GET /query?item=4 H
TTP/1.1" 200 -
Update in progress for Cooking for the Impatient Graduate Stude
nt
Update successful!
Invalidating Cooking for the Impatient Graduate Student
128.119.243.168 - - [24/Apr/2019 04:09:28] "POST /update?item=4
 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:28] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:29] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:33] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:34] "GET /heartbeat HTTP
/1.1" 200 -
Update in progress for Cooking for the Impatient Graduate Stude
nt
Update successful!
Invalidating Cooking for the Impatient Graduate Student
128.119.243.168 - - [24/Apr/2019 04:09:38] "POST /update?item=4
 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:39] "GET /heartbeat HTTP
/1.1" 200 -
```

```
128.119.243.147 - - [24/Apr/2019 04:09:28] "POST /up
date?item=4 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:28] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:29] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:33] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:34] "GET /hea
rtbeat HTTP/1.1" 200 -
Query in progress for Cooking for the Impatient Grad
uate Student
Query successful!
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /que
ry?item=4 HTTP/1.1" 200 -
Update in progress for Cooking for the Impatient Gra
duate Student
Update successful!
Invalidating Cooking for the Impatient Graduate Stud
ent
128.119.243.175 - - [24/Apr/2019 04:09:38] "POST /up
date?item=4 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:39] "GET /hea
rtbeat HTTP/1.1" 200 -
```

```
Invalidating Cooking for the Impatient Graduate Student
128.119.243.168 - - [24/Apr/2019 04:09:27] "GET /invalidate?item=4 HTTP/1.1" 200 -
Invalidating  gs
128.119.243.168 - - [24/Apr/2019 04:09:28] "GET /invalidate?topic=gs HTTP/1.1" 200 -
Invalidating Cooking for the Impatient Graduate Student
128.119.243.147 - - [24/Apr/2019 04:09:28] "GET /invalidate?item=4 HTTP/1.1" 200 -
Invalidating  gs
128.119.243.147 - - [24/Apr/2019 04:09:28] "GET /invalidate?topic=gs HTTP/1.1" 200 -
128.119.243.164 - - [24/Apr/2019 04:09:28] "GET /buy?item=4 HTTP/1.1" 200 -
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
Starting a buy request for item Cooking for the Impatient Graduate Student
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /getcatalog HTTP/1.1" 200 -
Invalidating Cooking for the Impatient Graduate Student
128.119.243.147 - - [24/Apr/2019 04:09:38] "GET /invalidate?item=4 HTTP/1.1" 200 -
Invalidating  gs
128.119.243.147 - - [24/Apr/2019 04:09:38] "GET /invalidate?topic=gs HTTP/1.1" 200 -
Invalidating Cooking for the Impatient Graduate Student
128.119.243.168 - - [24/Apr/2019 04:09:38] "GET /invalidate?item=4 HTTP/1.1" 200 -
Invalidating  gs
128.119.243.168 - - [24/Apr/2019 04:09:38] "GET /invalidate?topic=gs HTTP/1.1" 200 -
128.119.243.164 - - [24/Apr/2019 04:09:38] "GET /buy?item=4 HTTP/1.1" 200 -
Getting Heartbeats
Getting Heartbeats
```

```
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:19] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:23] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:24] "GET /heartbeat HTTP
/1.1" 200 -
Querying for Cooking for the Impatient Graduate Student
{'books': [{'cost': 400, 'topic': 'gs', 'id': 4, 'stock': 4, 't
itle': 'Cooking for the Impatient Graduate Student'}]}
Bought Cooking for the Impatient Graduate Student
128.119.243.175 - - [24/Apr/2019 04:09:28] "GET /buy?item=4 HTT
P/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:28] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:29] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:33] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:34] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /heartbeat HTTP
/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:39] "GET /heartbeat HTTP
/1.1" 200 -
```

```
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:23] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:24] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:28] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:29] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:33] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:34] "GET /hea
rtbeat HTTP/1.1" 200 -
Querying for Cooking for the Impatient Graduate Stud
ent
{'books': [{'stock': 3, 'cost': 400, 'title': 'Cooki
ng for the Impatient Graduate Student', 'topic': 'gs
', 'id': 4}]}
Bought Cooking for the Impatient Graduate Student
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /buy
?item=4 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:38] "GET /hea
rtbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:09:39] "GET /hea
rtbeat HTTP/1.1" 200 -
```

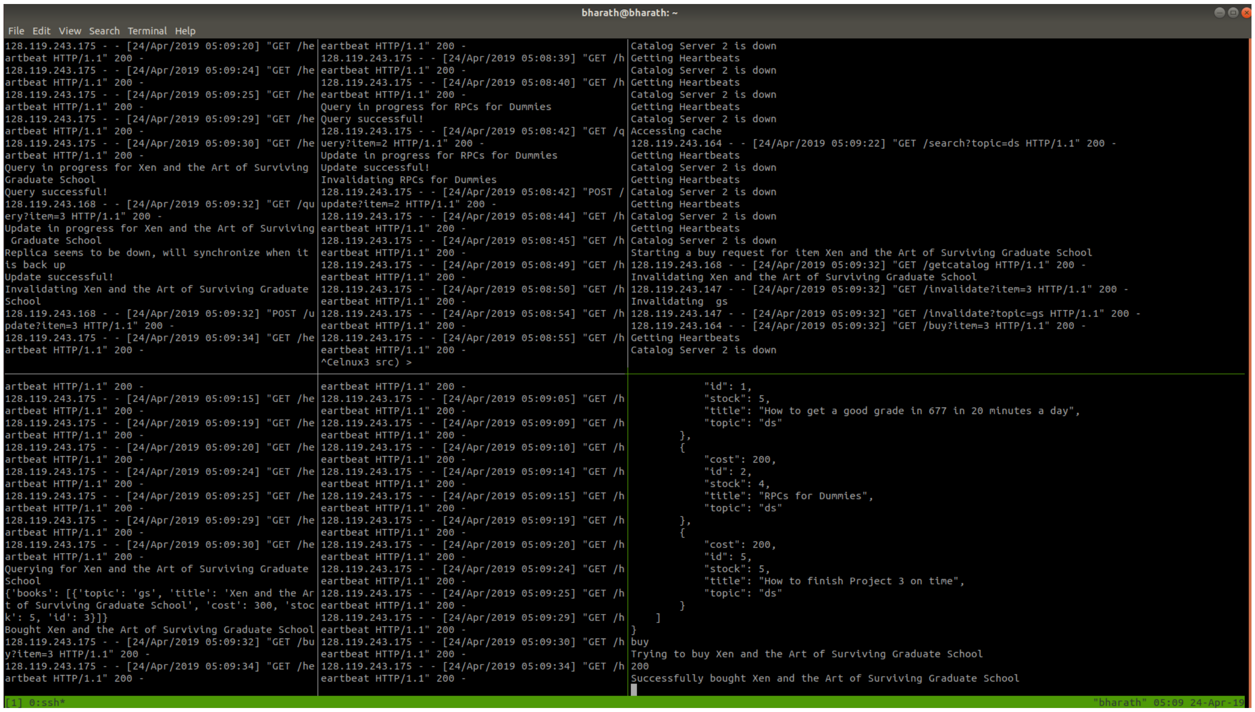
```
        },
        {
            "cost": 300,
            "id": 6,
            "stock": 5,
            "title": "Why theory classes are so hard",
            "topic": "gs"
        },
        {
            "cost": 400,
            "id": 7,
            "stock": 5,
            "title": "Spring in the Pioneer Valley",
            "topic": "gs"
        }
    ]
}
^[[1;5D^[[1;5D^[[1;5Dbuy
Trying to buy Cooking for the Impatient Graduate Student
200
Successfully bought Cooking for the Impatient Graduate Student
buy
Trying to buy Cooking for the Impatient Graduate Student
200
Successfully bought Cooking for the Impatient Graduate Student
```

Figure 3: Screenshot showing write consistency (first two panes)

4.4 Fault Tolerance

All servers were started as described earlier and crash faults simulated as shown in Figure 4. The frontend server is aware of the state of all catalog and order servers through the periodic hearbeat. The states which are acquired using simple GET requests are enclosed in try-except blocks to prevent any untoward behaviour. We use a round-robin, per-request design, keeping in mind all active servers, and iterating through them using the next() operator. The iterator used is *cycle* from *itertools* library.

For the resync case, we need to only look at the starting of a catalog server after it has gone down (handling an order server resurrection is trivial). This is shown in Figure 5. We have a REST endpoint that checks if the catalog server has crashed in the past when it is starting up. Depending on if it has or not, it is resynced with its replica or initialized with a default catalog. Note - The case of both catalogs down is not in the scope of this assignment, although relevant debug messages will be printed by the Frontend server.


bharath@bharath: ~
File Edit View Search Terminal Help
[1] 0:ssh*
"bharath" 05:09 24-Apr-19

```
128.119.243.175 - - [24/Apr/2019 05:09:20] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:24] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:25] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:29] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:30] "GET /he
artbeat HTTP/1.1" 200 -
Query in progress for Xen and the Art of Surviving
Graduate School
Query successful!
128.119.243.168 - - [24/Apr/2019 05:09:32] "GET /qu
ery?item=3 HTTP/1.1" 200 -
Update in progress for Xen and the Art of Surviving
 Graduate School
Replica seems to be down, will synchronize when it
is back up
Update successful!
Invalidating Xen and the Art of Surviving Graduate
School
128.119.243.168 - - [24/Apr/2019 05:09:32] "POST /u
pdate?item=3 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:34] "GET /he
artbeat HTTP/1.1" 200 -
```

```
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:39] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:40] "GET /h
eartbeat HTTP/1.1" 200 -
Query in progress for RPCs for Dummies
Query successful!
128.119.243.175 - - [24/Apr/2019 05:08:42] "GET /q
uery?item=2 HTTP/1.1" 200 -
Update in progress for RPCs for Dummies
Update successful!
Invalidating RPCs for Dummies
128.119.243.175 - - [24/Apr/2019 05:08:42] "POST /
update?item=2 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:44] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:45] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:49] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:50] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:54] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:08:55] "GET /h
eartbeat HTTP/1.1" 200 -
^Celnux3 src) >
```

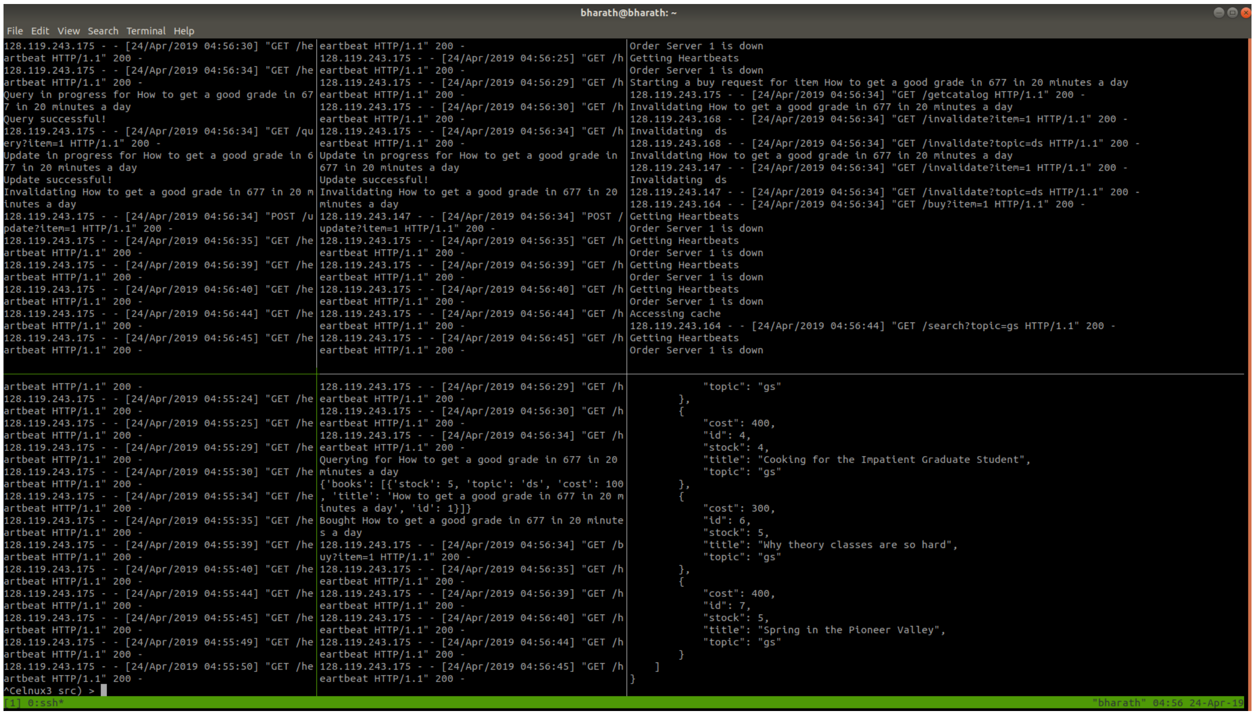
```
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Accessing cache
128.119.243.164 - - [24/Apr/2019 05:09:22] "GET /search?topic=ds HTTP/1.1" 200 -
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Starting a buy request for item Xen and the Art of Surviving Graduate School
128.119.243.168 - - [24/Apr/2019 05:09:32] "GET /getcatalog HTTP/1.1" 200 -
Invalidating Xen and the Art of Surviving Graduate School
128.119.243.147 - - [24/Apr/2019 05:09:32] "GET /invalidate?item=3 HTTP/1.1" 200 -
Invalidating  gs
128.119.243.147 - - [24/Apr/2019 05:09:32] "GET /invalidate?topic=gs HTTP/1.1" 200 -
128.119.243.164 - - [24/Apr/2019 05:09:32] "GET /buy?item=3 HTTP/1.1" 200 -
Getting Heartbeats
Catalog Server 2 is down
```

```
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:15] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:19] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:20] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:24] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:25] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:29] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:30] "GET /he
artbeat HTTP/1.1" 200 -
Querying for Xen and the Art of Surviving Graduate
School
{'books': [{'topic': 'gs', 'title': 'Xen and the Ar
t of Surviving Graduate School', 'cost': 300, 'stoc
k': 5, 'id': 3}]}
Bought Xen and the Art of Surviving Graduate School
128.119.243.175 - - [24/Apr/2019 05:09:32] "GET /bu
y?item=3 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:34] "GET /he
artbeat HTTP/1.1" 200 -
```

```
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:05] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:09] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:10] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:14] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:15] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:19] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:20] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:24] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:25] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:29] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:30] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 05:09:34] "GET /h
eartbeat HTTP/1.1" 200 -
```

```
            "id": 1,
            "stock": 5,
            "title": "How to get a good grade in 677 in 20 minutes a day",
            "topic": "ds"
        },
        {
            "cost": 200,
            "id": 2,
            "stock": 4,
            "title": "RPCs for Dummies",
            "topic": "ds"
        },
        {
            "cost": 200,
            "id": 5,
            "stock": 5,
            "title": "How to finish Project 3 on time",
            "topic": "ds"
        }
    ]
}
buy
Trying to buy Xen and the Art of Surviving Graduate School
200
Successfully bought Xen and the Art of Surviving Graduate School
```

(a) Catalog Server 2 down (Notice update to catalog replica is on hold - top-left pane)


bharath@bharath: ~
File Edit View Search Terminal Help
[1] 0:ssh*
"bharath" 04:56 24-Apr-19

```
128.119.243.175 - - [24/Apr/2019 04:56:30] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:34] "GET /he
artbeat HTTP/1.1" 200 -
Query in progress for How to get a good grade in 67
7 in 20 minutes a day
Query successful!
128.119.243.175 - - [24/Apr/2019 04:56:34] "GET /qu
ery?item=1 HTTP/1.1" 200 -
Update in progress for How to get a good grade in 6
77 in 20 minutes a day
Update successful!
Invalidating How to get a good grade in 677 in 20 m
inutes a day
128.119.243.175 - - [24/Apr/2019 04:56:34] "POST /u
pdate?item=1 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:35] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:39] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:40] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:44] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:45] "GET /he
artbeat HTTP/1.1" 200 -
```

```
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:25] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:29] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:30] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:34] "GET /h
eartbeat HTTP/1.1" 200 -
Update in progress for How to get a good grade in
677 in 20 minutes a day
Update successful!
Invalidating How to get a good grade in 677 in 20
minutes a day
128.119.243.147 - - [24/Apr/2019 04:56:34] "POST /
update?item=1 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:35] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:39] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:40] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:44] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:45] "GET /h
eartbeat HTTP/1.1" 200 -
```

```
Order Server 1 is down
Getting Heartbeats
Order Server 1 is down
Starting a buy request for item How to get a good grade in 677 in 20 minutes a day
128.119.243.175 - - [24/Apr/2019 04:56:34] "GET /getcatalog HTTP/1.1" 200 -
Invalidating How to get a good grade in 677 in 20 minutes a day
128.119.243.168 - - [24/Apr/2019 04:56:34] "GET /invalidate?item=1 HTTP/1.1" 200 -
Invalidating  ds
128.119.243.168 - - [24/Apr/2019 04:56:34] "GET /invalidate?topic=ds HTTP/1.1" 200 -
Invalidating How to get a good grade in 677 in 20 minutes a day
128.119.243.147 - - [24/Apr/2019 04:56:34] "GET /invalidate?item=1 HTTP/1.1" 200 -
Invalidating  ds
128.119.243.147 - - [24/Apr/2019 04:56:34] "GET /invalidate?topic=ds HTTP/1.1" 200 -
128.119.243.164 - - [24/Apr/2019 04:56:34] "GET /buy?item=1 HTTP/1.1" 200 -
Getting Heartbeats
Order Server 1 is down
Getting Heartbeats
Order Server 1 is down
Getting Heartbeats
Order Server 1 is down
Getting Heartbeats
Order Server 1 is down
Accessing cache
128.119.243.164 - - [24/Apr/2019 04:56:44] "GET /search?topic=gs HTTP/1.1" 200 -
Getting Heartbeats
Order Server 1 is down
```

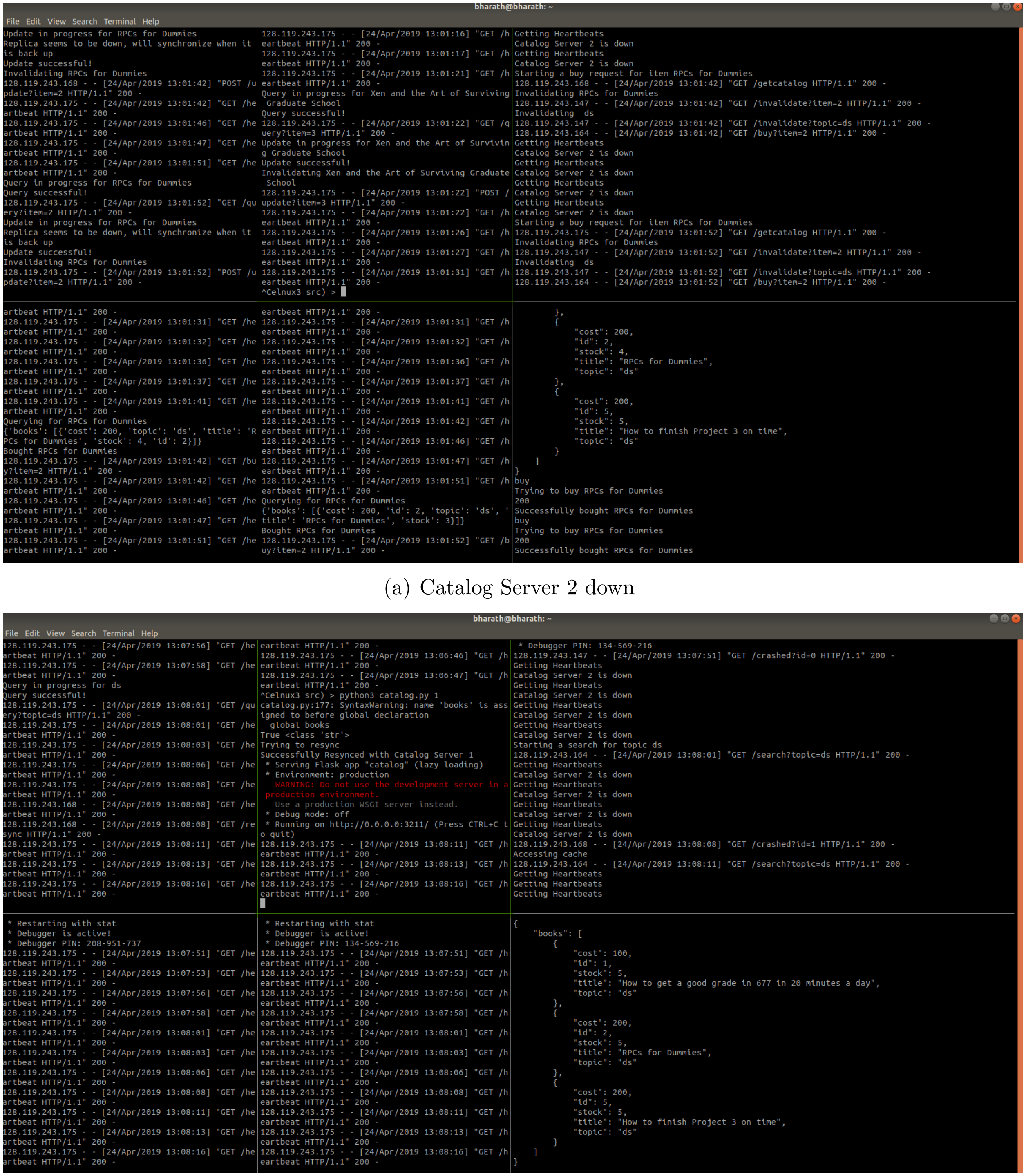
```
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:24] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:25] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:29] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:30] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:34] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:35] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:39] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:40] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:44] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:45] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:49] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:55:50] "GET /he
artbeat HTTP/1.1" 200 -
^Celnux3 src) >
```

```
128.119.243.175 - - [24/Apr/2019 04:56:29] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:30] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:34] "GET /h
eartbeat HTTP/1.1" 200 -
Querying for How to get a good grade in 677 in 20
minutes a day
{'books': [{'stock': 5, 'topic': 'ds', 'cost': 100
, 'title': 'How to get a good grade in 677 in 20 m
inutes a day', 'id': 1}]}
Bought How to get a good grade in 677 in 20 minute
s a day
128.119.243.175 - - [24/Apr/2019 04:56:34] "GET /b
uy?item=1 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:35] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:39] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:40] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:44] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 04:56:45] "GET /h
eartbeat HTTP/1.1" 200 -
```

```
            "topic": "gs"
        },
        {
            "cost": 400,
            "id": 4,
            "stock": 4,
            "title": "Cooking for the Impatient Graduate Student",
            "topic": "gs"
        },
        {
            "cost": 300,
            "id": 6,
            "stock": 5,
            "title": "Why theory classes are so hard",
            "topic": "gs"
        },
        {
            "cost": 400,
            "id": 7,
            "stock": 5,
            "title": "Spring in the Pioneer Valley",
            "topic": "gs"
        }
    ]
}
```

(b) Order Server 1 down

Figure 4: Screenshots showing fault tolerance design



```
Update in progress for RPCs for Dummies
Replica seems to be down, will synchronize when it
is back up
Update successful!
Invalidating RPCs for Dummies
128.119.243.168 - - [24/Apr/2019 13:01:42] "POST /u
pdate?item=2 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:42] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:46] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:47] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:51] "GET /he
artbeat HTTP/1.1" 200 -
Query in progress for RPCs for Dummies
Query successful!
128.119.243.175 - - [24/Apr/2019 13:01:52] "GET /qu
ery?item=2 HTTP/1.1" 200 -
Update in progress for RPCs for Dummies
Replica seems to be down, will synchronize when it
is back up
Update successful!
Invalidating RPCs for Dummies
128.119.243.175 - - [24/Apr/2019 13:01:52] "POST /u
pdate?item=2 HTTP/1.1" 200 -
```

```
128.119.243.175 - - [24/Apr/2019 13:01:16] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:17] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:21] "GET /h
eartbeat HTTP/1.1" 200 -
Query in progress for Xen and the Art of Surviving
 Graduate School
Query successful!
128.119.243.175 - - [24/Apr/2019 13:01:22] "GET /q
uery?item=3 HTTP/1.1" 200 -
Update in progress for Xen and the Art of Survivin
g Graduate School
Update successful!
Invalidating Xen and the Art of Surviving Graduate
 School
128.119.243.175 - - [24/Apr/2019 13:01:22] "POST /
update?item=3 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:22] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:26] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:27] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:31] "GET /h
eartbeat HTTP/1.1" 200 -
^Celnux3 src) >
```

```
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Starting a buy request for item RPCs for Dummies
128.119.243.168 - - [24/Apr/2019 13:01:42] "GET /getcatalog HTTP/1.1" 200 -
Invalidating RPCs for Dummies
128.119.243.147 - - [24/Apr/2019 13:01:42] "GET /invalidate?item=2 HTTP/1.1" 200 -
Invalidating  ds
128.119.243.147 - - [24/Apr/2019 13:01:42] "GET /invalidate?topic=ds HTTP/1.1" 200 -
128.119.243.164 - - [24/Apr/2019 13:01:42] "GET /buy?item=2 HTTP/1.1" 200 -
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Starting a buy request for item RPCs for Dummies
128.119.243.175 - - [24/Apr/2019 13:01:52] "GET /getcatalog HTTP/1.1" 200 -
Invalidating RPCs for Dummies
128.119.243.147 - - [24/Apr/2019 13:01:52] "GET /invalidate?item=2 HTTP/1.1" 200 -
Invalidating  ds
128.119.243.147 - - [24/Apr/2019 13:01:52] "GET /invalidate?topic=ds HTTP/1.1" 200 -
128.119.243.164 - - [24/Apr/2019 13:01:52] "GET /buy?item=2 HTTP/1.1" 200 -
```

```
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:31] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:32] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:36] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:37] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:41] "GET /he
artbeat HTTP/1.1" 200 -
Querying for RPCs for Dummies
{'books': [{'cost': 200, 'topic': 'ds', 'title': 'R
PCs for Dummies', 'stock': 4, 'id': 2}]}
Bought RPCs for Dummies
128.119.243.175 - - [24/Apr/2019 13:01:42] "GET /bu
y?item=2 HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:42] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:46] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:47] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:51] "GET /he
artbeat HTTP/1.1" 200 -
```

```
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:31] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:32] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:36] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:37] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:41] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:42] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:46] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:47] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:01:51] "GET /h
eartbeat HTTP/1.1" 200 -
Querying for RPCs for Dummies
{'books': [{'cost': 200, 'id': 2, 'topic': 'ds', '
title': 'RPCs for Dummies', 'stock': 3}]}
Bought RPCs for Dummies
128.119.243.175 - - [24/Apr/2019 13:01:52] "GET /b
uy?item=2 HTTP/1.1" 200 -
```

```
        },
        {
            "cost": 200,
            "id": 2,
            "stock": 4,
            "title": "RPCs for Dummies",
            "topic": "ds"
        },
        {
            "cost": 200,
            "id": 5,
            "stock": 5,
            "title": "How to finish Project 3 on time",
            "topic": "ds"
        }
    ]
}
buy
Trying to buy RPCs for Dummies
200
Successfully bought RPCs for Dummies
buy
Trying to buy RPCs for Dummies
200
Successfully bought RPCs for Dummies
```

(a) Catalog Server 2 down

```
128.119.243.175 - - [24/Apr/2019 13:07:56] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:58] "GET /he
artbeat HTTP/1.1" 200 -
Query in progress for ds
Query successful!
128.119.243.175 - - [24/Apr/2019 13:08:01] "GET /qu
ery?topic=ds HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:01] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:03] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:06] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:08] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.168 - - [24/Apr/2019 13:08:08] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.168 - - [24/Apr/2019 13:08:08] "GET /re
sync HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:11] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:13] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:16] "GET /he
artbeat HTTP/1.1" 200 -
```

```
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:06:46] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:06:47] "GET /h
eartbeat HTTP/1.1" 200 -
^Celnux3 src) > python3 catalog.py 1
catalog.py:177: SyntaxWarning: name 'books' is ass
igned to before global declaration
  global books
True <class 'str'>
Trying to resync
Successfully Resynced with Catalog Server 1
 * Serving Flask app "catalog" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a
 production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:3211/ (Press CTRL+C t
o quit)
128.119.243.175 - - [24/Apr/2019 13:08:11] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:13] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:16] "GET /h
eartbeat HTTP/1.1" 200 -
```

```
 * Debugger PIN: 134-569-216
128.119.243.147 - - [24/Apr/2019 13:07:51] "GET /crashed?id=0 HTTP/1.1" 200 -
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Starting a search for topic ds
128.119.243.164 - - [24/Apr/2019 13:08:01] "GET /search?topic=ds HTTP/1.1" 200 -
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
Getting Heartbeats
Catalog Server 2 is down
128.119.243.168 - - [24/Apr/2019 13:08:08] "GET /crashed?id=1 HTTP/1.1" 200 -
Accessing cache
128.119.243.164 - - [24/Apr/2019 13:08:11] "GET /search?topic=ds HTTP/1.1" 200 -
Getting Heartbeats
Getting Heartbeats
Getting Heartbeats
```

```
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 208-951-737
128.119.243.175 - - [24/Apr/2019 13:07:51] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:53] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:56] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:58] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:01] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:03] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:06] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:08] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:11] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:13] "GET /he
artbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:16] "GET /he
artbeat HTTP/1.1" 200 -
```

```
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 134-569-216
128.119.243.175 - - [24/Apr/2019 13:07:51] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:53] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:56] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:07:58] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:01] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:03] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:06] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:08] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:11] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:13] "GET /h
eartbeat HTTP/1.1" 200 -
128.119.243.175 - - [24/Apr/2019 13:08:16] "GET /h
eartbeat HTTP/1.1" 200 -
```

```
{
    "books": [
        {
            "cost": 100,
            "id": 1,
            "stock": 5,
            "title": "How to get a good grade in 677 in 20 minutes a day",
            "topic": "ds"
        },
        {
            "cost": 200,
            "id": 2,
            "stock": 5,
            "title": "RPCs for Dummies",
            "topic": "ds"
        },
        {
            "cost": 200,
            "id": 5,
            "stock": 5,
            "title": "How to finish Project 3 on time",
            "topic": "ds"
        }
    ]
}
```

(b) Catalog Server 2 back up and resynced - second pane

Figure 5: Screenshots showing successful resynchronization

4.5 Consensus using RAFT

The extra credit part was done in a different branch **lab-3-dosboys/raft** so as to prevent untoward changes to existing code.

Raft implements consensus by first electing a distinguished leader, then giving the leader complete responsibility for managing the replicated log. The leader accepts log entries from clients, replicates them on other servers(in our case these are the order server replicas), and tells servers when it is safe to commit the log entries.

The order servers have state assigned to them from the following:

1. **Follower** (Default)
   - Here the server starts with a randomized timeout.
   - In the meantime it also responds to the request received from other server replicas.
   - Depending upon which server timeouts first without receiving any other calls from server replica, that server converts to **Candidate**.
2. **Candidate**
   - On conversion to Candidate state, the replica increments the currentTerm, resets the election timer, send request vote to all other servers.
   - If it receives majority votes it changes it's state to leader.
   - But before that if receives any appendEntry call it's state changes back to **Follower**.
   - Finally if the elections timeouts then it will again start a new election
3. **Leader**
   - On being elected as a leader, the server sends periodic requests to the replica servers to prevent the election timeouts.
   - If a request comes from frontend it appends entry to local log and responds after entry is committed.
   - Commiting process requires the leader to send appendEntity requests to the replicas. After passing all the checks either the requests gets committed or just appended to the logs.

   Each time when a leader is elected the **TERM** gets incremented. There are two pointers, one is *commitIndex* which points to highest log entry, the other is *lastApplied* which points to the highest index of the log entry which has been committed(in our case passed on to the catalog server)

   Running of order servers implemented using RAFT protocol is same as the initial parts. We were able to implement almost all of the features of RAFT in the above manner, but did not have the time to run experiments and performance evaluations.