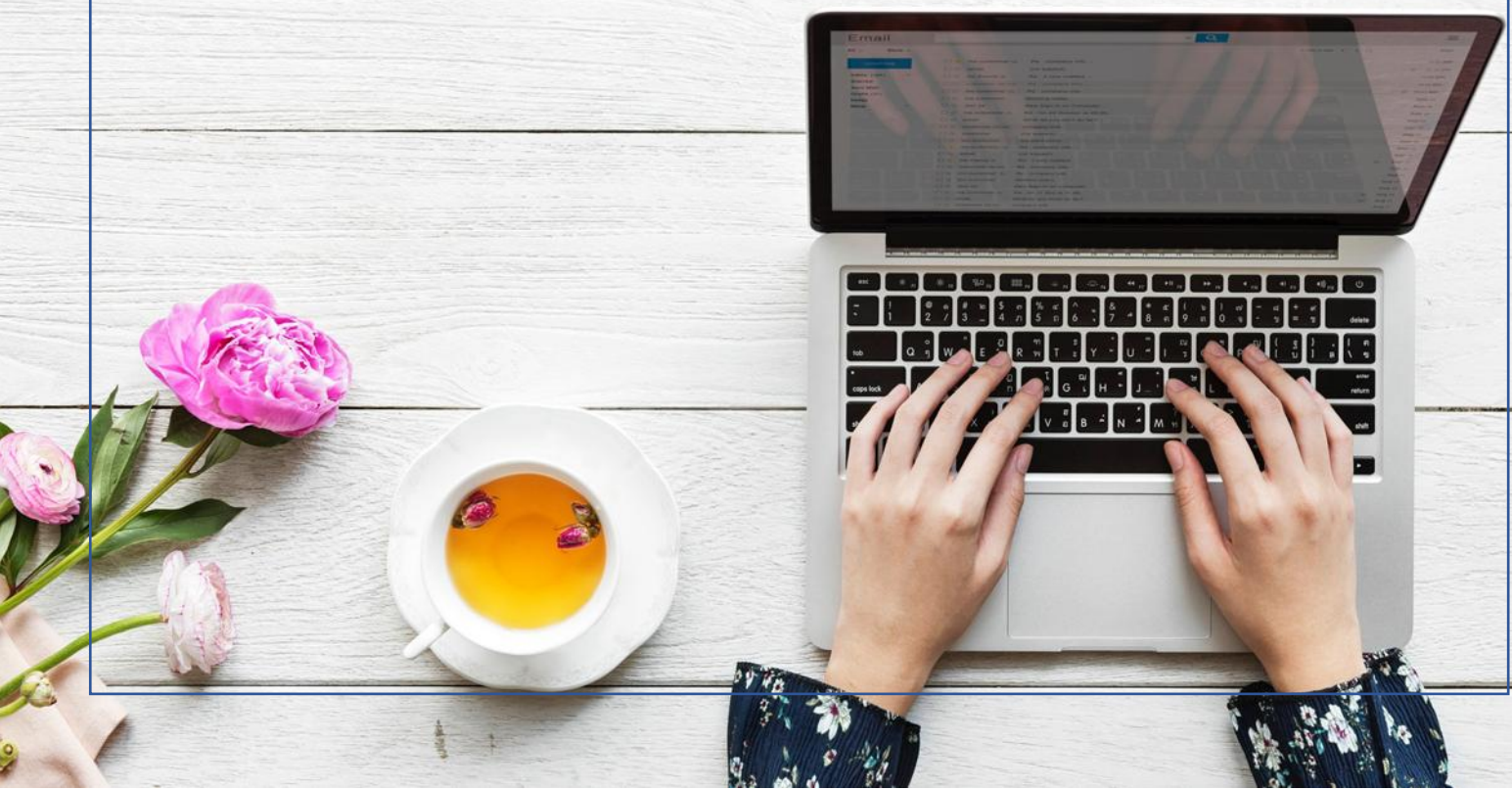
Output generated for Bazar.com project Part 2

Lina Qurom

11924435



- ## Caching & Replication

  Sending a request for specific book information by frontend server 5002:

```
C:\Users\ZBOOK>curl http://localhost:5002/info/1
{
  "price": 25.0,
  "quantity": 13,
  "title": "How to get a good grade in DOS in 40 minutes a day"
}
```

Here in frontend terminal, and we notice that this item does not exist in the cache that has been limited to 100 items, so the result is “Cache Miss”. Thus, this item is the first item to be stored in the cache, we also see the time taken to implement this request, which is **2.02860** seconds**.**

```
 * Running on http://127.0.0.1:5002
 * Running on http://192.168.1.103:5002
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 850-146-242
Book info endpoint. Using catalog server: http://localhost:5003
Cache Miss! Item Number: 1, Cache Capacity: 1/100, Time Taken: 2.02860 seconds
127.0.0.1 - - [12/Jan/2024 23:30:37] "GET /info/1 HTTP/1.1" 200 -
```

If we repeat this request, we find this item already in the cache, so it is “Cache Hit”, and the time taken is **0.00000** seconds**.**

```
Book info endpoint. Using catalog server: http://localhost:5003
Cache Miss! Item Number: 1, Cache Capacity: 1/100, Time Taken: 2.02860 seconds
127.0.0.1 - - [12/Jan/2024 23:30:37] "GET /info/1 HTTP/1.1" 200 -
Book info endpoint. Using catalog server: http://localhost:5000
Cache Hit! Item Number: 1, Cache Capacity: 1/100, Time Taken: 0.00000 seconds
127.0.0.1 - - [12/Jan/2024 23:30:56] "GET /info/1 HTTP/1.1" 200 -
```

We also note in the screen above that in the first info request the choice was made on the catalog replica server 5003, and upon request again, the choice was signed on the catalog replica server 5000 in the so-called load balancing (round robin).

This is a screen from terminal of catalog replica server 5000 to see the response of the request

```
Replica 1 on Port 5000: Catalog Info! Item Number: 1
127.0.0.1 - - [12/Jan/2024 23:30:37] "GET /info/1 HTTP/1.1" 200 -
```

Let’s try a search request by frontend server:

```
C:\Users\ZBOOK>curl http://localhost:5002/search/distributed%20systems
[
  {
    "id": "1",
    "title": "How to get a good grade in DOS in 40 minutes a day"
  },
  {
    "id": "2",
    "title": "RPCs for Noobs"
  }
]
```

We see here again “Cache Miss” for the first time and the size of the cache has increased, with time taken **2.05878** seconds. And when was “Cache Hit” the time taken is 0.00000 seconds.

```
Search endpoint. Using catalog server: http://localhost:5003
Cache Miss! Item Name: distributed systems, Cache Capacity: 2/100, Time Taken: 2.05878 seconds
127.0.0.1 - - [12/Jan/2024 23:31:42] "GET /search/distributed%20systems HTTP/1.1" 200 -
Cache Hit! Item Name: distributed systems, Cache Capacity: 2/100, Time Taken: 0.00000 seconds
127.0.0.1 - - [12/Jan/2024 23:32:20] "GET /search/distributed%20systems HTTP/1.1" 200 -
```

This is a screen from terminal of catalog replica server on 5003, to see the response details of the request above:

```
 * Running on http://127.0.0.1:5003
 * Running on http://192.168.1.103:5003
Press CTRL+C to quit
 * Restarting with stat
Replica 2 on Port 5003: Catalog Server Running on Port 5003
 * Debugger is active!
 * Debugger PIN: 850-146-242
Replica 2 on Port 5003: Catalog Search! Item Name: distributed systems
127.0.0.1 - - [12/Jan/2024 23:31:42] "GET /search/distributed%20systems HTTP/1.1" 200 -
```

- Invalidate Request

Let’s try get information of item number 2:

```
C:\Users\ZBOOK>curl http://localhost:5002/info/2
{
  "price": 25.0,
  "quantity": 20,
  "title": "RPCs for Noobs"
}
```

Now let’s send a purchase request for this item, the round robin select order replica server 5004 for that, and after the purchase we see the message “Cache invalidated successfully for item 2” with response of invalidate cache endpoint that implemented by order server to frontend server, and this message to tell the server that this item was changed to remove it from the cache.

```
Book info endpoint. Using catalog server: http://localhost:5003
Cache Miss! Item Number: 2, Cache Capacity: 1/100, Time Taken: 2.08206 seconds
127.0.0.1 - - [12/Jan/2024 23:42:18] "GET /info/2 HTTP/1.1" 200 -
Purchase endpoint. Using order server: http://localhost:5004
Cache invalidated successfully for item 2
127.0.0.1 - - [12/Jan/2024 23:42:48] "POST /invalidate_cache/2 HTTP/1.1" 200 -
```

Here is we see the quantity of item 2 before and after purchase process.

```
C:\Users\ZBOOK>curl http://localhost:5002/info/2
{
  "price": 25.0,
  "quantity": 20,
  "title": "RPCs for Noobs"
}

C:\Users\ZBOOK>curl -X POST http://localhost:5002/purchase/2
{
  "message": "Book RPCs for Noobs purchased successfully"
}

C:\Users\ZBOOK>curl http://localhost:5002/info/2
{
  "price": 25.0,
  "quantity": 19,
  "title": "RPCs for Noobs"
}
```

And the same invalidate request is sent to the frontend server when updating an item, I will put a screen and point that in the next part.

- Consistency

Let’s try updating the quantity and the price of item 4, note that we can update any value we want, price or quantity or both as I indicated in the part 1.

```
C:\Users\ZBOOK>curl http://localhost:5002/info/4
{
  "price": 12.0,
  "quantity": 11,
  "title": "Cooking for the Impatient Undergrad"
}

C:\Users\ZBOOK>curl -X PUT -H "Content-Type: application/json" -d "{\"price\": 25.
0, \"quantity\": 20}" http://localhost:5003/update/4
{
  "message": "Book updated successfully"
}

C:\Users\ZBOOK>curl http://localhost:5002/info/4
{
  "price": 25.0,
  "quantity": 20,
  "title": "Cooking for the Impatient Undergrad"
}
```

And here as we can see that after the cache was hit for item 4, it became a miss because the invalidate request from catalog server, so the cache deleted the item and next time sent the request to the server.

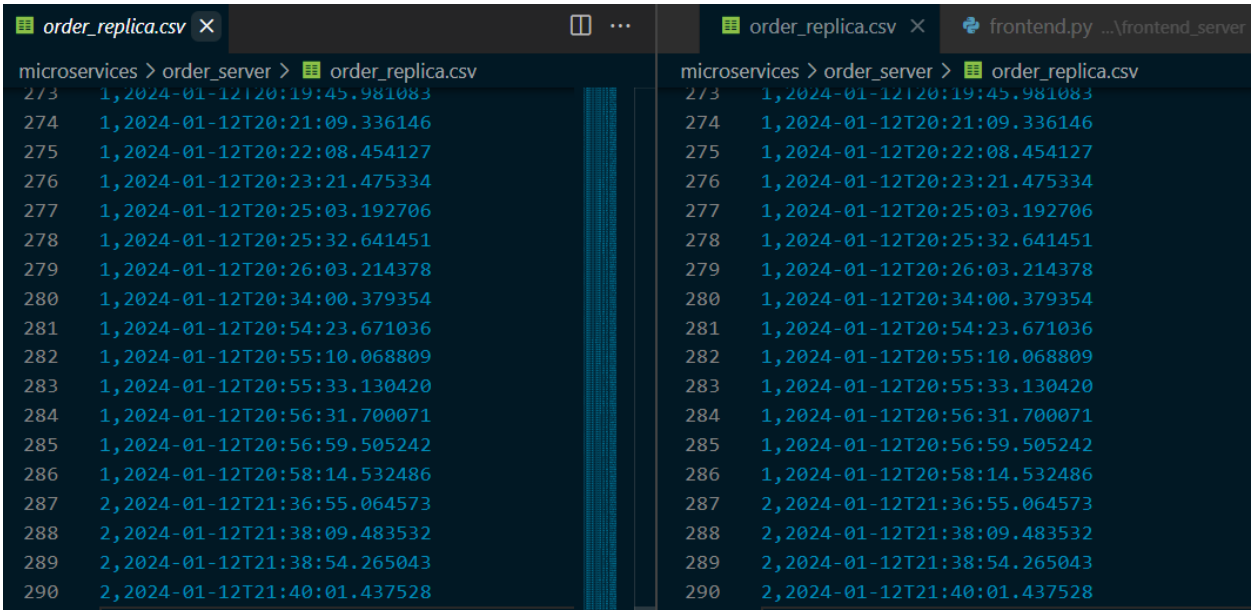
```
Book info endpoint. Using catalog server: http://localhost:5003
Cache Miss! Item Number: 4, Cache Capacity: 5/100, Time Taken: 2.04154 seconds
127.0.0.1 - - [12/Jan/2024 23:48:28] "GET /info/4 HTTP/1.1" 200 -
Book info endpoint. Using catalog server: http://localhost:5000
Cache Hit! Item Number: 4, Cache Capacity: 5/100, Time Taken: 0.00000 seconds
127.0.0.1 - - [12/Jan/2024 23:48:38] "GET /info/4 HTTP/1.1" 200 -
Cache invalidated successfully for item 4
127.0.0.1 - - [12/Jan/2024 23:49:24] "POST /invalidate_cache/4 HTTP/1.1" 200 -
Book info endpoint. Using catalog server: http://localhost:5003
Cache Miss! Item Number: 4, Cache Capacity: 5/100, Time Taken: 2.03830 seconds
127.0.0.1 - - [12/Jan/2024 23:49:39] "GET /info/4 HTTP/1.1" 200 -
```

For the above update request, here is the terminal of catalog replicas on servers 5000 and 5003, they notifying each other when updating any data, to ensure the consistency between replicas databases, which ensure that the information matches between them.

```
Replica 1 on Port 5000: Book updated successfully (Replica)
127.0.0.1 - - [12/Jan/2024 23:55:18] "PUT /update_replica/4 HTTP/1.1" 200 -
```

```
Replica 2 on Port 5003: Book updated successfully (Replica)
127.0.0.1 - - [12/Jan/2024 23:55:21] "PUT /update_replica/4 HTTP/1.1" 200 -
```

And of course the same goes for the order replicas databases, as we see:



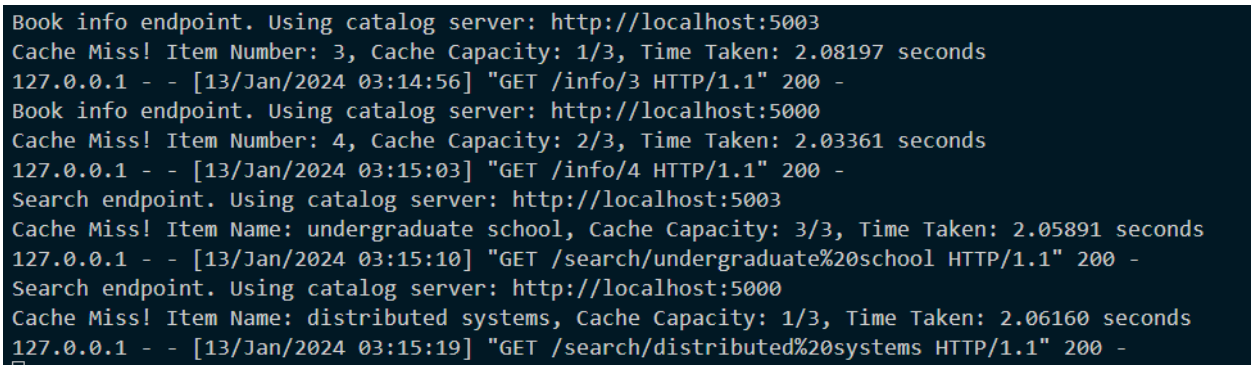
It should be noted that at each time of each purchase request, each of the following is notified:

- Catalog replicas, to update their databases.
- Order replicas, notifying each other to record the new order in their databases.

Here is the terminal of one order replicas to see the response:

```
127.0.0.1 - - [13/Jan/2024 02:59:10] "POST /purchase/2 HTTP/1.1" 200 -
127.0.0.1 - - [13/Jan/2024 02:59:52] "POST /notify_purchase/2 HTTP/1.1" 200 -
127.0.0.1 - - [13/Jan/2024 02:59:54] "POST /notify_purchase/2 HTTP/1.1" 200 -
```

Let’s minimize the cache size to see how to act if the maximum size is reached, it’s supposed to clear its data and starts over:



I would have liked to implement the dockerization part, but unfortunately, time constraints and being alone working prevented it.