**Bazar.com** **(Lab2)**

*Introduction*

In this time, we will upgrade and improve the performance of our application due to increasing of loading on the basic server from lab1. We will use caching (proxy) in the implicitly in the frontend server. The catalog servers (two servers replicated) can send invalidate messages to the frontend (proxy) when an object in the database is updated.

By this process, the frontend removes the dirty object from cache so that if the user requests the object again, the proxy will have the clean object in its cache!

We created two types of caches, lookup cache & search cache both of them hold the tag – value. In the case of lookup cache, the tag is the ID of the book, and the value is the book object coming from DB. In case of search cache, the tag is the topic, and the value is an array of books that its topic is the same as the tag.

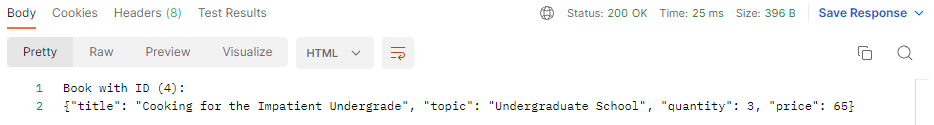
When the user performs purchase operation, the catalog edits the quantity in the stock, send to the other replica to update its quantity too, so that saving the consistency manner. Order1 talks with Catalog1 server and Order2 talks with Catalog2 server.

*Demo*

We will show you a GET method which gets a book by ID (info operation), as we request, the frontend server (same as proxy) will cache the object, so in the second time will fetch the object from cache, we will also show you the difference between response times:

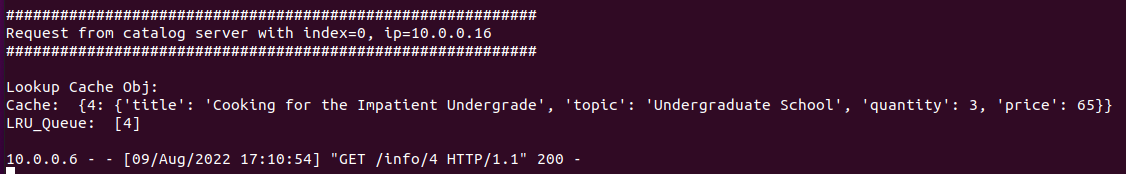
First time without caching:



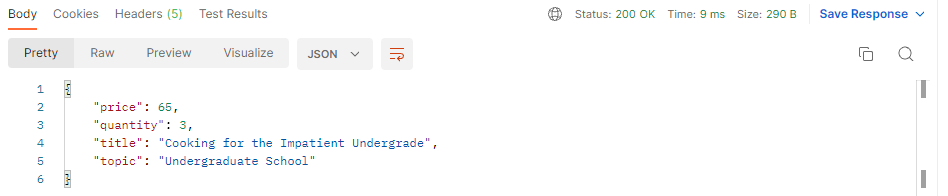


We can see that the time = 25ms.

From console of frontend (the request is done from catalog has IP address: 10.0.0.16:5000)

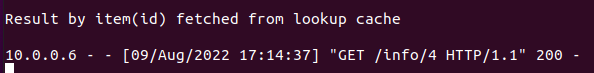


Request the same object, will fetch from cache with less response time:



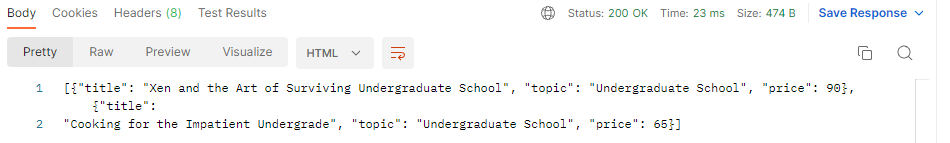
We can see the time = 9ms, which is a large difference!!

From frontend server’s console:

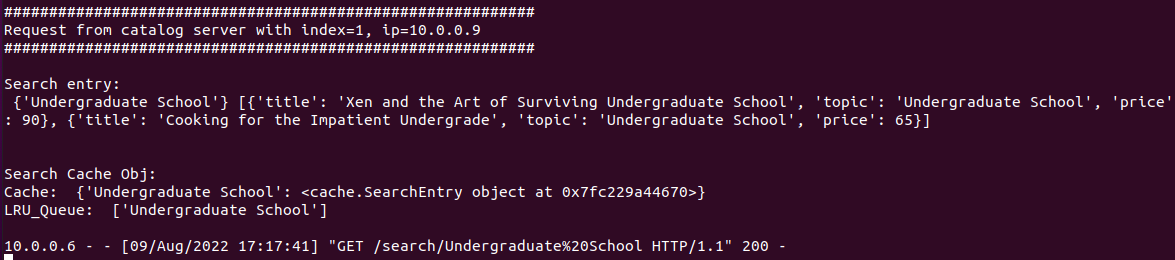


Now we will fetch all books with topic “Undergraduate%20School”, the request should be done from the 2nd replica with IP address: 10.0.0.9:5003:

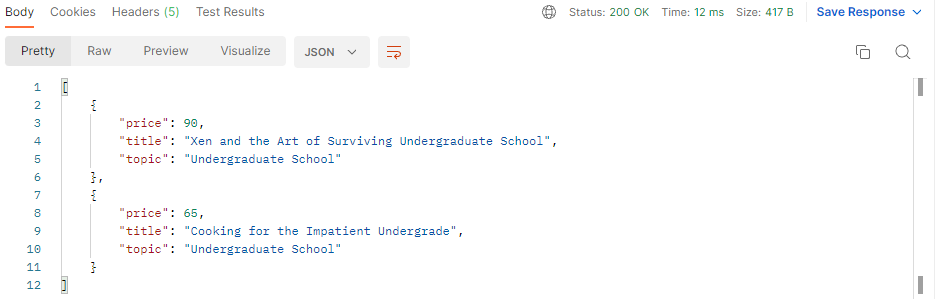




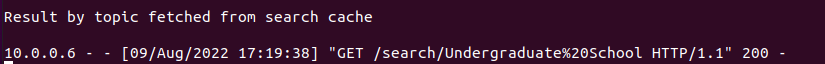
Response time is 23ms. From console of frontend server:



Retry the same request:



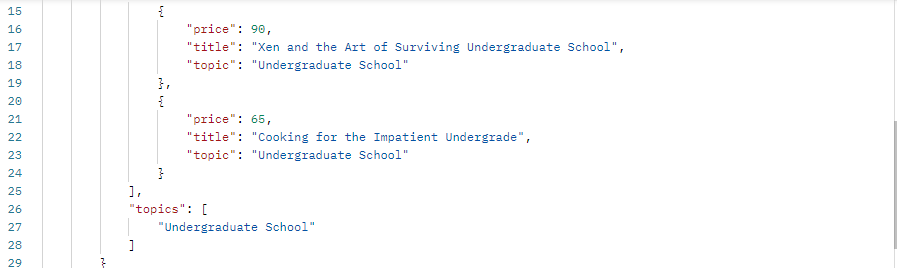
Response time = 12ms!!



*Now we will show you the cache, we build an API to see the content of the cache:*



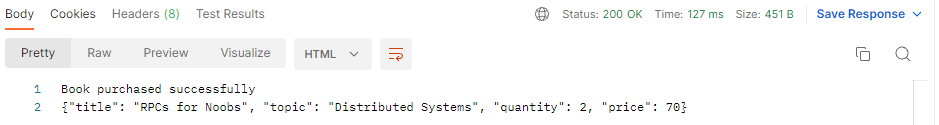




Now we will do a purchase operation, we have in two DBs all books with quantity = 3. We are going to buy a book with id = 2. So, let’s see:



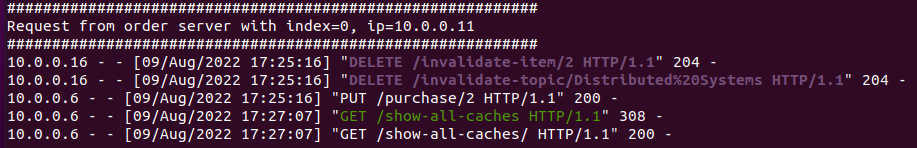
PUT method, the result and response time:



Response time = 127!! Relatively High!!

The cache will not affect, because the book with id = 2 is not in cache before!

Console (front-end):

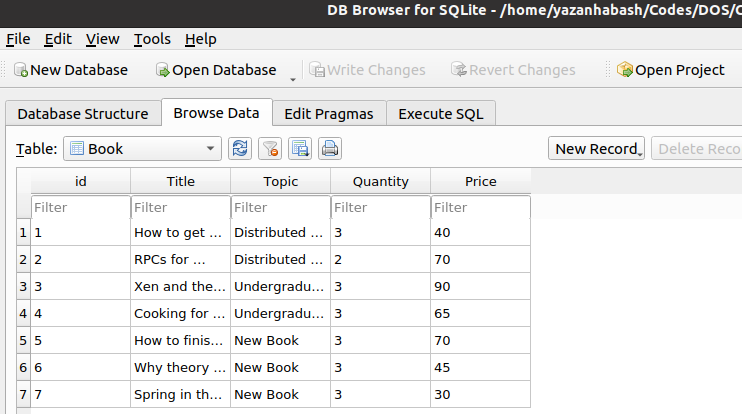


*We can see the steps of requests: invalidation first, purchase to order server (first replica: 10.0.0.11) then we did see-all-caches (cache not affected) as above:*

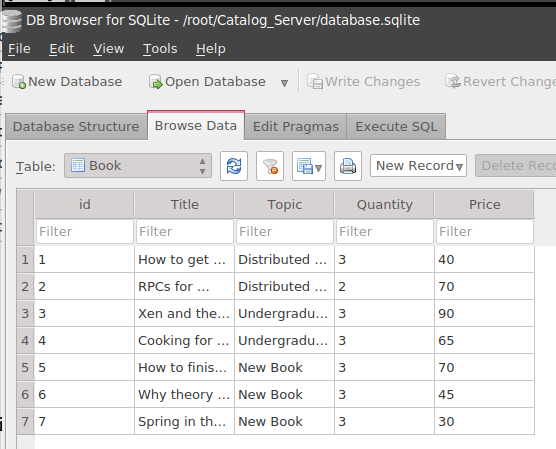


See the databases, the values become 2 instead of 3:

First replica (Ubuntu 20.04):



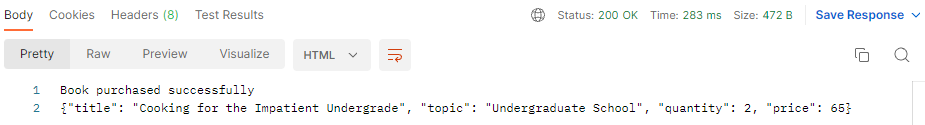
Second replica (Puppy Linux):



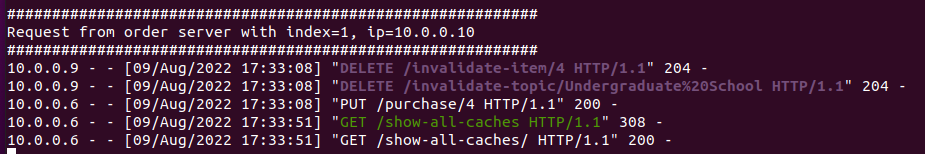
Purchase book with id = 4 (exists in cache):



Result:



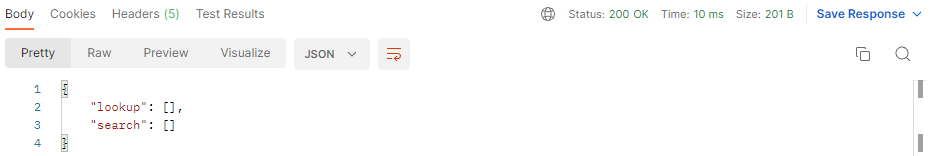
Console of front-end:



*We can see the steps of requests: invalidation first, purchase to order server (second replica: 10.0.0.9) then we did see-all-caches (cache affected, becomes empty):*

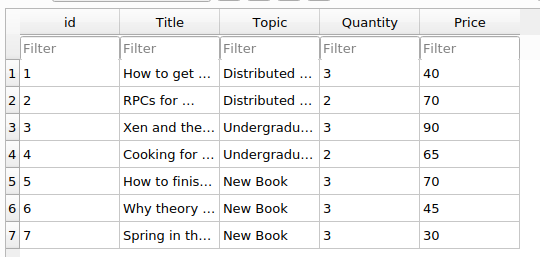
The caches will be empty due to invalidate messages from catalog replicas:



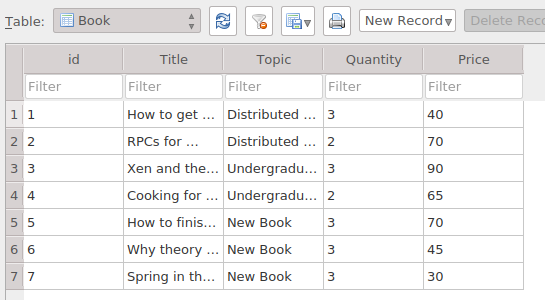


See DBs:

Ubuntu (first replica), book id = 4, quantity becomes 2:



Second replica (Puppy Linux), as the same as above:



**Yazan Habash & Ashraf Hab-Rumman**