Rasheed Hendawi & Mohamad Rehan

Najah National University

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

In this part we upgrade the architecture of the application into a new level of scalability, we have used a lot of methods such as Round Robin for the load balancing, caching and make replicas of instances for each service (Catalog, Order).
First we will talk about how the data flow then we will give photos and metrics for the improvements between cache hit and cache miss.

How does the data flow:

The use will interact with the Frontend service which is just a simple CLD application that provide some services to the user to interact with the system, also the Frontend service have a lot of functionality such as Load-Balancing using Round Robin to handle the requests that come from the user and balance it on the replicas for the catalog and order services, it also handles the caching service so it provide two data structure to handle the cache in the memory and handle the deletion from the memory if the user did a write operation on the data that is already saved on the memory and give direct cache miss if the user request that specific input again.
After the Frontend service handle the requests from the user it rout these requests to the order and catalog replicas so it can handle the request and give back the data as Json file, by the way all the calling methods are done via an HTTPS protocol.

```
1. View All Catalog Items
2. View Catalog Item
3. Purchase Book
4. Search Book Topic
5. Exit
Select an option: 1

            View All Books:
Title                                      Price   Quantity
-----------------------------------------------------------------------
How to get a good grade in DOS in 40 ...     ¤40.00          7
RPCs for Noobs                               ¤50.00          7
Xen and the Art of Surviving Undergra...     ¤30.00          2
Cooking for the Impatient Undergrad          ¤25.00          0
How to finish Project 3 on time              ¤35.00          6
Why theory classes are so hard.              ¤55.00         12
Spring in the Pioneer Valley                 ¤20.00         10
```

We can notice that the first view item option will get the data from the first replica as shown in the figure.

```
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
catalog-1          |       Request starting HTTP/1.1 GET http://catalog-1:8080/catalog/health - - -
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
catalog-1          |       Request finished HTTP/1.1 GET http://catalog-1:8080/catalog/health - 200 - application/json;+c
harset=utf-8 85.1882ms
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
catalog-1          |       Request starting HTTP/1.1 GET http://catalog-1:8080/catalog/info - - -
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
catalog-1          |       Request finished HTTP/1.1 GET http://catalog-1:8080/catalog/info - 200 - application/json;+cha
rset=utf-8 7.4676ms
```

But if we enter the view item again, we will see it coming from the next replica as the Round Robin says.

```
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
catalog-1          |       Request starting HTTP/1.1 GET http://catalog-1:8080/catalog/health - - -
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
catalog-1          |       Request finished HTTP/1.1 GET http://catalog-1:8080/catalog/health - 200 - application/json;+c
harset=utf-8 85.1882ms
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
catalog-1          |       Request starting HTTP/1.1 GET http://catalog-1:8080/catalog/info - - -
catalog-1          | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
catalog-1          |       Request finished HTTP/1.1 GET http://catalog-1:8080/catalog/info - 200 - application/json;+cha
rset=utf-8 7.4676ms
catalog-2          | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
catalog-2          |       Request starting HTTP/1.1 GET http://catalog-2:8080/catalog/health - - -
catalog-2          | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
catalog-2          |       Request finished HTTP/1.1 GET http://catalog-2:8080/catalog/health - 200 - application/json;+c
harset=utf-8 71.5049ms
catalog-2          | info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
catalog-2          |       Request starting HTTP/1.1 GET http://catalog-2:8080/catalog/info - - -
catalog-2          | info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
catalog-2          |       Request finished HTTP/1.1 GET http://catalog-2:8080/catalog/info - 200 - application/json;+cha
rset=utf-8 8.8750ms
```

Now for the caching section if we order an item with the ID one, we will see the it will give cache miss as follows.

```
Enter item ID: 1
🔄      Cached data miss:

[ID:1] How to get a good grade in DOS in 40 minutes a day – $40 – Qty: 7

⏱️      Cache MISS (Info): 31 ms
1. View All Catalog Items
2. View Catalog Item
3. Purchase Book
4. Search Book Topic
5. Exit
Select an option:
```

But if we revisit the same item, we will find it in the cache in the memory because it was ordered more than one time at least as follows.

Rasheed Hendawi & Mohamad Rehan
Najah National University

```
Enter item ID: 1
🔄      Using cached data:

[ID:1] How to get a good grade in DOS in 40 minutes a day – $40 – Qty: 7

⏱️      Cache HIT (Info): 0 ms
1. View All Catalog Items
2. View Catalog Item
3. Purchase Book
4. Search Book Topic
5. Exit
Select an option:
```

we can see the time difference which shows that the caches is must faster than cache miss that went the data base to get the information about the books in the library.

```
Select an option: 3
Enter Book ID to purchase: 1
✅Order placed!
🔄Synced decrement to: http://catalog-2:8080
1. View All Catalog Items
2. View Catalog Item
3. Purchase Book
4. Search Book Topic
5. Exit
Select an option:
```

Here in the above figure we can see that if the item have been purchased it will give an indicator that the purchase done successfully and the data is synced to the other replicas like http://catalog-2:8080 because the load balancer used the http://catalog-1:8080 as a service provider for the purchase.

But we will notice that it must not be in the cache after that because the data has been changed so it must be removed from the memory and to check that we can search for the Item with the ID - > 1 and check if it gives cache miss or cache hit so we must get a cache miss as follows.

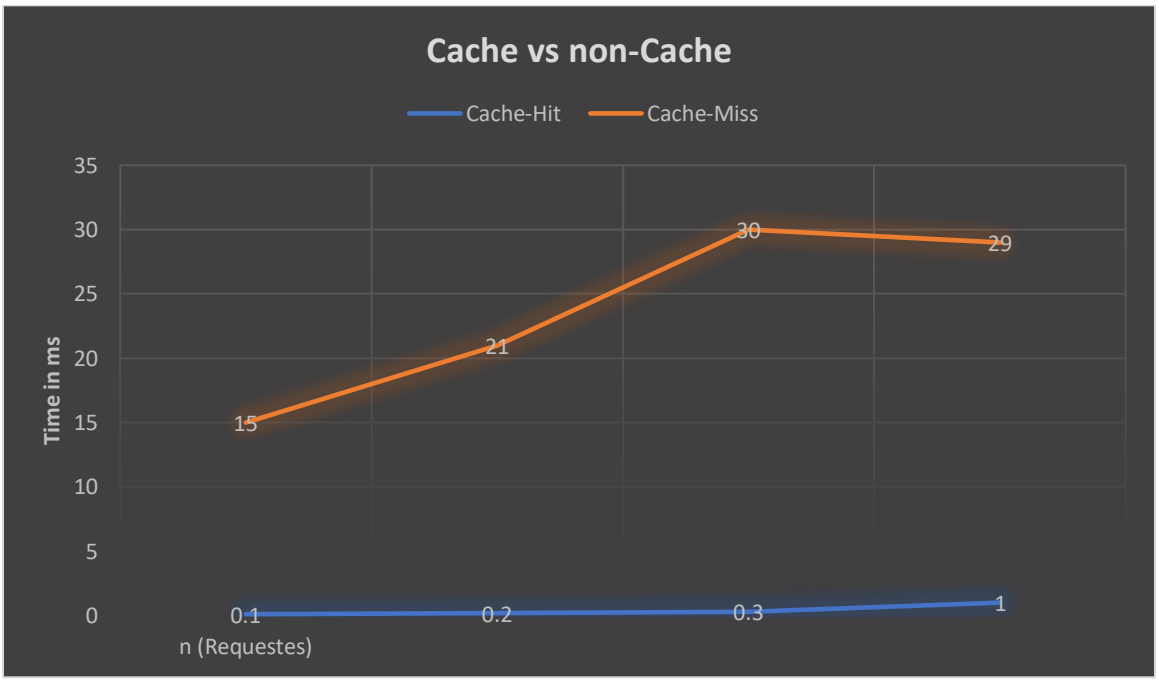
Rasheed Hendawi & Mohamad
Rehan
Najah National University

```
Enter item ID: 1
🔄      Cached data miss:

[ID:1] How to get a good grade in DOS in 40 minutes a day – $40 – Qty: 6

⏱️      Cache MISS (Info): 17 ms
1. View All Catalog Items
2. View Catalog Item
3. Purchase Book
4. Search Book Topic
5. Exit
```

Yes, we can see that it gives a cache miss with 17ms also the quantity has been decremented by 1.

Performance and Measurements

We will see the improvements of our upgrade for the first part here is a little graph that shows the difference between using cache in the frontend service.



But we can see that the trade off will have some overhead in the sync for all the other replicas but in the other hand we can scale the project to be unlimited to a such number of users.

Rasheed Hendawi & Mohamad Rehan

Najah National University

How to Run the Code

First use docker compose up –build and then after opening the frontend service u just simply hit the dotnet run UserAction.dll.
After Just hitting those two commands u will be able to use the program.