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

Distributed Operating Systems

Dr Samer Arandi

By Zain Abubaker && Abd-Alkareem Al Masri

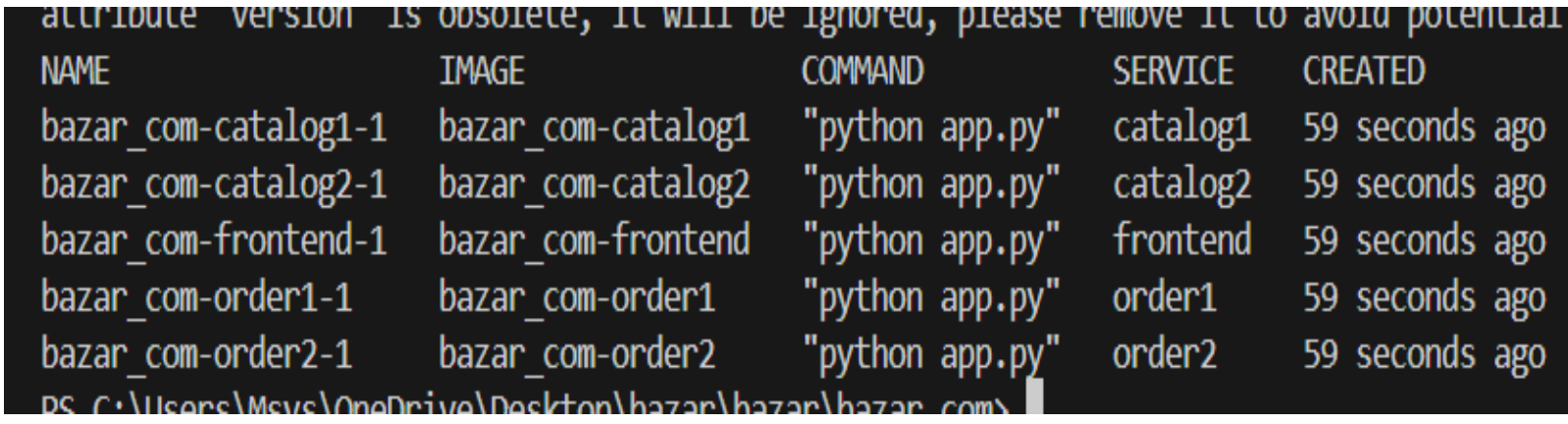
Overview

A containerized microservices-based bookstore.
Enhanced from Lab 1 with caching, replication, load balancing, and consistency management.

System Architecture

5 services

- ➔ frontend: manages all incoming requests, load balancer, and in-memory cache.
- ➔ Catalogs 1 and 2: replicate updates and serve book data.
- ➔ Orders 1 and 2: manage purchases and duplicate logs and updates.
- ➔ REST APIs are used by all services to communicate.
- ➔ Orchestration is done with Docker Compose.



Caching Mechanism

➔ Implemented in-memory cache in frontend.
➔ Cache stores /info/item_id responses.
➔ TTL is 60 seconds.
➔ Cache is invalidated when a purchase or update occurs.
➔ Optional extension: LRU policy can be added.


```
{
    "price": 30.0,
    "quantity": 9,
    "title": "How to get a good grade in DOS in 40 minutes a day"
}
```


Replication & Load Balancing

➔ Two replicas each for `catalog` and `order`.
➔ Frontend uses round-robin strategy for both services.
➔ Replicas sync using REST calls:
- **catalog1** sends updates to catalog2 and vice versa.
- **order1** syncs purchase to order2 and vice versa.

➔ Backend updates invalidate the cache via /invalidate/<item_id> on frontend.

When running POST http://localhost:5000/purchase/1


```
{
    "message": "bought book How to get a good grade in DOS in 40 minutes a day"
}
```


And then we run GET http://localhost:5000/info/1, we will notice a decrease in quanity

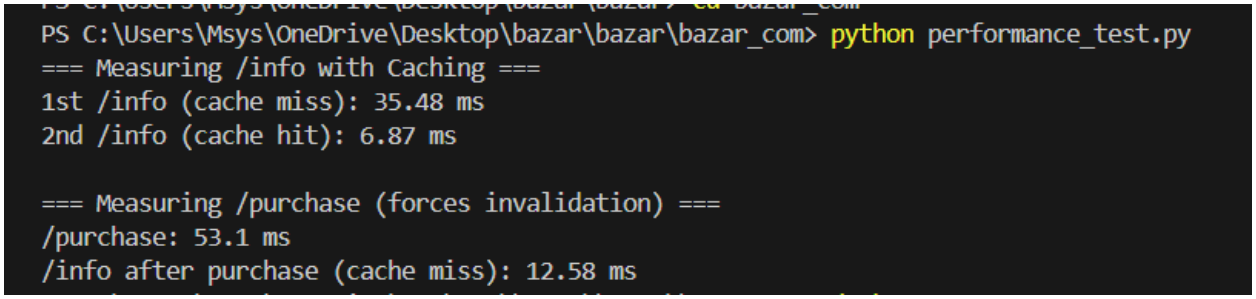

```
{
    "price": 30.0,
    "quantity": 8,
    "title": "How to get a good grade in DOS in 40 minutes a day"
}
```

Consistency Strategy

➔ Strong consistency via server-push invalidation.
➔ Writes are synchronized across replicas.
➔ Only original replica performs the stock update; replica just acknowledges.

Performance Evaluation

Operation	Time (ms)
/info (1st call - cache miss)	35.48
/info (2nd call - cache hit)	6.87
/purchase	53.10
/info after invalidation	12.58



Observations:

➔ Cache hits reduce latency by -80%.

➔ Invalidation ensures correctness with acceptable latency overhead.