

November 2024

Report: Bazar Project

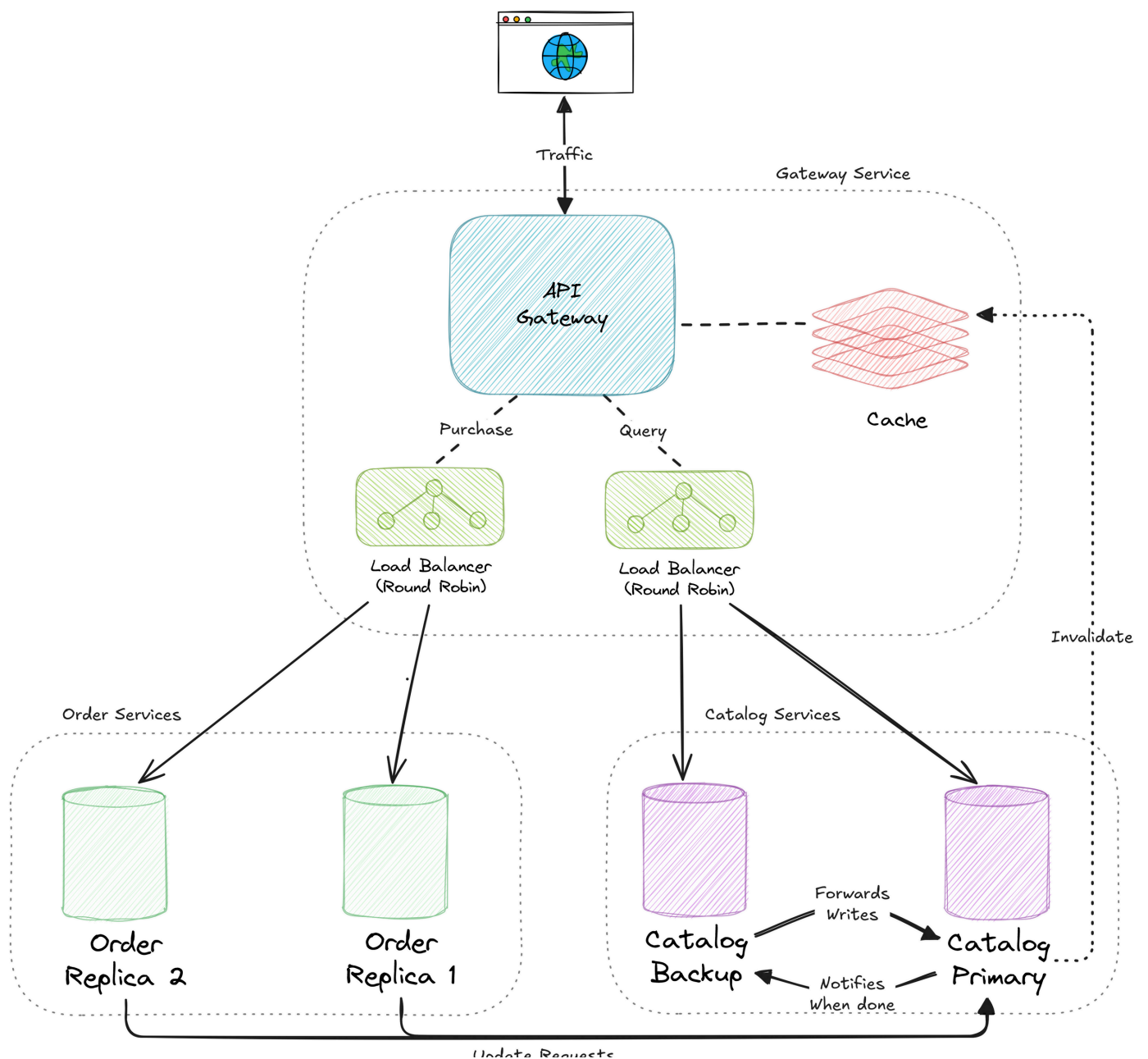
Supervised by:
Dr. Samer Arandi

Students:

- Izzat Alsharif
- Asem Diab

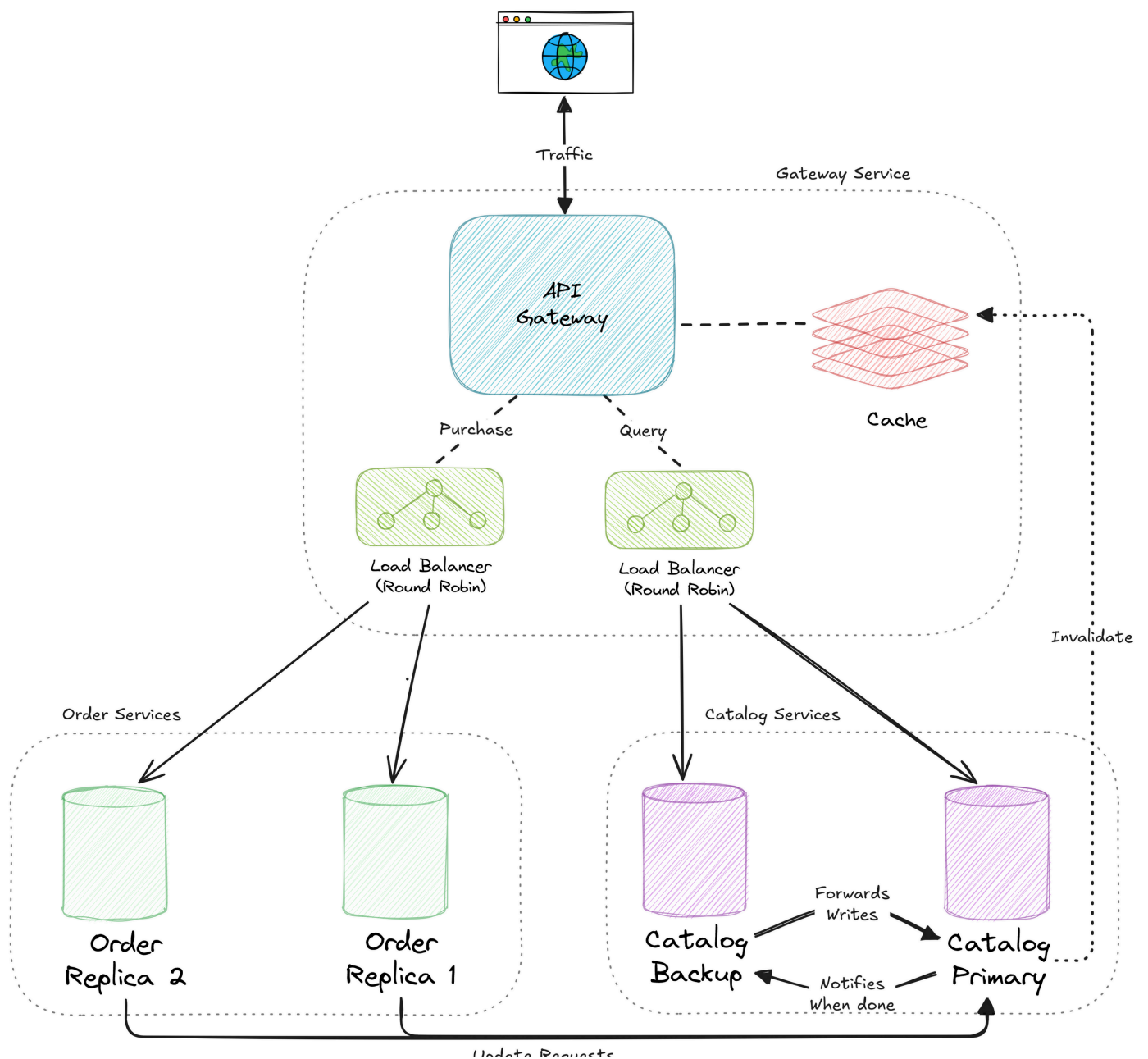
Architecture

The Bazar application uses a microservices-based architecture, with each service responsible for a specific domain of the application. These services communicate with each other using REST. The system includes caching and replication to improve performance and ensure fault tolerance.



Architecture

The Bazar application uses a microservices-based architecture, with each service responsible for a specific domain of the application. These services communicate with each other using REST. The system includes caching and replication to improve performance and ensure fault tolerance.



Cont.

1. Gateway Service:

- Acts as the entry point for all client traffic.
- Handles routing, caching, and load balancing for the backend services. It uses Round-Robin load balancing algorithm to distribute load evenly among nodes.
- Ensures high availability and improved performance through in-memory caching and intelligent request distribution.
- An in-memory cache in the Gateway Service stores frequently accessed data (e.g., book details). It uses LRU method to eliminate data when cache size limit is hit.
- Includes an invalidation mechanism triggered by updates to ensure consistency. (The primary catalog service invalidates cache entries after any update).

2. Catalog Service:

- Maintains the book catalog, including details such as stock, price, and topic.
- Implements a primary-backup replication for fault tolerance and performance:
 - The primary instance handles all write operations, and syncs updates with the backup.
- Integrates with the gateway for cache invalidation upon updates.

3. Order Service:

- Processes purchase requests, verifies stock availability, and updates catalog data.
- Uses multiple replicas to handle increased loads.
- Works with the gateway for balanced request distribution.

Project Structure

The application is version controlled via the github organization: [Bazar Platform](#). And it includes the following repositories:

- **bazar-gateway-service**: Handles user interactions and routes requests to backend services.
- **bazar-catalog-service**: Manages the book catalog, including stock levels and book details.
- **bazar-order-service**: Manages order requests, verifies stock availability, and processes purchases.
- **.github**: Stores project documentation, including design documents.

Endpoints

- **Gateway Service (Front-End)**
 - GET /search/<topic>: Search for books by topic.
 - GET /info/<item_number>: Get information about a specific book.
 - POST /purchase/<item_number>: Purchase a specific book.
- **Catalog Service (Back-End)**
 - GET /query?topic=<topic>: Query books by topic.
 - GET /query?item_number=<item_number>: Query book details by item number.
 - PUT /update/<item_number>: Update stock or price information.
- **Order Service (Back-End)**
 - POST /purchase/<item_number>: Handle purchase requests, verify stock, and update quantities.

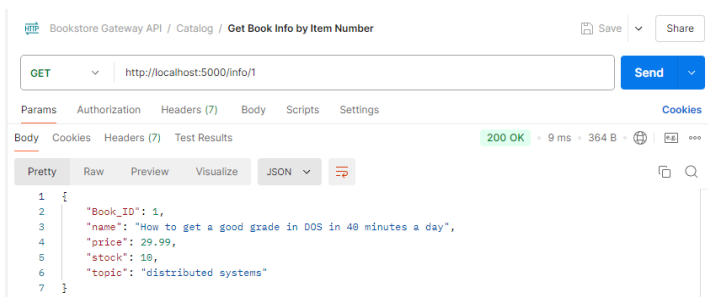
Performance Measurements

The main performance improvement that was implemented in the second part of the project (Lap 2) is the gateway caching mechanism. Replicating does affect performance but is only noticeable when there is concurrent requests which is hard to test with postman.

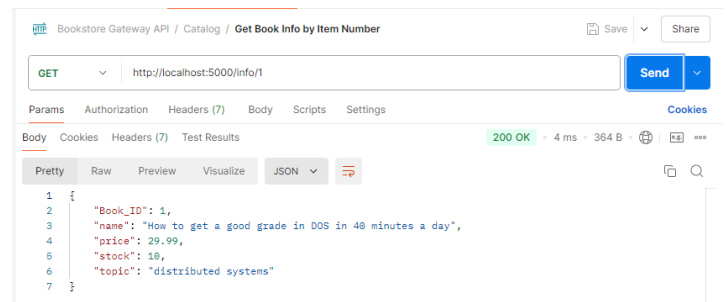
For most requests it takes about **6 - 8 ms** to execute them. When caching is used this remains the case only for the initial request. On repeated requests we get cache hits which reduces the execution time to about **2 - 3 ms**.

When a cache entry is invalidated, the next request for it will increase back to **6 - 8 ms** because we get a cache miss.

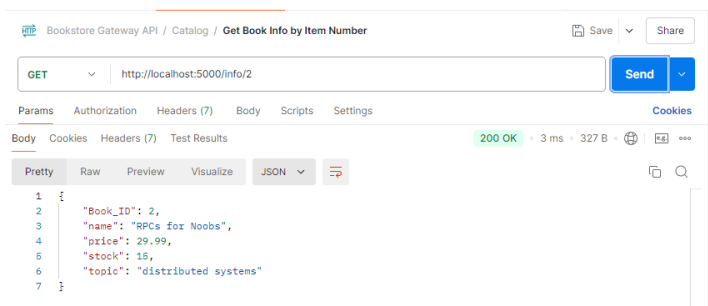
Initial Request For Item 1



Repeated Request for Item 1



Repeated Request For Item 2



Server Logs

```

Fetching info for book ID: 1 from
http://bazar-catalog-primary:5001
Cache hit for book ID: 1
Cache hit for book ID: 1
Cache hit for book ID: 1
Fetching info for book ID: 2 from
http://bazar-catalog-backup:5001
Cache hit for book ID: 2

```