Inventory manager for shop

Jan Pastucha 193662, Wiktor Nazaruk 190454, Maksymilian Pek 193595

1. Description

The inventory management system is designed to help big online stores, that use drop selling method, first in first served. It involves a lot of people trying to buy the same things simultaneously. It tracks and handles inventory in real-time across warehouses. Redis allows system to enforce atomic operations when multiple users attempt to buy the same product at the same time, preventing overselling. Pub/Sub functionality grants ability to constantly update users about low stock in the stores or warehouse. By using Redis we ensure that system can handle high traffic from multiple different stores using it at the same time.

Requirements and assumptions:

- High concurrency support to handle traffic from multiple users and stores simultaneously

- Distributed warehouses and stores require synchronized inventory updates in real time

- Atomic operations to prevent overselling and maintain consistent stock levels

- Users rely on low-stock and out-of-stock notifications for decision-making and effective inventory management

2. Redis features, that would be beneficial:

2.1 Data Types (Hash)

Hashes can store item stock levels, prices, and locations of each product in a way which is easily accessible – by ID of the product. It allows fast access to the specific fields, makes storing data more efficient and enables real-time inventory management, with usage of atomic updates.

Benefits:

- Efficient insertions and deletions
- Avoiding duplicates
- Requires less memory compares to other data structures

Operations:

1. Storing Product Details: Store inventory details like **stock_level**, **price**, and **location** for each product using hashes.

> HMSET inventory:product:12345 name "Laptop" stock_level 50 price 1200 location "Warehouse_A"

2. Updating Stock Level: Update only the **stock_level** field when an item is sold or restocked, which helps avoid retrieving the entire hash.

> HSET inventory:product:12345 stock_level 45

3. Updating Price: Modify the **price** of a product without affecting other fields.

> HSET inventory:product:12345 price 1150

4. Retrieving Product Details: Retrieve all information for a specific product, including **stock_level, price**, and **location**.

> HGETALL inventory:product:12345

2.2 Pub/Sub (Publish/Subscribe)

Redis Pub/Sub allows real-time notifications about the amount products in an inventory. It will help to alert relevant systems or administrators when stock drops below a threshold. When a popular item's stock is low, Redis can broadcast a message to trigger restocking actions or notify users who signed up for restock alerts.

Benefits of Pub/Sub for Real-Time Inventory Management:

- Instant Notifications to Stakeholders
- Scalability for Large Subscriber Base
- Reduced Risk of Stockouts

Operations:

1. Setting Up Redis Servers for Pub/Sub Communication:

> redis-server --port 6380 --slaveof 127.0.0.1 6379
> redis-cli -p 6380

2. Subscribing to Inventory Notifications:

> SUBSCRIBE inventory_alerts

3. Publishing Inventory Alerts:

> PUBLISH inventory_alerts "Stock Alert: Product 12345 is low in stock"
> PUBLISH inventory_alerts "Stock Alert: Product 12345 is now available"

2.3 Transactions for Atomic Inventory Updates

Redis transactions ensure the atomicity of operations. When an item is purchased or restocked, the transaction feature prevents partial updates by grouping all relevant actions - like stock deduction, stock checks, and status updates - into a single transaction. This guarantees that stock data remains consistent.

Benefits:

- Prevents changing the same data by 2 or more users at the same time.
- Ensure data consistency.
- Streamlines Multi-Step Operations.

Operations:

1. Stock Check and Deduction**:**

This transaction first checks what the stock level is, and then we use the value that we got from that operation in the HSET transaction which reduces the stock number by 1, meaning that someone bought the product.

> HGET inventory:product:12345 stock_level

> HINCRBY inventory:product:12345 stock_level -1

> HGET inventory:product:12345 stock_level

> HINCRBY inventory:product:12345 stock_level -2

2. Restock Operation:

When a restocking occurs, multiple fields may need updating, such as the **stock_level** and **last_restocked** date. A transaction ensures that these changes are saved together, providing a consistent view of the product's status.

> HGET inventory:product:12346 stock_level

```
> MULTI

> HINCRBY inventory:product:12346 stock_level 50

> HSET inventory:product:12346 last_restocked "2024-11-14"

> EXEC

> HGETALL inventory:product:12346
```