Recommendation System Optimization Summary

Generated on: 2025-06-12 14:40:16

# 1. Overview

This document summarizes the major performance optimizations and infrastructure improvements made to the Flask-based recommendation system. The goal was to significantly reduce latency for both restaurant and product recommendations using caching, efficient computation, and multi-threaded data fetching.

# 2. Key Improvements

- Integrated Redis for caching full and collaborative recommendations.

- Implemented `ThreadPoolExecutor` to parallelize MongoDB access.

- Precomputed and cached top user neighbors to reduce runtime vector comparison overhead.

- Optimized fallback recommendation logic to reduce redundant DB hits.

- Fixed JSON serialization and Redis setex bug.

- Structured code across modules with clean config handling.

# 3. Redis Caching Setup

Redis was launched via Docker on port 6379. We used `redis-py` to store results from collaborative filtering and full recommendation responses with a 10-day expiry. Keys are namespaced with prefixes like `recommendations:full:{user\_id}`.

# 4. Precomputing Neighbors

The script `precompute\_neighbors.py` was created to precompute top neighbors for each user and store them in Redis. This drastically reduces cold start latency during the first recommendation fetch.

# 5. JMeter Testing Results

We used Apache JMeter to benchmark API endpoints before and after optimization. Below is a comparison:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Initial Load Time (ms) | Post-Optimization Load Time (ms) | Status |
| Restaurant Recommendation (1st call) | 15070 | 18 | ✅ Success |

# 6. Git Commit Message

feat: Optimize recommendation system with Redis caching, threading, and neighbor precomputation  
  
- Integrated Redis caching for full and collaborative results  
- Added parallel MongoDB fetches using ThreadPoolExecutor  
- Implemented precompute\_neighbors.py for top neighbor caching  
- Fixed JSON serialization and undefined variable bugs  
- Validated performance using JMeter