

Bazar Bookstore 2 – Program Output

D. Samer Arandi

Rand Johari	12027653
Abeer Kharouf	12028125

1- Average response time

General Steps for Performance Evaluation (Info & Purchase)

To evaluate the effect of caching and measure performance, we built a script that performs the following steps:

1. **Clear the cache** using the `/cache/clear` endpoint with admin authorization.
This ensures all measurements start from a cold cache (no preloaded data).
2. **Measure response times without cache**
Loop over a list of book IDs and send requests (either `/info/:id` or `/purchase/:id`) one by one for all items.
Since the cache is empty, all responses will come directly from the back-end servers.
3. **Measure response times with cache**
Repeat the same loop with the same list of book IDs.
This time, results should be served from the cache (for `/info`) or benefit indirectly from faster access paths (for `/purchase`, if any).
4. **Log the results** in a structured table showing:
 - ID
 - Time taken without cache
 - Time taken with cache
5. **Compute averages** for both with and without cache cases, then use these metrics to analyze improvements.

a. Query -Info Requests (Caching Evaluation).

To evaluate the benefit of caching, we measured the response time of book information queries (`/info/:id`) with and without cache. Each query was executed against a front-end server that uses an in-memory LRU cache of **size 7 with TTL = 1** minute.

First Run (Includes Cold Start / Warm-up Time)

We cleared the cache and executed the first batch of 7 queries. The results are shown below:


```
C:\Users\pc\bazar-bookstore\bazal-eval>node evaluate-cache-info.js
Clearing cache...
Measuring WITHOUT cache...
Measuring WITH cache...

Results:
ID      WithoutCache(ms)      WithCache(ms)
1        35                    4
2        13                    3
3         5                    3
4         8                    2
5         6                    3
6         6                    2
7         6                    2

AVERAGES:
Without Cache: 11.29 ms
With Cache:    2.71 ms
```

Observation:

- The first query (ID 1) took 35 ms, which is significantly higher than the rest.
- This is due to warm-up overhead — the first network or disk access after starting the server typically involves:
 - Node.js server just finishing startup,
 - internal modules or DB connections initializing,
 - possible file I/O or Docker container spin-ups.

 **Conclusion:** We excluded the first query result from our final measurements to get a more realistic view of steady-state system performance.

Average Response Time (Excluding First Warm-up)

We re-ran the test and calculated averages using the remaining 10 query times. The updated values were:

Query	Without Cache (ms)	With Cache (ms)
2	6.14	2.43
3	5.43	2.00
4	5.43	2.57
5	6.17	2.29
6	5.29	2.43
7	7.43	2.57
8	5.29	2.71
9	5.43	2.57
10	5.00	2.14
11	5.86	2.29
AVG:	5.747	2.40

Why use an average?

Individual response times can slightly vary due to OS scheduling, network jitter, or background processes. Averaging across multiple requests gives a more stable and meaningful performance estimate.

Conclusion:

Caching reduced average query response time from 5.747 ms to 2.40 ms, which is a 58.23% improvement in performance.

This demonstrates the efficiency of using an in-memory cache for read-heavy workloads.

b. Buy Requests (Caching Evaluation).

To evaluate the system's performance for write operations, we measured the response time of the purchase endpoint with and without the cache. Since buy requests directly modify the catalog database and are never served from cache, we do not expect caching to affect their performance.

In the first trial, the very first buy operation took a noticeably higher time (613ms). This is expected and can be attributed to warm-up effects, such as container startup delays, initial DNS resolution, or module loading. Therefore, we excluded it from our average calculations.

```
C:\Users\pc\bazar-bookstore\bazal-eval>node evaluate-cache-buy.js
Clearing cache...
Measuring BUY WITHOUT cache...
Measuring BUY WITH cache...

Buy Results:
ID      WithoutCache(ms)      WithCache(ms)
1        613                    36
2         45                    33
3         39                    34
4         37                    33
5         34                    31
6         33                    35
7         36                    38

AVERAGES:
Buy Without Cache: 119.57 ms
Buy With Cache:    34.29 ms
```

Here is a table showing the response times from 10 subsequent trials

Query	Without Cache (ms)	With Cache (ms)
2	29.29	28.57
3	29.14	29.14
4	30.00	26.43
5	27.14	27.00
6	33.71	26.00
7	27.00	25.14
8	25.86	27.43
9	26.43	26.29
10	26.29	26.57
11	30.29	25.43
AVG:	28.515	26.8

Conclusion:

As expected, the effect of caching on write operations is minimal.

The average response time with caching was **26.6 ms**, compared to **28.9 ms** which is a **6.014%** improvement in performance.

Although the cache is not directly used during /purchase requests, we still observed a slight improvement in response time (around **6%**). This improvement likely comes from **indirect effects** such as reduced overall system load due to caching of previous read requests, warm connections between services, and general stabilization of the system state after initial usage.

2- Cache Consistency:

To evaluate the cost of maintaining cache consistency, we constructed an experiment where we:

1. Cleared the cache initially.
2. Fetched an item using `/info/:id` to ensure a cache miss.
3. Repeated the same request to confirm a cache hit.
4. Performed a `/purchase` operation, which triggers cache invalidation on the front-end.
5. Immediately requested `/info/:id` again to measure the latency of a post-invalidation cache miss.
6. Repeated the request once more to validate the next cache hit.

First query:

```
C:\Users\pc\bazar-bookstore\bazal-eval>node evaluate-consistency.js
Clearing cache...

Initial request (should be MISS)
GET info (miss) took 40 ms

Repeated request (should be HIT)
GET info (hit) took 3 ms

Performing BUY (triggers invalidation)
BUY took 313 ms

Request after BUY (should be MISS)
GET info after BUY (miss) took 4 ms

Request again (should be HIT)
GET info (hit again) took 7 ms

Summary:
First MISS: 40 ms
First HIT: 3 ms
Buy (write): 313 ms
Post-BUY MISS: 4 ms
Final HIT: 7 ms

Overhead of cache consistency 313 ms

latency of a subsequent request that sees a cache miss 4 ms
```

Multiple Trials

To reduce variance, we repeated this experiment 10 times. The following table shows the results:

Query	overhead of cache consistency	latency of a subsequent request
2	17	4
3	16	3
4	17	4
5	13	3
6	14	4
7	13	3
8	12	4
9	14	3
10	12	3
11	13	3
AVG:	14.1	3.4

Conclusion

- Overhead of cache consistency operations (due to write coordination + invalidation) averaged 14.1 ms.
- The latency of a subsequent request that sees a cache miss was very low, averaging just 3.4 ms.
- This confirms that our invalidation mechanism is effective and lightweight, maintaining strong consistency with minimal additional cost.