Experimental Evaluation and Measurement

Running all the component on Edlab machine for several times and with different client numbers, the average latency of each request contrast with lab2 without caching shows below. The clients will concurrently send out data massively, each of them will send out 1100 requests which contain 700 lookup 300 search and 100 buy requests.

**One client**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cache | lab2 | Improvement |
| buy | 20.23 | 30.35 | 10.12 |
| search | 3.57 | 13.3 | 9.73 |
| lookup | 1.82 | 14.4 | 12.58 |

**Three Client**

|  |  |  |  |
| --- | --- | --- | --- |
|  | cache | lab2 | improvement |
| buy | 39.35333 | 189 | 149.64667 |
| search | 4.8222 | 14.17 | 9.3478 |
| lookup | 1.9423 | 42.39 | 40.4477 |

**Five Clients**

|  |  |  |  |
| --- | --- | --- | --- |
|  | lab3 | lab2 | Improvement |
| buy | 83.66 | 224.95 | 141.29 |
| search | 4.846666 | 9.2 | 4.353334 |
| lookup | 1.953428 | 50 | 48.046572 |

We can see that, with cache, the latency of all kinds of request greatly decreased, especially when the number of clients is high. Also, by decreasing the number of lookup and search request sent to the catalog server, the server can handle buy request faster.

Server crash impact

We can see that, when a catalog server crashed, the lookup and search delay slightly increased. However, the buy request delay decreased significantly, this is because the frontend will send buy request to all replica, when a replica crash, it actually decrease the overhead of writing to multiple database and reach consensus.

To mimic the situation where the cache is missing and overhead increase, I set up the client to send out lookup request and buy request to a single item alternatively.

The average latency for a lookup request to get data from DB without cache: 3.89

The average latency for a lookup request to get data from DB with cache missing: 4.1

Which is acceptable.

To test the fault tolerance. I start all servers normally, and during the processing, I create artificially crashing by shut down one of the catalog servers. If the number of items in both server matches after recovery and all client’s request are served, we can draw the conclusion that the system can tolerate k server crashing (theoretically).

A close up of text on a white background

Description automatically generated

Shut down the server A screenshot of a cell phone

Description automatically generated

Relaunch the server

A screenshot of a cell phone

Description automatically generated

Since the clients sent out 500 buy requests to buy book 1 and both servers’ end result matches. The recovery mechanism worked.

How does fault tolerance and disaster recovery works:

When the front-end receives a buy request, it will forward buy request to both server clusters, and it will not forward any new buy request until all servers respond or crash, so both databases are always synchronized when they didn’t crash. When one server with database crash, all requests will be forward to another cluster and get processed. So when the crash server just send out a request to its replica in another cluster to get correct stock, the replica server at this time will lock down the database and will not process any database update related request until the sync is completed.