Documentation of Tests

The tests are designed to check the functionality of the server and database to handle requests from outside clients. We designed the tests around the common user, and situations that could arise from multiple users accessing the same data, or a user mistyping a command.

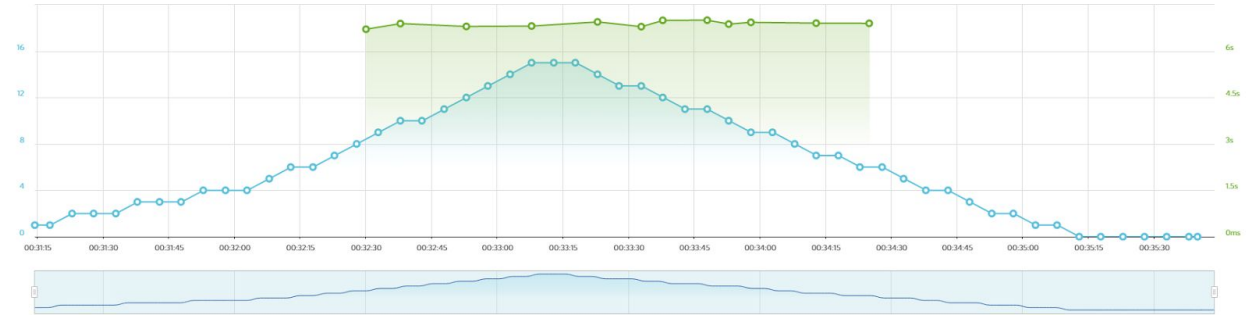
Tests Implemented

The following tests are implemented in our container:

1. The Invalid ID Test: Tests to see if the problem ID is valid and exists in the database.
2. The Key Search Test: Ensures that the Search function is working and returning the requested Value belonging to the Key.
3. The Invalid Key Test: Ensures that the server responds with the proper error codes related to an invalid key provided for a Key Request.
4. Invalid JSON: Tests to ensure the server can handle Json requests that are incomplete or invalid format.
5. The Requests Version Test: Tests to ensure that requests can be sent to create, update, and destroy a Problem.
6. The Version Base Test: Tests the system response if versions become out of date.
7. The Version Conflict Test: Tests the server response if two clients try to update the same version.

In our testing, we determined that typical usage (i.e. a proper request with all correct required parameters) works as expected. In conflict testing, our server correctly fails with the correct error codes.

Server Metrics and Performance Review



Main chart

VUs active VU load time

Based on our testing protocol, we noticed increase in load time at the point of eight users using the server simultaneously with the same methods. We found that requests per second rise proportionally to the load from the users. Load on CPU stays on a slow consistent rise and is not affected as much by peaks in activity. Failure is artificially inflated by some tests intentionally causing error codes based on expected behavior. In other words, we found that as the user load increases, the performance of our server decreases after the eight-user mark accessing the same database. More effective usage of Docker Swarm may have mitigated this issue, as our swarms were running the database on a single server location and therefore queuing their calls.

Conclusion

In conclusion, these series of tests show the correct response of situations that can arise by user input. The server responds accurately to misuse and promptly to proper user events. All of the core functions are working and operational.