Server-Side Performance Testing

Dilbar Singh Lamba

1 Introduction

This report presents a detailed analysis of load tests conducted on a web application hosted on GitHub Pages. The tests were run using `k6`, a performance testing tool, to evaluate the application's capacity to handle an increasing number of Virtual Users (VUs). Tests were conducted with user loads of 100, 1000, 2000, 3000, and finally 10,000 VUs. Additionally, the application often encountered a rate limit, leading to errors and reduced successful request rates.

2 Testing Setup

The testing was performed with the following configurations:

- The application was hosted on GitHub Pages, which imposes rate limits on the number of requests.
- The `k6` tool was used for simulating load and measuring various metrics, such as request duration, response times, request success rates, and failed requests.

Each test was run over a 2-minute duration, with the number of VUs increased incrementally to observe the system's scalability and the effect of GitHub Pages' rate limiting.

3 Test Results and Observations

3.1 100 Users

- **Success Rate:** 100%
- **Request Metrics:** With 100 VUs, all requests were successfully processed without any failures, as shown in Figure 1.
- **Inferences:** The server handled this load without rate limiting, indicating it can sustain light traffic.

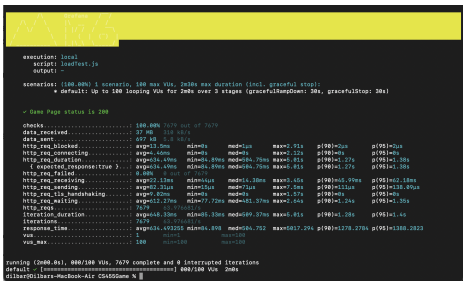

Figure 1: Performance results for 100 Users

3.2 1000 Users

- **Success Rate:** 99.85%
- **Request Metrics:** With 1000 VUs, 99.85% of the requests were successfully processed, with very minimal failures, as shown in Figure 2.
- **Inferences:** The server began to show slight signs of stress but maintained a high success rate, indicating stability under moderate load.

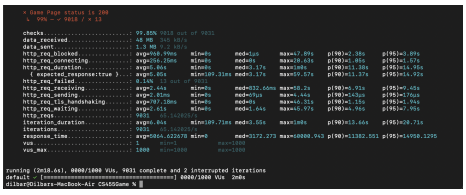

Figure 2: Performance results for 1000 Users

3.3 2000 Users

- **Success Rate:** 96.71%
- **Request Metrics:** With 2000 VUs, the success rate dropped to 96.71%, and rate limiting began affecting the performance, as shown in Figure 3.
- **Inferences:** The GitHub Pages hosting began rate limiting, causing a noticeable decline in successful requests and affecting overall response times.

3.4 3000 Users

- **Success Rate:** 85.10%

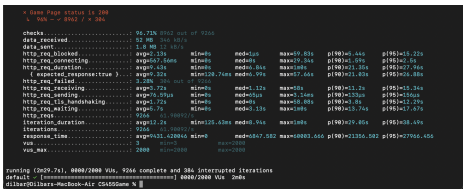

Figure 3: Performance results for 2000 Users

- **Request Metrics:** With 3000 VUs, the success rate dropped further to 85.10%, as shown in Figure 4. Rate limiting by GitHub Pages became more pronounced, with a significant increase in failed requests.
- **Inferences:** The server could no longer sustain the load effectively due to rate limiting, resulting in a notable increase in failed requests.

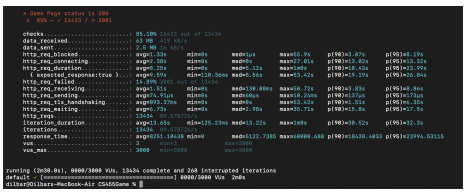

Figure 4: Performance results for 3000 Users

3.5 10,000 Users

Two separate tests were conducted with 10,000 VUs:

- **Test 1:** Success Rate of 21.45% (See Figure 5)
- **Test 2:** Success Rate of 27.59% (See Figure 6)
- **Request Metrics:** Under extreme load, the server's success rate dropped significantly to 21.45% and then 27.59% on subsequent runs. This drastic decline can be attributed to aggressive rate limiting by GitHub Pages.
- **Inferences:** The GitHub Pages hosting environment proved inadequate for such high levels of traffic, with the majority of requests being blocked or delayed. An alternative hosting solution without such rate limits would be required for scalability.

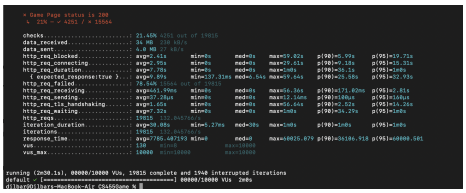

Figure 5: Performance results for 10,000 Users (21.45% requests passed)

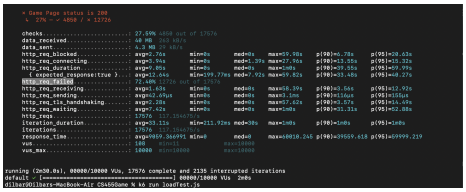

Figure 6: Performance results for 10,000 Users (27.59% requests passed)

4 Additional Observations

During the tests, frequent "Rate Limit Exceeded" errors were encountered, as seen in Figure 7. These errors occurred because GitHub Pages imposes a rate limit on the number of requests that can be made within a specific timeframe. The higher the user load, the more frequently these errors occurred, which directly impacted the success rate of the requests.

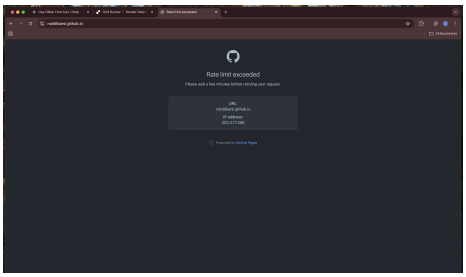

Figure 7: Rate Limit Exceeded error on GitHub Pages

5 Conclusion

The load tests highlighted the limitations of hosting a high-traffic application on GitHub Pages. While the server handled up to 1000 VUs with minimal failures, performance deteriorated sharply at higher user loads due to rate limiting. For scalable and robust performance under heavy loads, migrating the application to a dedicated hosting solution without such rate restrictions is recommended.