Comprehensive Web Application Performance Testing & Analysis

Introduction

Network performance testing refers to a process of analyzing a network's efficiency and quality of service by evaluating metrics such as throughput, speed, download/upload, latency, and packet loss. In modern web development, performance testing plays a crucial role in ensuring that applications remain fast, stable, and reliable under different levels of user activity. As websites grow and user expectations rise, it becomes essential to evaluate how a system behaves under normal traffic, sudden load spikes, and extreme stress conditions. Performance testing helps identify bottlenecks, validate system capacity, and also provide optimization insights.

This article presents a comprehensive performance analysis of the web application https://test.k6.io, a demo site provided by K6 specifically for safe and ethical load testing. Using the K6 performance testing tool, I conducted three main test type:

1. Load Test
    - Load testing is a type of performance testing that evaluates how a system behaves under expected or peak user load. The goal is to measure the system's ability to handle concurrent users, maintain response times, and sustain throughput.
2. Stress Test
    - Stress testing is a type of performance testing where a system is pushed beyond its expected load to determine the maximum capacity the system can handle, at what point the system starts to degrade, and how well the system can recovers after the heavy load is removed
3. Spike Test
    - Spike testing is a type of performance testing that focuses on how a system behaves when it experiences sudden and extreme increases in load for a short period of time

This report documents the methodology, test scripts, data collected, interpretation, and recommendations for optimization. A demonstration video has also been created to walk through the execution process and results.

Tool Selection Justification

In this performance testing, K6 is selected because it is one of the most modern, developer-friendly, and efficient performance testing tools available today. Key reasons for choosing K6 include:

- Lightweight and efficient
  Designed in Go, K6 delivers high performance even on modest hardware.
- JavaScript-based scripting
  Test scenarios are easy to write, maintain, and extend.
- Powerful CLI interface
  Tests can be executed quickly from terminal, which is ideal for automation and CI/CD pipelines.
- Rich built-in metrics
  K6 provides response time percentiles, throughput, error rates, and more.
- Safe test target
  K6 provides an official public website for ethical testing.
- Active community & extensive documentation

Test Environment Setup

1. Hardware and Software
   - This network performance testing is being done using the latest stable version of K6 software running on Ubuntu machine with CPU of 12 cores and 16GB RAM
2. Network Conditions
   - Mobile networks with average speeds of 50Mbps
3. Target Web Application
   - The target web application link is https://test.k6.io, which is an official K6 demo site for performance testing purposes, with features such as lightweight HTML, realistic authentication, and also various testable endpoints

---

Performance Hypothesis

1. Load Test

The objective of this load test is to evaluate how the system performs under expected and sustained user load, ensuring that it can handle typical traffic without degradation. It is hypothesized that the system will maintain low response times, high throughput, and minimal errors while supporting up to 500 concurrent users. Specifically, the p95 response time is expected to remain below 800ms, and the error rate should stay near zero. Throughput should scale proportionally with the number of virtual users until the system reaches its practical capacity and without causing service interruptions

2. Stress Test

The system is expected to maintain acceptable performance as the load gradually increases up to 500 virtual users. Within this range, the average response time should stay below 800 ms, the p95 latency should remain under one second, and the error rate is not expected to exceed 1%. Once the load goes beyond 500 and approaches 1000 virtual users, the system is likely to show signs of stress. This may be reflected in slower response times, higher latency percentiles, a drop in throughput, or an increase in failed requests. After the load returns to zero, the system is expected to recover quickly and return to its normal performance levels within one to two minutes.

3. Spike Test

The purpose of this spike test is to evaluate how the system responds to sudden and extreme increases in traffic. The hypothesis is that the application should maintain acceptable performance under abrupt load spikes up to 1000 concurrent users, with average response times remaining below 500 ms, p95 latency under 1 second, and an error rate below 1. It is expected that throughput will increase sharply during the spikes but will plateau once the system reaches its maximum processing capacity. The test also anticipates that the system will recover quickly once the load returns to baseline, with no lasting performance degradation or persistent errors. This evaluation aims to identify potential bottlenecks under extreme traffic conditions and confirm the system’s resilience to unexpected surges in user activity.

Expected Bottlenecks

Across all three test types, the following bottlenecks may appear:

1. Server-Side Bottlenecks
    - CPU saturation when too many requests are processed simultaneously
    - Memory exhaustion (OOM kills, swap usage)
    - Single-threaded or blocking operations causing queue buildup
    - Insufficient worker processes or threads
2. Application-Level Bottlenecks
    - Slow database queries
    - Lack of caching
    - Session or authentication bottlenecks
    - High latency external API calls
3. Infrastructure Bottlenecks
    - Rate-limited web servers
    - Reverse proxy choking (Nginx/Apache)
    - Load balancer unable to distribute sudden spikes
    - Limited bandwidth or network congestion

Results and Raw data interpretation

1. Load Test

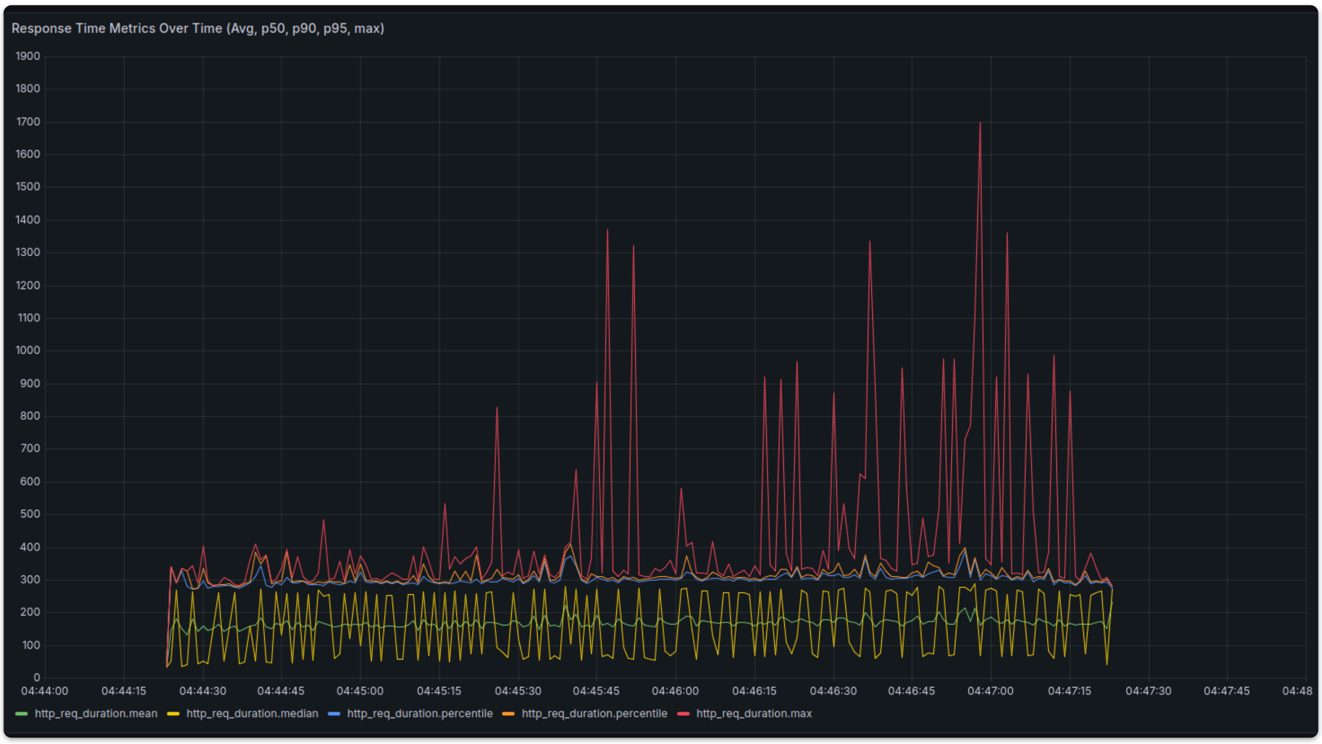
    The load test was designed to gradually increase virtual users (VUs) up to 500 over a 3-minute test window. The goal was to evaluate the application's performance under steady, increasing traffic and determine whether it meets the hypothesis of stable performance below 800ms (p95) with less than 1% error rate.

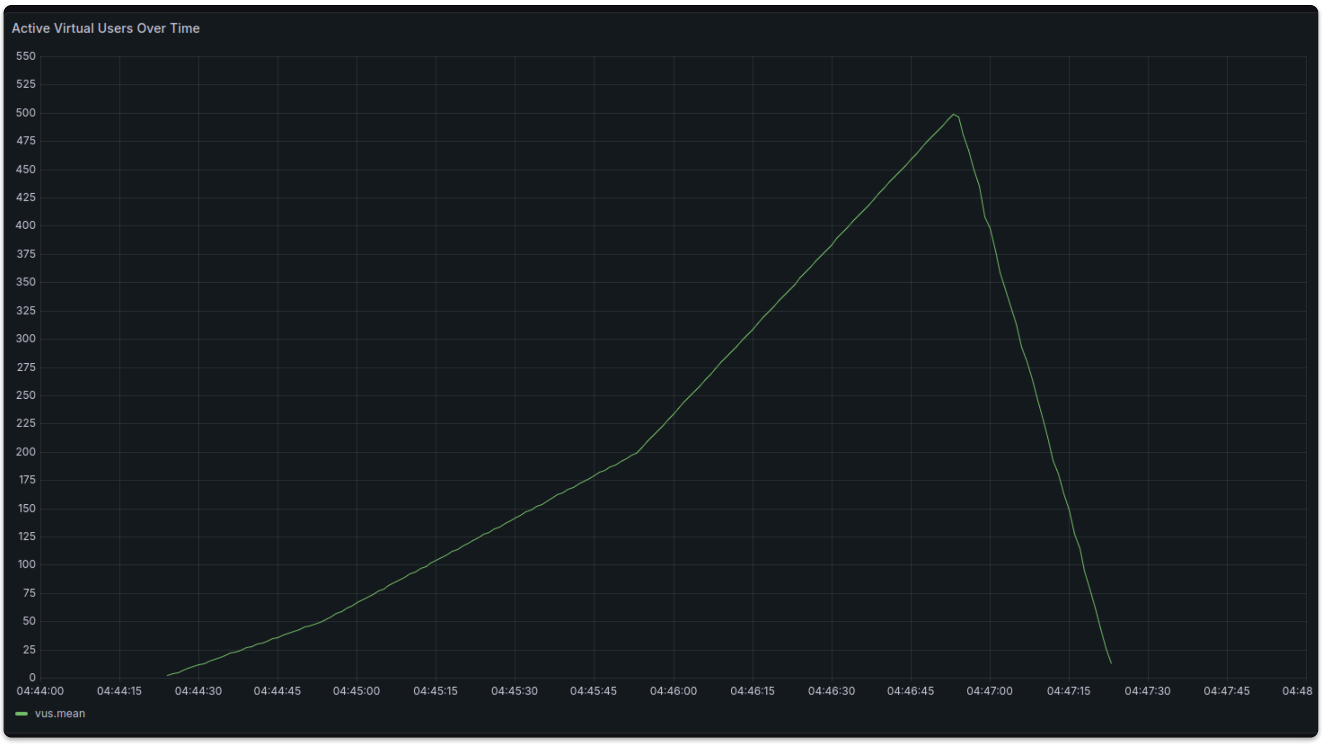
Summary of the Key Metrics:

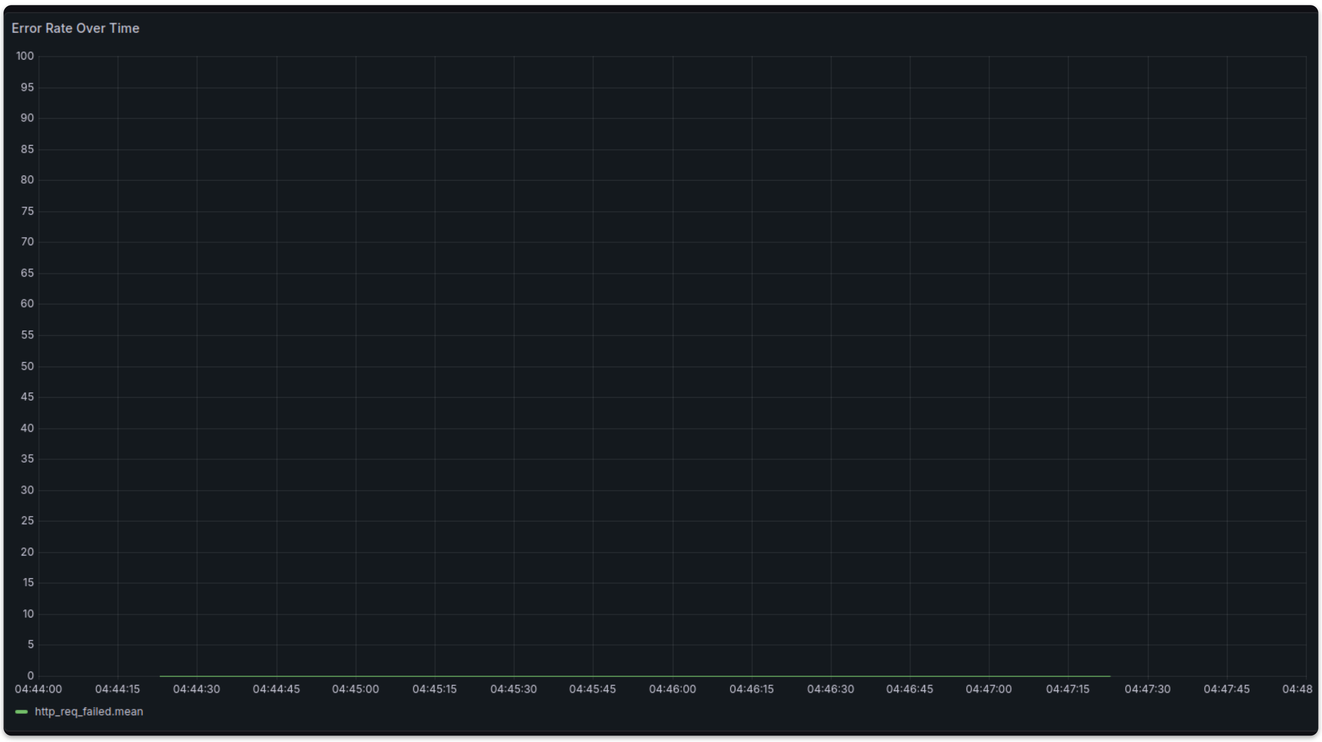
Metric	Value	Interpretation
Total Requests	54,508	High number of requests; provides a robust dataset for analysis

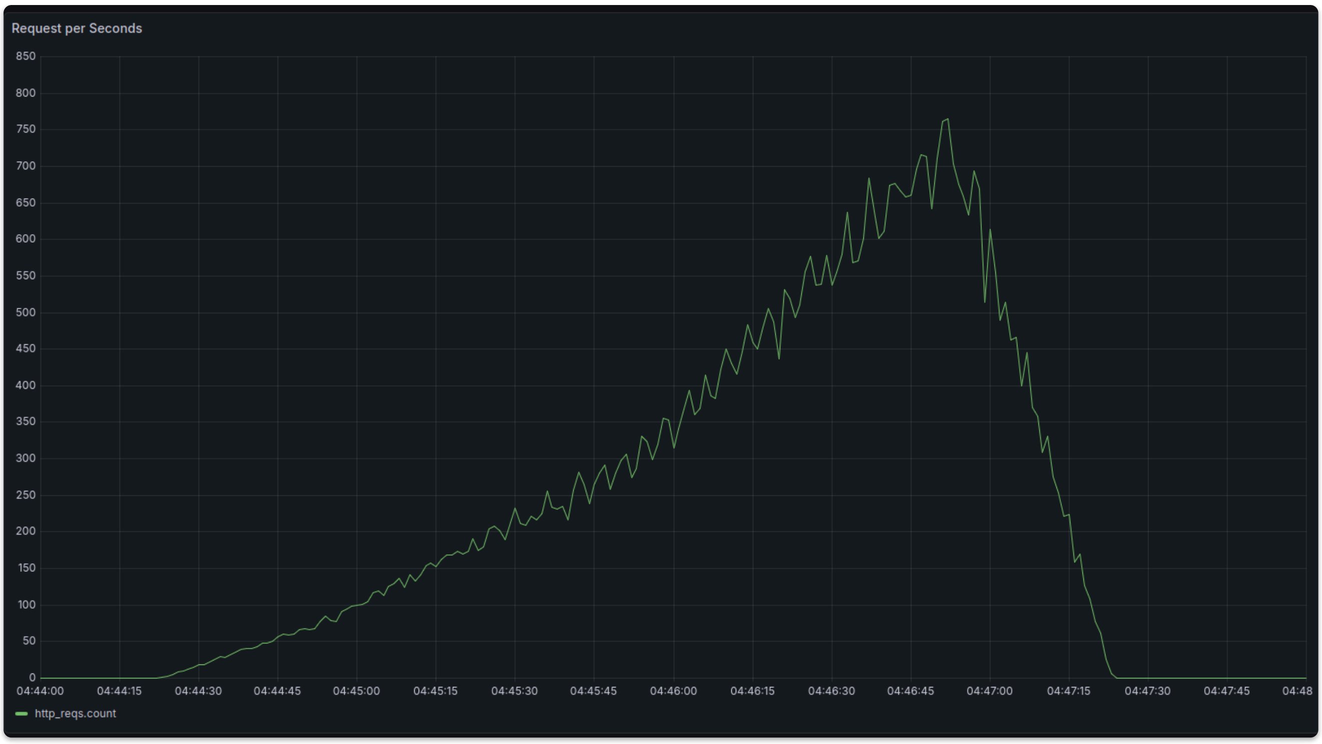
Metric	Value	Interpretation
Avg Response Time	172.95 ms	Overall latency is low, indicating the system handles expected load efficiently
Median (p50)	244.03 ms	Most requests complete quickly; typical user experience is fast
p90	307.81 ms	90% of requests are served under 308 ms; indicates strong performance under high load
p95	320.55 ms	Well below the 1-second threshold; system maintains SLA-compliant response times
Max Response Time	1.69 s	Occasional outliers likely due to temporary resource contention or longer backend processing
Error Rate	0.00%	No request failures; demonstrates excellent system stability under sustained load
Throughput	301 req/s	System efficiently processes a high volume of requests; throughput scales with user load
Max VUs	500	System supports up to 500 concurrent users without degradation

Time-series Graphs:




Active Virtual Users Over Time
550
525
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
04:44:00
04:44:15
04:44:30
04:44:45
04:45:00
04:45:15
04:45:30
04:45:45
04:46:00
04:46:15
04:46:30
04:46:45
04:47:00
04:47:15
04:47:30
04:47:45
04:48
vus.mean


Error Rate Over Time
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
04:44:00
04:44:15
04:44:30
04:44:45
04:45:00
04:45:15
04:45:30
04:45:45
04:46:00
04:46:15
04:46:30
04:46:45
04:47:00
04:47:15
04:47:30
04:47:45
04:48
http_req_failed.mean



2. Stress Test

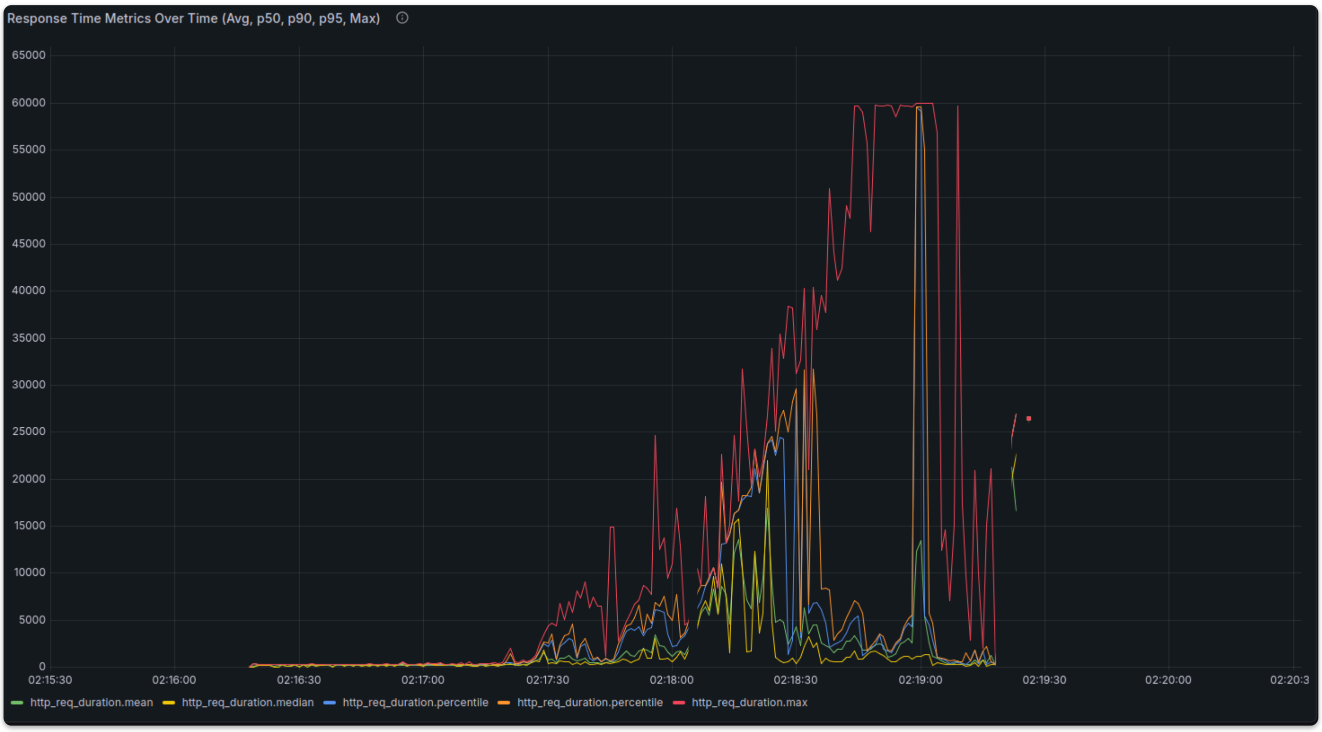
The stress test was carried out by gradually increasing the number of virtual users (VUs) from 50 up to 1000, with thresholds such as p95 latency and error rate

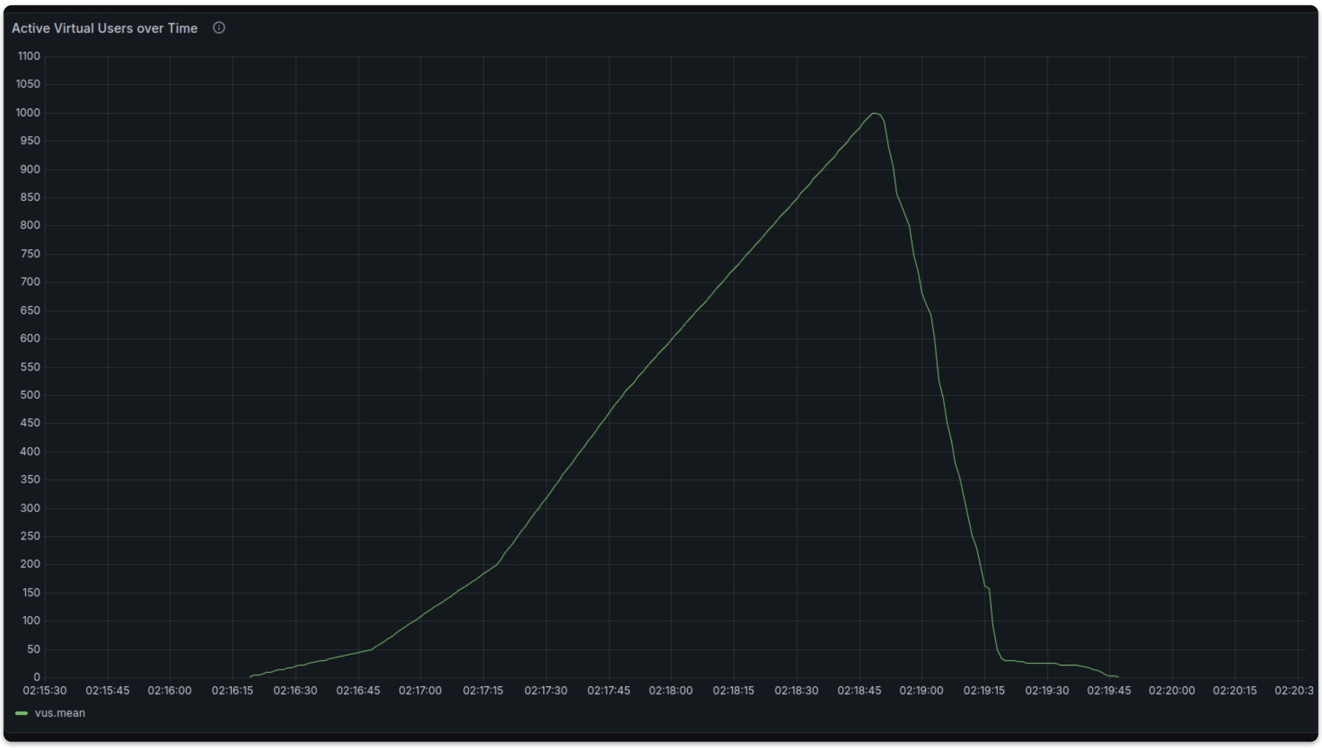
Summary of the Key Metrics:

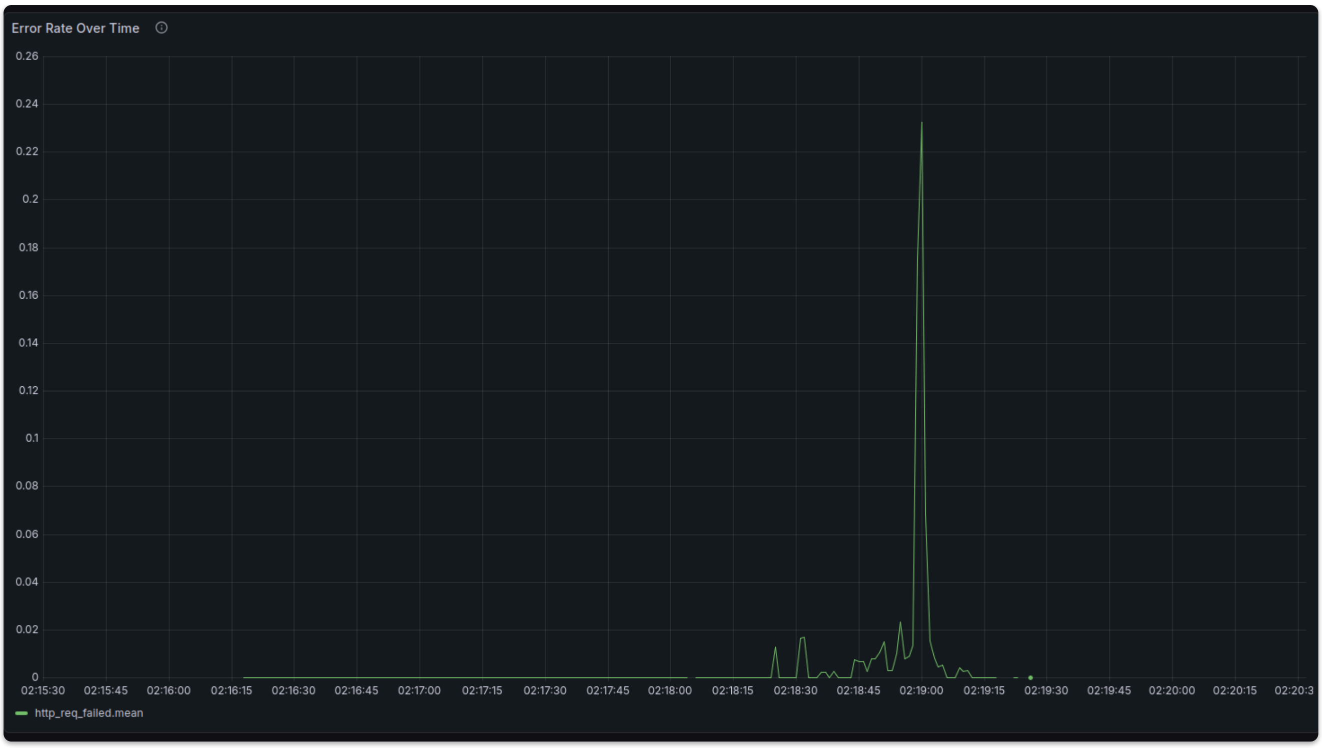
Metric	Value	Interpretation
Total Requests	34,851	High-volume test; sufficient data sample for meaningful analysis
Avg Response Time	1.49 s	Shows moderate latency under extreme load; slower than baseline but expected
Median (p50)	382.1 ms	Majority of requests completed quickly, indicating acceptable performance under mid-load
p90	2.61 s	Latency rises noticeably at high load; some requests experience delays
p95	4.5 s	Exceeds threshold (<1s); highlights performance degradation under stress
Max Response Time	60 s	Extreme outliers under peak load, indicating server saturation

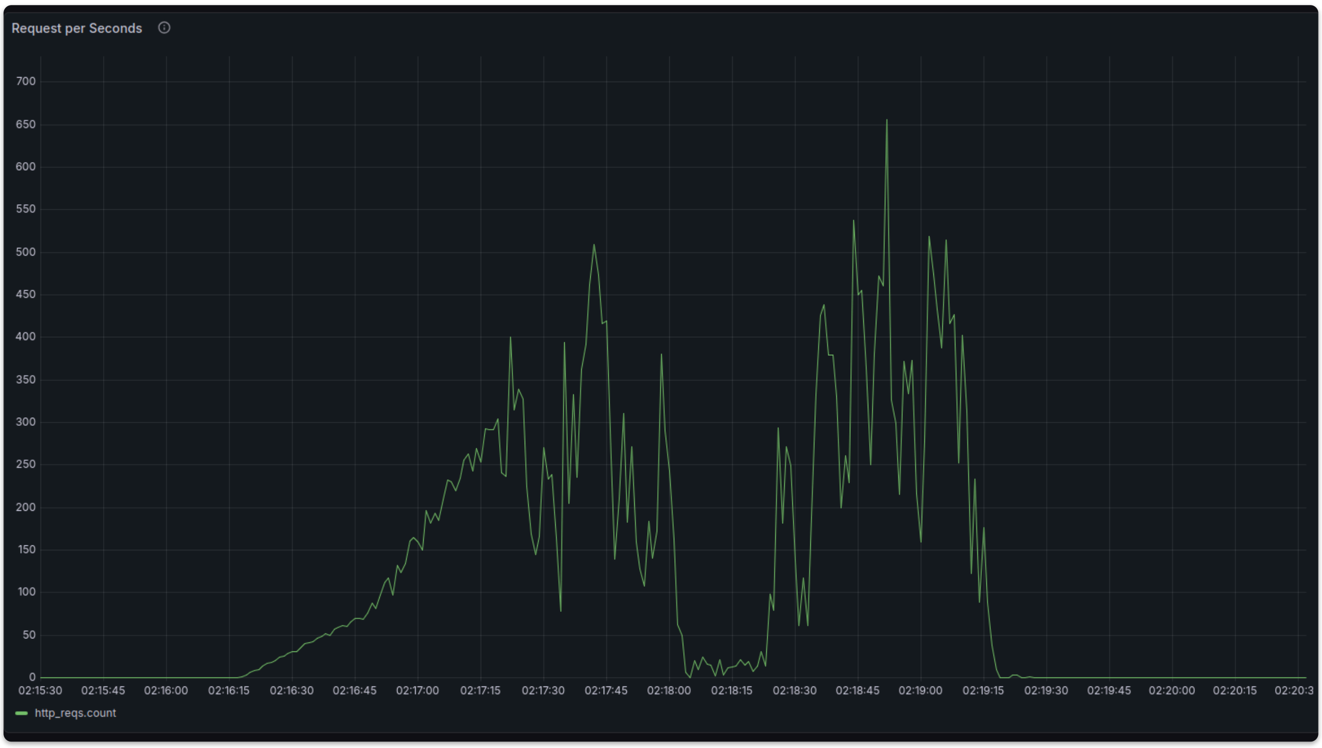
Metric	Value	Interpretation
Error Rate	0.48%	Low error rate despite high load; system remained mostly stable
Throughput	166 req/s	Throughput plateaued at high VUs, showing system reached capacity
Max VUs	1000	The system handled up to 500 VUs without critical issues. However, at 1000 VUs for 1-minute period, clear signs of overload appeared

Time-series Graphs :









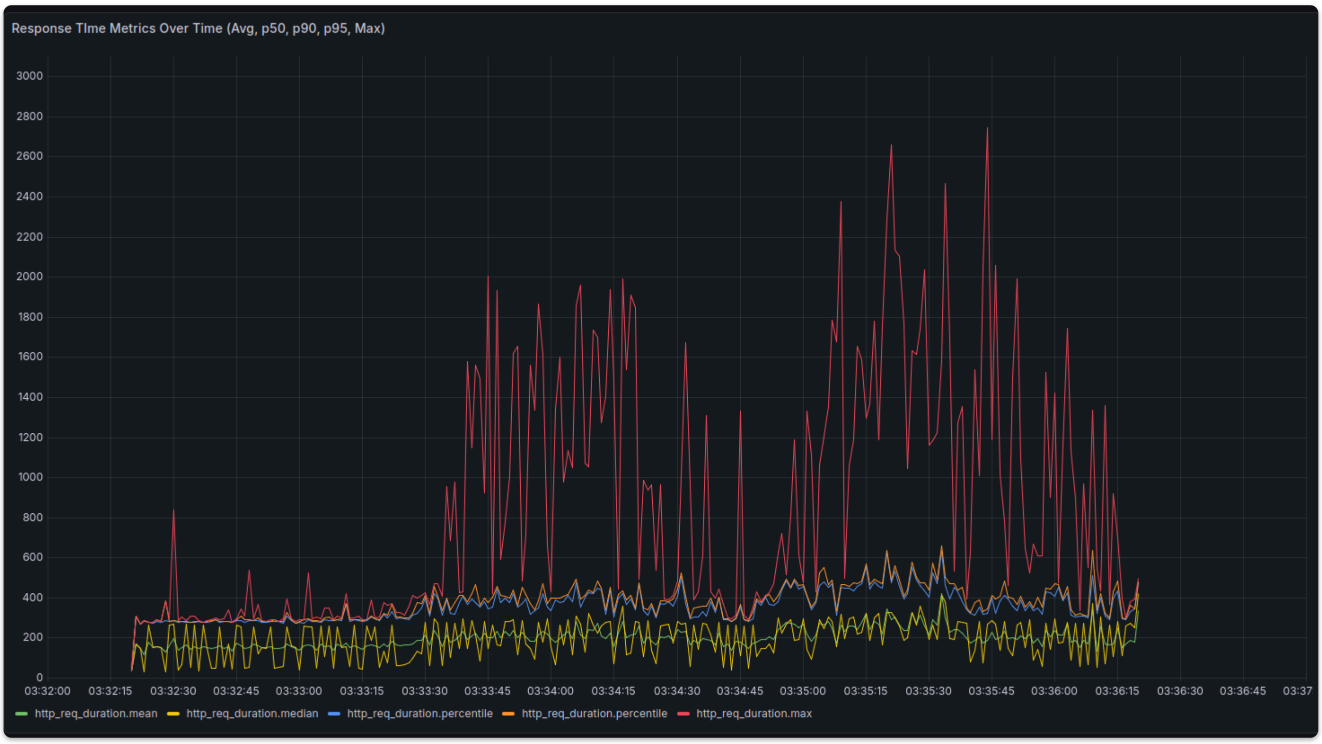
3. Spike Test

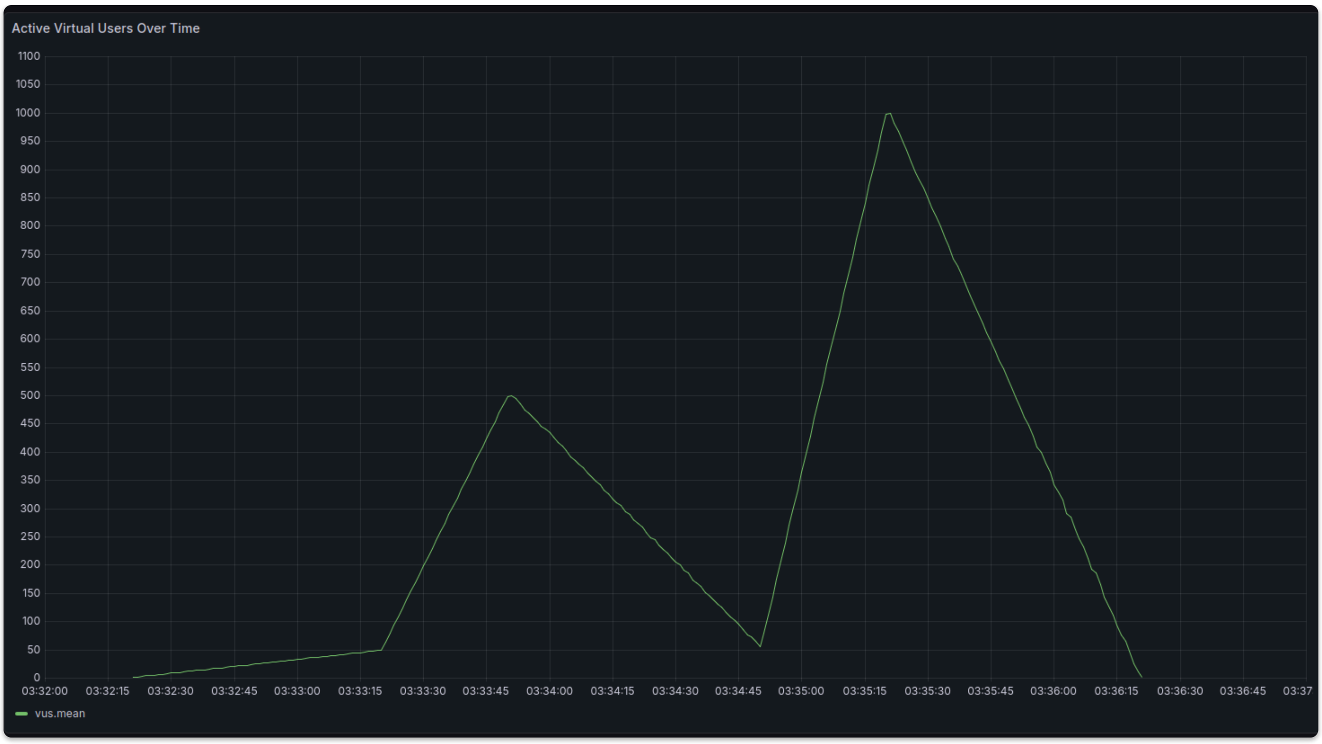
The spike test was conducted over 5 stages with a maximum of 1000 virtual users (VUs), gradually ramping up and then dropping to baseline to simulate traffic spikes.

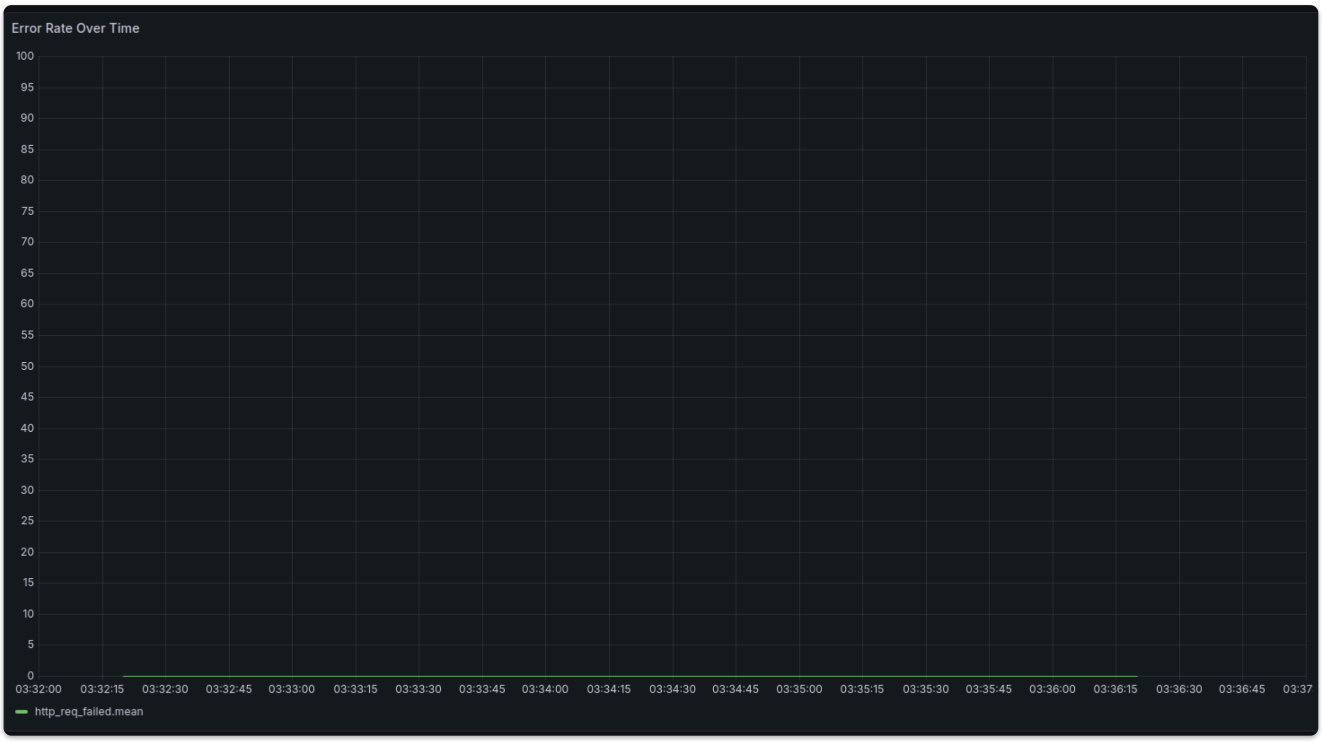
Summary of the key metrics :

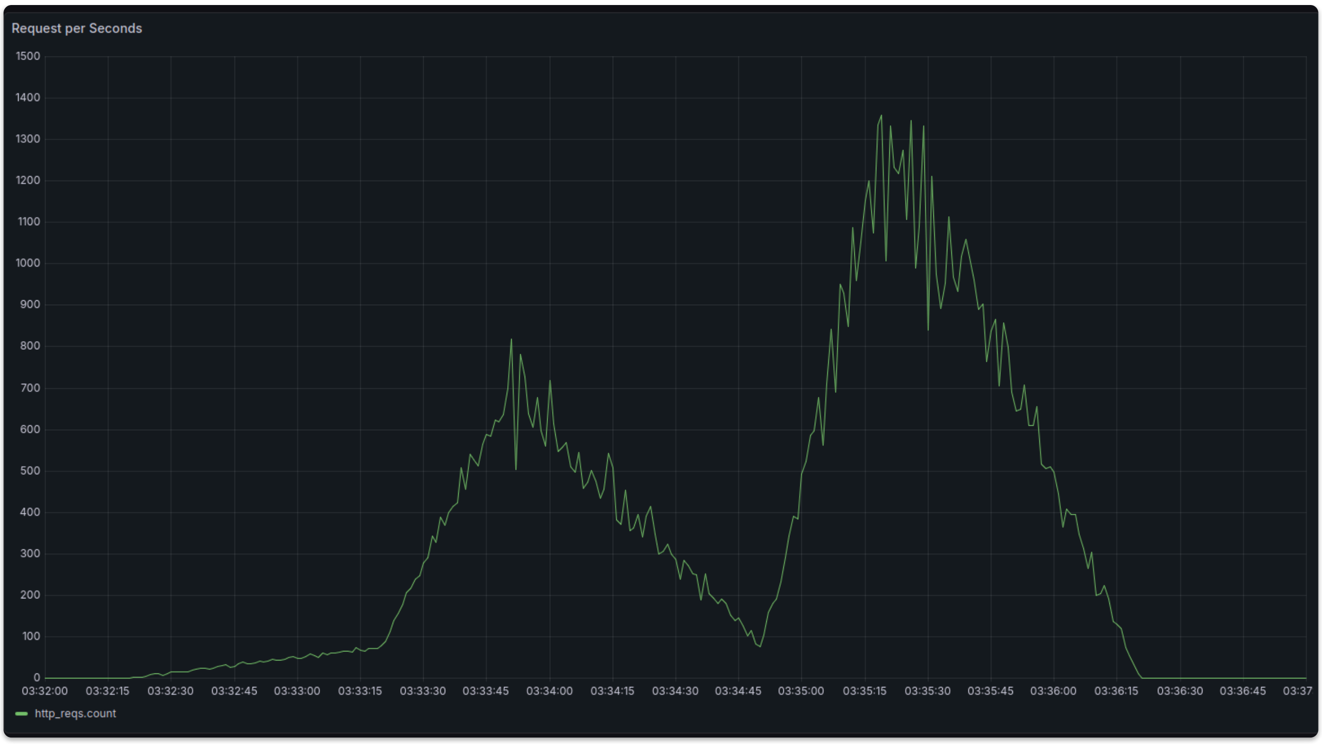
Metric	Value	Interpretation
Total Requests	98,892	High-volume test; sufficient data for meaningful spike analysis
Avg Response Time	229.02 ms	Low average latency; indicates system handles sudden spikes efficiently
Median (p50)	265.81 ms	Majority of requests complete quickly, showing good performance for most users
p90	421.71 ms	Latency increases slightly under peak load but remains acceptable
p95	466.32 ms	Below the threshold (1s); system maintains performance even during sudden spikes
Max Response Time	2.74 s	Outliers occur during peak VU spikes; temporary resource contention is evident
Error Rate	0.00%	No requests failed; demonstrates excellent stability under abrupt load
Throughput	409 req/s	System maintained high throughput, handling sudden spikes without degradation
Max VUs	1000	System successfully supported a spike of 1000 virtual users without degradation in service quality

Time-series Graphs:




Active Virtual Users Over Time
1100
1050
1000
950
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
03:32:00
03:32:15
03:32:30
03:32:45
03:33:00
03:33:15
03:33:30
03:33:45
03:34:00
03:34:15
03:34:30
03:34:45
03:35:00
03:35:15
03:35:30
03:35:45
03:36:00
03:36:15
03:36:30
03:36:45
03:37
vus.mean


Error Rate Over Time
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
03:32:00
03:32:15
03:32:30
03:32:45
03:33:00
03:33:15
03:33:30
03:33:45
03:34:00
03:34:15
03:34:30
03:34:45
03:35:00
03:35:15
03:35:30
03:35:45
03:36:00
03:36:15
03:36:30
03:36:45
03:37
http_req_failed.mean



Interpretation of Results and Identified Bottlenecks

1. Load Test

Interpretation of Results:

- The load test demonstrates that the system performs reliably under expected and sustained user load, with up to 500 concurrent virtual users (VUs. The average response time of 172.95 ms and median of 244.03 ms indicate that most users experience fast, responsive interactions. The p90 and p95 response times (≈308 ms and 321 ms remain well below the 1-second threshold, showing that the system maintains consistent performance even under high concurrency
- Throughput of 301 requests per second reflects the system's ability to efficiently handle a significant volume of requests, while the 0% error rate confirms excellent stability and error handling. The maximum response time of 1.69 seconds represents occasional outliers caused by temporary backend or resource contention, but these are infrequent and do not impact the overall system reliability.
- Overall, the results indicate that the system can sustain expected user loads without performance degradation, making it suitable for production use under typical traffic conditions.

Identified bottlenecks:

Despite strong performance, the test revealed minor bottlenecks:

Latency Peaks

- The occasional maximum response time of 1.69 seconds indicates that some requests experience temporary delays , which is likely due to database queries, CPU, or I/O contention during peak VU activity.

Iteration Duration Variability

- Some iterations took longer than average (up to 2.74 seconds), showing slight variability in processing time under load.

Throughput Plateau

- Throughput remained steady at around 301 requests/sec, suggesting the system has a practical maximum capacity under the current configuration and load pattern.

2. Stress Test

Interpretation of results:

- The stress test results show that the system performs well under low to moderate load but begins to experience noticeable performance degradation as the number of virtual users approaches its upper limit. The median response time of 382 ms indicates that most requests were served quickly, even when the system was under significant pressure. However, higher latency percentiles, such as p90 at 2.61 seconds and p95 at 4.5 seconds, reveal that a subset of requests experienced substantial delays, particularly when the system was handling the maximum load of 1000 virtual users.
- The maximum response time of 60 seconds represents extreme outliers, which are typical in stress testing scenarios where the system is pushed beyond its designed capacity. Despite these delays, the overall error rate remained very low at 0.48%, suggesting that the system largely maintained functionality and did not crash, even under extreme stress. Throughput plateaued at 166 requests per second, indicating that the system had reached its maximum processing capability.
- These results confirm that the system is stable and efficient under normal and moderate loads but exhibits predictable bottlenecks under

extreme stress conditions. The recovery to normal performance after the load was removed further demonstrates system resilience.

Identified Bottlenecks

Based on the stress test observations, several bottlenecks were identified:

Latency Increase at High Concurrency

- As the number of virtual users increased beyond 500, response times rose significantly. This suggests that server-side processing, database queries, or application threads may be reaching their maximum capacity.

Request Timeouts and Extreme Outliers

- The presence of very high maximum response times (up to 60 seconds) indicates that some requests were waiting in queues or being delayed due to resource contention.

Throughput Plateau

- Throughput stopped scaling proportionally with the number of virtual users. This indicates that the system had reached its maximum processing limit, and additional load no longer translated into higher request handling.

Minor Early Signs of Degradation

- Even before reaching peak load, the p90 and p95 latencies began increasing gradually. This early warning highlights potential bottlenecks in database access, external API calls, or thread pools that may need optimization.

3. Spike Test

Interpretation of results:

- The spike test demonstrates that the system is highly resilient under sudden, extreme increases in load. Across all stages, the majority of requests were served quickly, with an average response time of 229 ms and a median of 266 ms, indicating that typical users experience minimal delay even during traffic spikes. The p90 and p95 response times (422 ms and 466 ms, respectively) remained below the set threshold of 1 second, confirming that the system maintains acceptable performance for nearly all users.

- A small number of requests reached a maximum response time of 2.74 seconds, which reflects temporary resource contention at the peak of 1000 virtual users. Despite these occasional outliers, the system recorded zero request failures, highlighting robust stability and effective error handling. Throughput remained consistently high at 409 requests per second, demonstrating that the application can efficiently process a large number of simultaneous requests during abrupt spikes.
- Importantly, the system recovered immediately after each spike, with latency and throughput returning to baseline levels without intervention. This rapid recovery indicates strong resilience and the ability to handle sudden surges in traffic without lasting performance degradation.

Identified bottlenecks:

While the system handled the spike test effectively, the results reveal a few potential bottlenecks that could affect performance under even higher or more frequent spikes:

Latency Peaks at Maximum Load

- The occasional maximum response time of 2.74 seconds suggests temporary CPU, memory, or I/O contention.
- These peaks occurred only during the highest VU spikes and did not result in errors, but could affect user experience if spikes are sustained.

Throughput Limits

- Throughput plateaued at around 409 requests per second, indicating that the server has a practical upper limit for request processing under sudden load.
- Future scaling may require optimization of connection handling, thread pools, or backend resources to handle even higher spikes.

Iteration Duration Variability

- Some iterations took longer than average, reflecting that certain requests experienced slightly longer processing times under peak load.
- This is expected during spikes but points to potential areas for optimization, such as query performance or caching.

Recommendations for Improvement

Although the system performed strongly across all three tests, several areas can still be improved to enhance overall performance and resilience. The occasional high-latency outliers observed across the tests indicate that some backend operations or database queries may be slower than expected. Improving the efficiency of these endpoints through better query indexing, caching frequently accessed data, or reducing heavy input and output operations would help eliminate these response time spikes. Introducing server-side caching solutions such as Redis or Memcached can also reduce pressure on backend services during periods of high activity. The plateau in throughput seen during the stress and spike tests suggests that the system would benefit from greater scalability, which could be achieved by adding more server instances or upgrading existing resources. Stabilizing database performance through better connection pool tuning and offloading read operations to replica databases can further strengthen behavior under heavy load. Reducing latency caused by garbage collection or resource management pauses through runtime tuning may also yield noticeable improvements. Implementing rate limiting and graceful degradation mechanisms can protect the system during sudden traffic surges, while enhanced logging, monitoring, and alerting will ensure issues are detected and resolved more quickly.

Final Conclusion

Overall, the results from the load, stress, and spike tests show that the system is stable, responsive, and capable of handling a wide range of traffic conditions. Under normal and expected loads, performance remained consistently strong, with fast response times, zero errors, and smooth throughput, which confirms the system’s readiness for typical production use. When pushed beyond its designed capacity during stress testing, the system continued to operate reliably until it reached its saturation point, revealing its maximum performance limits without unexpected failures. The spike test also demonstrated the system’s resilience, showing that it can absorb sudden and intense bursts of traffic without crashing or experiencing significant drops in responsiveness. Taken together, these findings indicate that the current architecture is robust and well prepared for real-world use.

The few minor bottlenecks identified, mainly related to occasional latency spikes and limits in throughput, can be addressed through focused optimizations and better scaling strategies. With these improvements, the system will be even more capable of supporting future growth and sudden increases in traffic.