Performance Testing Report

for SGKMS API on Server Infrastructure

Overview

This document presents the results of performance testing conducted on the SGKMS API to evaluate its scalability and stability under various user loads and payload sizes, with a focus on how server specifications impact system behavior. Since API performance is highly dependent on the system environment, including network conditions and server capabilities, testing was conducted across multiple server infrastructures with different hardware specifications. The evaluation measures key performance metrics such as Transactions Per Second (TPS), response times, and error rates, providing a comparative analysis of how different processor configurations influence overall system performance.

Executive Summary

1. Objectives: The primary objective of this performance testing is to evaluate how server specifications, particularly processor capacity, impact SGKMS API performance under varying user loads and payload sizes.
2. Methodology: Performance tests were conducted using custom JavaScript scripts simulating 10 to 10,000 concurrent users with payloads from 0.1 KB to 100 KB. Metrics such as TPS, latency, and error rates were measured across different server infrastructures to compare processor performance.
3. Key Findings:
   - Architecture 1 performs better in terms of TPS, latency, and error rates. Architecture 2 struggles under high loads, with higher latency spikes and failure rates.
   - Performance degradations are more pronounced in Architecture 2, making it less suitable for workloads requiring consistent low-latency responses.
   - Latency spikes beyond 2000-2500 users indicate that both architectures reach their computational limits at high concurrency, requiring optimization or scaling strategies.
4. Conclusion: Processor capacity significantly impacts API performance. Architecture 1 is the preferred infrastructure for handling high-load cryptographic workloads with better stability and efficiency.

Objectives

The performance testing aims to:

1. Measure Transactions Per Second (TPS): Determine the number of successful transactions per second the API can process across different server environments and different loads.
2. Analyze Response Time: Evaluate how API latency changes under increasing concurrency levels.
3. Identify Performance Bottlenecks: Detect where and when performance degradation occurs.

4. Compare System Performance Across Different Architectures: Assess the impact of different processor configurations on API efficiency and stability.

Scope of Testing

Application Under Test

The performance testing was conducted on SGKMS API, focusing on encryption, data sealing, tokenization, and digital signature operations under varying server environments.

Test Environment

Architecture 1

- Server xFusion 1288H V6 (CCEV, LCEV)
  - CPU: Intel Xeon Silver 4314 @ 2.40GHz (32 Threads, 16 Cores, 1 Socket)
  - Memory: 117 GiB RAM (114 GiB available), 4.0 GiB Swap
  - OS: Fedora 41 (Workstation Edition)
  - Network: Download 48.30 Mbps, Upload 49.10 Mbps, Ping 58.982 ms
- PC ThinkCentre M70t (Audit)
  - CPU: Intel Core i5-10400 @ 2.90GHz (12 Threads, 6 Cores, 1 Socket)
  - Memory: 30 GiB RAM (26 GiB available), 8.0 GiB Swap
  - OS: Fedora 41 (Workstation Edition)
  - Network: Download 48.54 Mbps, Upload 49.49 Mbps, Ping 20.359 ms

Architecture 2

- Server xFusion 1288H V6 (Audit)
  - CPU: Intel Xeon Silver 4309Y @ 2.80GHz (16 Threads, 8 Cores, 1 Socket)
  - Memory: 123 GiB RAM (121 GiB available), 4.0 GiB Swap
  - OS: Fedora 33 (Server Edition)
  - Network: Download 172.45 Mbps, Upload 166.67 Mbps, Ping 12.524 ms
- Server DELL PowerEdge R340 (DRC)
  - CPU: Intel Xeon E-2276G @ 3.80GHz (12 Threads, 6 Cores, 1 Socket)
  - Memory: 62 GiB RAM (59 GiB available), 4.0 GiB Swap
  - OS: Fedora 33 (Server Edition)
  - Network: Download 174.63 Mbps, Upload 180.99 Mbps, Ping 12.632 ms
- Server HPE ProLiant DL20 Gen10 (CCEV)
  - CPU: Intel Xeon E-2276G @ 3.80GHz (6 Threads, 6 Cores, 1 Socket)
  - Memory: 7.5 GiB RAM (5.5 GiB available), 3.7 GiB Swap
  - OS: Fedora 33 (Server Edition)
  - Network: Download 163.85 Mbps, Upload 181.76 Mbps, Ping 12.085 ms
- Server HPE ProLiant DL20 Gen10 (LCEV)
  - CPU: Intel Xeon E-2276G @ 3.80GHz (6 Threads, 6 Cores, 1 Socket)

    - Memory: 7.5 GiB RAM (5.5 GiB available), 3.7 GiB Swap
    - OS: Fedora 33 (Server Edition)
    - Network: Download 89.33 Mbps, Upload 93.97 Mbps, Ping 12.925 ms

API Operations Tested

- Encryption:
    - DataEncryptionwithAES
    - DataEncryptionwithRSA4096andSessionKey
- Data Sealing:
    - SealDatawithAES
    - SealDatawithRSA2048
- Digital Signature:
    - DigitalSignatureswithECDSA
    - DigitalSignatureswithRSA3072
- Tokenization

Testing Methodology

Load Simulation

The tests simulated different user loads ranging from **10 to 10,000 concurrent users** and varied **payload sizes between 0.1 KB and 100 KB** to evaluate system behavior under diverse workloads.

Testing Tools

- Custom JavaScript scripts were used to automate API calls and measure response characteristics under different load levels.
- Server monitoring tools were utilized to track CPU utilization, memory usage, and network performance during test execution.

Metrics Captured

The following key performance indicators were measured to analyze system performance:
- **Transactions Per Second (TPS)**: The number of successful API transactions processed per second.
- **Average Latency (ms)**: The mean response time of successful API requests under different load conditions.
- **Maximum Latency (ms)**: The longest recorded response time among all processed requests, indicating worst-case performance.
- **Error Rate (%)**: The percentage of failed API requests out of the total sent.

Test Results

Due to the extensive nature of the dataset, the detailed results of the performance test, including latency, TPS, and error rates under various user loads and payload sizes, are available in the CSV files linked below.

- Download Test Results - Architecture 1 (CSV)
- Download Test Results - Architecture 2 (CSV)

Analysis and Findings

Performance Metrics

- **Transactions Per Second (TPS):**
    - Architecture 1 achieves a higher maximum TPS (547.7 TPS) compared to Architecture 2 (370.05 TPS).
    - The lowest TPS is 20.54 TPS for Architecture 1 (Tokenization, 2800 users) and 4.96 TPS for Architecture 2 (Encrypt RSA-4096, 3000 users).
    - This suggests that Architecture 1 handles higher loads more efficiently, particularly for cryptographic operations.
- **Latency:**
    - Architecture 1 has a lower average latency (3018.06 ms) compared to Architecture 2 (5496.75 ms), indicating faster response times.
    - However, both architectures exhibit extreme latency spikes under heavy load, with max latency reaching 129,635.2 ms (Tokenization, 2800 users) in Architecture 1 and 508,901.9 ms (Encrypt RSA-4096, 3000 users) in Architecture 2.
    - Architecture 2 shows higher max latency, indicating that it struggles more under peak loads.

Stability & Error Rate

- **Error Rate:**
    - Architecture 1 maintains a much lower average error rate (0.0099%) compared to Architecture 2 (0.2477%), indicating better stability.
    - The maximum error rate observed in Architecture 1 was 2.71% (Tokenization, 2800 users), while in Architecture 2, the maximum error rate reached 15.67% (Encrypt RSA-4096, 3000 users). This indicates that Architecture 2 is more susceptible to failures under high demand compared to Architecture 1.
- **Maximum Concurrent Users Before Failure:**
    - Architecture 1 remains stable up to 2500 concurrent users, while Architecture 2 starts failing at 2000 users, showing that Architecture 1 scales better.

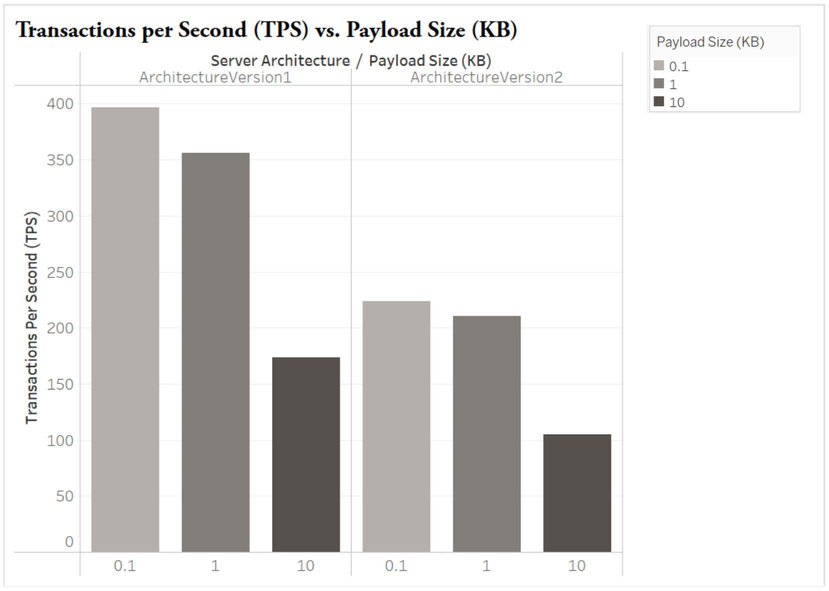
Performance Bottlenecks & Scalability

- Increasing user loads and payload sizes significantly impact performance, with latency spikes and error rates increasing sharply beyond 2000 users (Architecture 2) and 2500 users (Architecture 1).

- Architecture 1 consistently outperforms Architecture 2 in handling high loads with better stability.

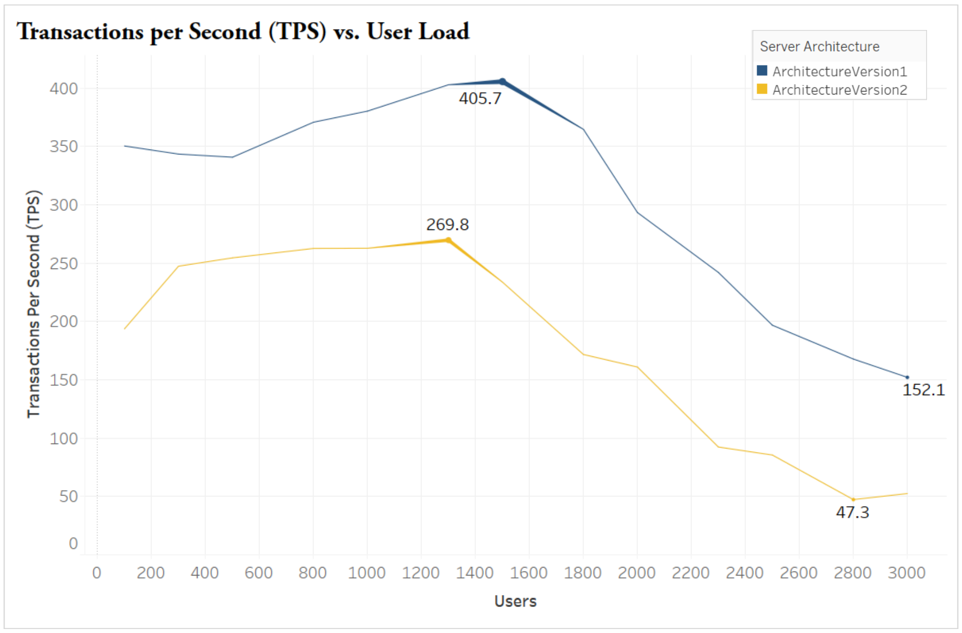
Visualizations

- Transactions per Second (TPS) vs. Payload Size (KB): This chart shows how TPS changes as payload size increases. Larger payloads tend to reduce the TPS.



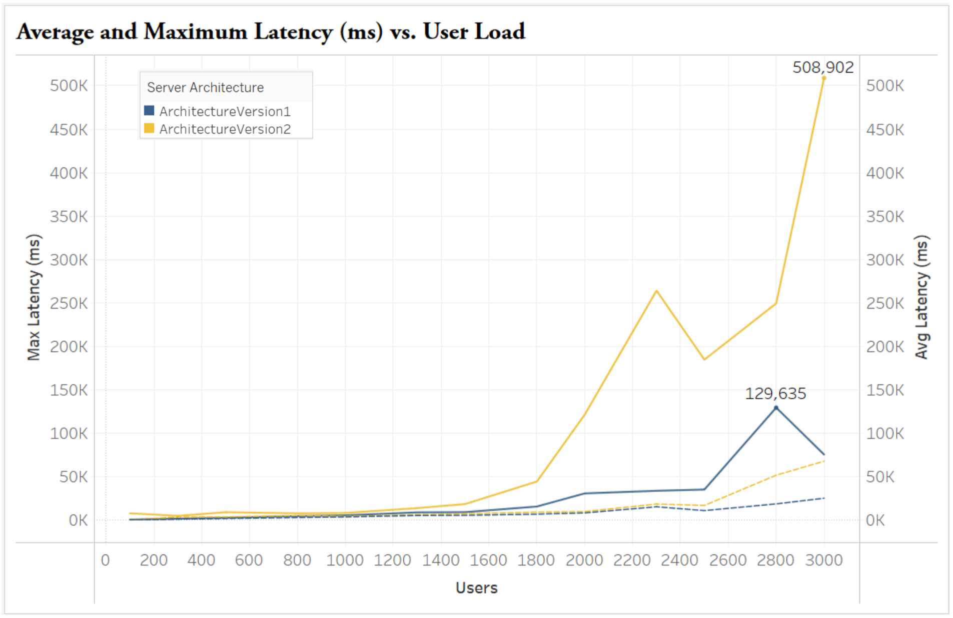
View interactively: Click Here

- Transactions per Second (TPS) vs. User Load: This graph illustrates the drop in TPS as user load increases, highlighting the system's limitations under higher concurrency.



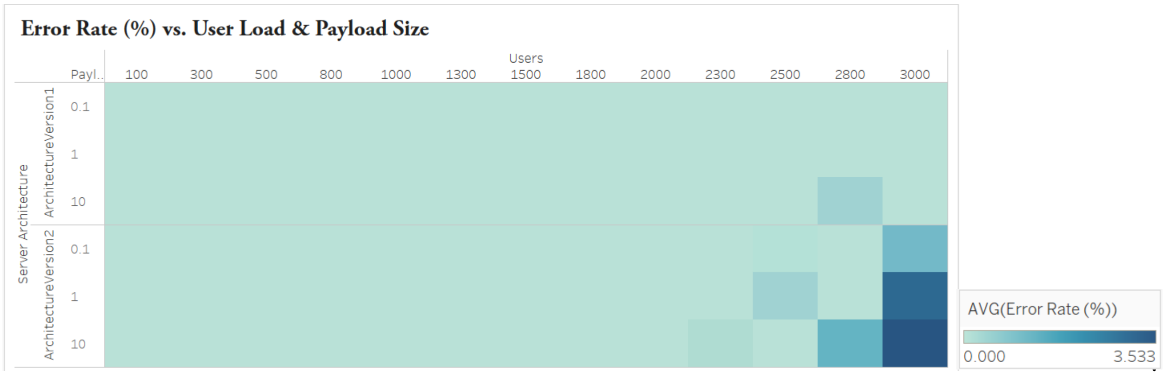
View interactively: Click Here

- Average and Maximum Latency (ms) vs. User Load: Latency spikes as the number of users increases, showing the system's response time under load.



View interactively: Click Here

- Error Rate (%) vs. User Load & Payload Size: This chart demonstrates how error rates rise with higher user load and larger payload sizes, indicating system stress points.



View interactively: Click Here

Conclusion

This performance testing report provides a comprehensive assessment of SGKMS API performance, ensuring that it meets the expected operational requirements under varying workloads. The results demonstrate that **server specifications,** particularly **processor capacity, significantly impact API performance.** Architecture 1 offers **better TPS, lower latency, and higher stability** compared to Architecture 2, making it the preferred option for high-load cryptographic operations. These insights will help in optimizing the API for better scalability and efficiency.