Load Testing & Benchmarking Report: Sarvam Transliteration API

Date: June 5, 2024
Author: Divyam Talwar

1. Executive Summary

This report details the findings of a performance and load testing evaluation conducted on the Sarvam Transliteration API. Using a series of controlled tests, we measured the API's latency, throughput, and reliability under increasing load. The API demonstrated excellent performance and stability under low to medium load (1-10 concurrent users). A performance degradation threshold was identified when scaling to 25 concurrent users, characterized by a significant increase in response times and the emergence of errors. Language-specific analysis revealed minor, consistent latency variations, with languages like Hindi performing faster than others like Bengali and Telugu.

2. Objective & Scope

The primary objective was to evaluate the **performance, scalability, and reliability** of the Sarvam Transliteration API.

- **In Scope:**
    - Testing the /v1/translate/transliterate endpoint.
    - Simulating traffic from 1 to 25 concurrent users.
    - Measuring key metrics: RPS, latency percentiles (p50, p75, p95), and error rate.
    - Analyzing performance differences across 8 different Indian languages.
- **Out of Scope:**
    - Testing other Sarvam API endpoints.
    - Server-side infrastructure monitoring.

3. Methodology

- **Load Generation Tool**: Locust (v2.x)
- **Test Environment**: Headless execution via a shell script for automation.
- **Test Data**: A CSV file containing sample text for 8 languages (hi, ta, bn, te, gu, mr, kn, pa).
- **Load Profile**: A load sweep was conducted using the configurations below.

Configuration	Concurrency (Users)	Spawn Rate	Run Time
1	1	1	1m
2	5	2	1m
3	10	2	3m
4	25	4	5m

4. Results & Analysis

The API's performance scaled effectively until the final stress test. Below is a summary of the aggregated results.

Table 1: Key Performance Indicators (KPI) Summary

Metric	Config 1 (1 User)	Config 2 (5 Users)	Config 3 (10 Users)	Config 4 (25 Users)
RPS (Avg)	0.85	4.21	8.35	19.55
Error Rate	0%	0%	0.2%	**1.5%**
Avg Latency (ms)	215	225	240	**380**
p95 Latency (ms)	280	310	355	**650**

4.1. Performance Under Load

The throughput (Requests Per Second) scaled almost linearly with the number of users up to the 10-user mark. The average and percentile latencies remained stable and low, indicating the system was comfortably handling the load.

A clear performance threshold was crossed between the 10-user and 25-user tests. At 25 concurrent users:

- **Latency Spike:** The p95 latency (the experience of the 95th percentile user) nearly doubled from 355ms to 650ms. This indicates users would perceive a noticeable slowdown.
- **Error Rate Increase:** The error rate jumped from a negligible 0.2% to 1.5%. While still low, this signals that the server is beginning to struggle, potentially queueing or dropping requests.

4.2. Language-Specific Latency

Analysis of the custom metrics collected during the 25-user stress test revealed consistent latency differences between languages.

Table 2: Language-wise p95 Latency (at 25 Users)

Language	p95 Latency (ms)	Observation
Hindi (hi)	580	Consistently among the fastest.
Tamil (ta)	595	Fast performance.
Marathi (mr)	610	Average performance.
Bengali (bn)	710	Consistently ~20% slower than Hindi.
Telugu (te)	725	Consistently the highest latency.

This suggests that the underlying transliteration model may have varying computational complexity or efficiency based on the source language script.

5. Conclusions & Recommendations

Conclusion: The Sarvam Transliteration API is highly performant and reliable for applications with low to moderate concurrent traffic (up to ~10-15 simultaneous users or ~10 RPS). The system exhibits signs of stress and potential bottlenecks when traffic exceeds 20 RPS, making it a key threshold to consider for scaling.

Recommendations:

1. **Establish a Service Level Objective (SLO):** Define an official p95 latency target (e.g., under 500ms). The current tests show this SLO is met up to the 10-user configuration.
2. **Investigate the 25-User Bottleneck:** For services requiring higher throughput, a server-side investigation is recommended to identify the root cause of the latency spike and errors. This could involve profiling CPU, memory, I/O, or dependent services.
3. **Document Language Performance:** The observed latency differences between languages, while minor, should be documented for developers so they can set realistic expectations for end-user experience. No immediate action is required unless the variance is deemed unacceptable.