Performance Analysis – REST vs tRPC vs gRPC

Name: Muhammad Ibtihaj
Roll No: SP23-BAI-037
Course: Parallel and Distributed Computing
Assessment: Final Lab – Step 5 (Performance Analysis)

This report presents a performance comparison of REST, tRPC, gRPC (Unary), and gRPC-based microservices using experimental results from an image classification system. The evaluation focuses on response time, payload size, network efficiency, batching, and type safety.

gRPC Microservice Internal Latency

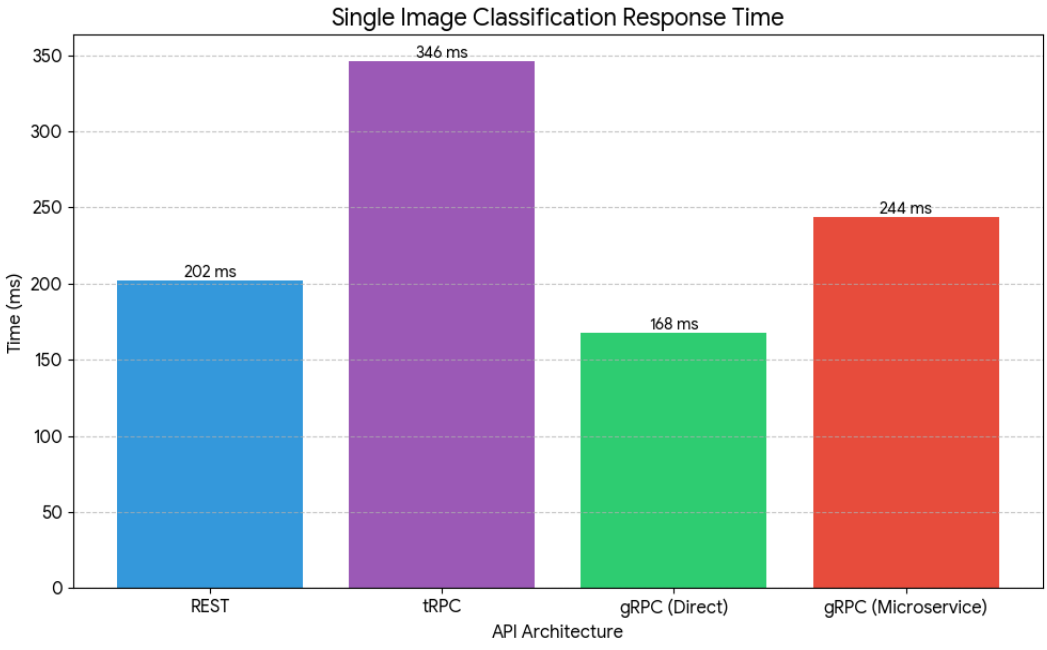
The internal latency between Service A (Gateway) and Service B (Model) was observed to range between **139 ms and 184 ms**, with an average of approximately **154 ms**. This indicates efficient binary communication within a microservice architecture.

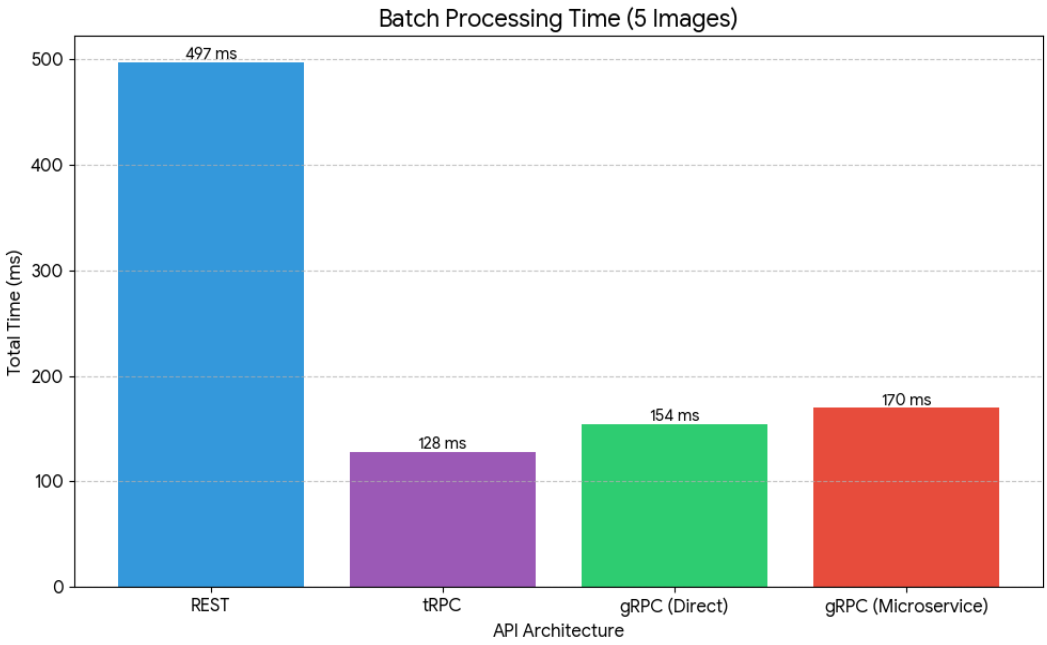
Performance Comparison Summary

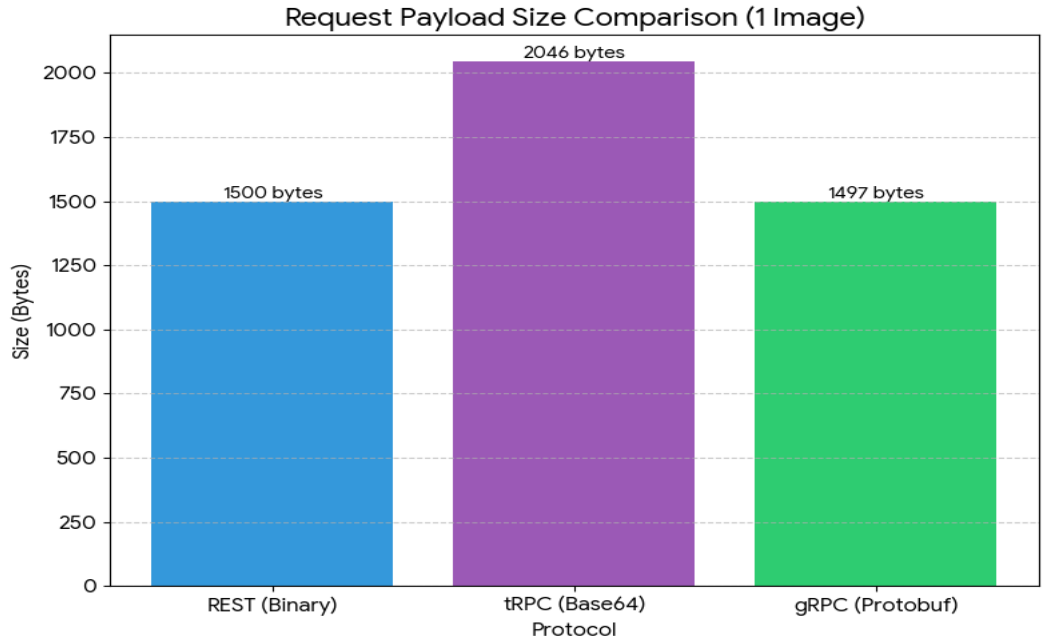
Technology	Response Time	Payload Size	Network Calls	Notes
REST (Single)	202 ms	38 bytes	1	Simple, higher overhead
REST (Batch 5)	497 ms	312 bytes	1	Slower batch processing
tRPC (Single)	346 ms	2046 bytes	1	Type-safe but heavier
tRPC (Batch 5)	128 ms	291 bytes	1	Very efficient batching
gRPC (Unary)	168 ms	1497 bytes	1	Fast & binary
gRPC (Microservice)	170 ms	1497 bytes	5	Low per-call latency

Visual Performance Comparison

The following charts visually compare payload size, batch processing time, and single-image response time across different API architectures.







Response Time: gRPC (Unary) shows the lowest response time for single image classification due to efficient protobuf serialization. REST performs reasonably but degrades significantly during batch operations. tRPC single requests are slower, however batching dramatically improves performance.

Payload Size: REST responses are the smallest due to minimal JSON metadata. tRPC introduces higher overhead because of Base64 encoding. gRPC maintains a balanced payload size using compact binary encoding.

Batching & Network Efficiency: tRPC excels in batching by reducing multiple requests into a single network call. gRPC microservices introduce additional internal calls but maintain low latency.

Conclusion: gRPC is the most suitable choice for high-performance distributed and microservice-based systems. tRPC provides strong type safety and excellent batching performance, making it ideal for TypeScript-based applications. REST remains simple and widely supported but is less optimal for performance-critical workloads.