Performance Comparison Report: Linux Namespaces vs. Docker

Overview

This report compares the performance of the microservices architecture implemented using raw Linux network namespaces vs. the Docker containerized implementation.

Benchmark Parameters

- **Tool**: Apache Benchmark (ab)
- **Concurrency**: 50
- **Total Requests**: 1000
- **Endpoint**: `/api/products`

1. Key Metrics

Metric	Linux Namespaces	Docker (Compose)	Difference
Requests per Second (RPS)	**54.07**	29.21	**-46%** (Docker is slower)
Mean Latency (ms)	**924.70**	1711.55	**+85%** (Docker higher)
Median Latency (ms)	806	**805**	~0% (Similar)
Failed Requests	**0**	996 (Length Mismatch)	Significant
Reliability	100%	~0.4% (Non-2xx)	-

2. Detailed Performance Analysis

Linux Namespace Implementation

- **Throughput**: Significantly higher throughput at ~54 RPS.
- **Stability**: Very stable with zero failed requests and consistent latency.
- **Networking**: Benefitted from direct `veth` pairs and host-level routing with minimal abstraction.

Docker Implementation

- **Throughput**: Lower throughput at ~29 RPS.
- **Overhead**: The lower RPS and higher mean latency are likely due to the additional layers inherent in Docker (container runtime, `docker-proxy`, and Docker bridge networking).
- **Errors**: High number of "Length Mismatch" failures indicates that under high concurrency (50), some backend responses were truncated or served error pages (confirmed by 4 non-2xx responses).

3. Conclusions

Raw Linux namespaces offer **higher raw performance** and **lower latency** for networking-intensive tasks by avoiding the overhead of a container orchestration layer. However, Docker provides superior **portability** and **ease of management**, which usually outweighs the performance cost in modern development workflows.

Recommendations for Docker Optimization

1. **Reduce Overhead**: Use `network_mode: host` if extreme performance is needed (though this loses isolation).
2. **Buffering**: Use a production-grade WSGI server like `gunicorn` instead of the Flask development server (`Werkzeug`) to handle concurrency better.
3. **Resource Tuning**: Assign more CPU/Memory resources to containers in `docker-compose.yml` .