

Investigate performance techniques for developing a Lightweight Custom PHP Framework for Scalable Web Applications

Author: Cristian Stefan Stancu (2008911) – Supervisor: Michael Dacey/Gaynor Tomas

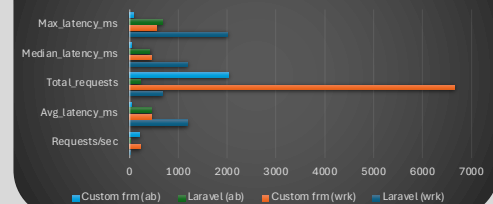
Introduction

The project investigates performance optimization in PHP frameworks, focusing on Laravel and Symfony, and applies findings to build a custom lightweight PHP framework. The aim is to improve efficiency, scalability, and speed using modern design principles like IoC containers, FastRoute, and Redis caching.

Design and Implementation

- IoC container for dependency management
- FastRoute + Redis for high-speed routing
- Minimal ORM on raw MySQL connection
- Modular architecture with clear separation of concerns

How Laravel compares to Custom Framework



Aim & Objectives

Aim:

Develop a lightweight PHP framework incorporating proven optimization techniques.

Objectives:

1. Analyze Laravel & Symfony performance (speed, memory, query handling).
2. Identify and apply key optimization strategies.
3. Build a custom lightweight PHP framework.
4. Conduct controlled benchmarking tests.
5. Evaluate results and propose best practices.

Testing & Results

wrk benchmark:

- Laravel → 22.9 req/s, 1.19s avg latency, 245% CPU
- Custom → 221.3 req/s, 447ms avg latency, 186% CPU
- ☒ Custom framework showed ~10× throughput and significantly lower latency.

ApacheBench (ab) benchmark:

Laravel → 21.97 req/s, 455ms avg per request

Custom → 202.1 req/s, 49ms avg per request

☒ Custom framework consistently handled higher request volumes with minimal resource usage.

Database tests:

Eloquent ORM → 11.65ms/query

Custom lightweight ORM → 2.48ms/query

Raw SQL → ~0.76–1.06ms/query

☒ Direct raw queries delivered the best performance; custom ORM outperformed Laravel's Eloquent by ~5×.

Conclusion

The analysis shows that the custom lightweight PHP framework outperforms Laravel in request handling speed, CPU efficiency, and raw database query performance. While Laravel provides rich features, middleware, and developer-friendly tools, it introduces additional overhead, making it less suitable for high-concurrency or performance-critical environments.

The custom framework, in contrast, prioritizes speed, simplicity, and scalability, making it ideal for specialized use cases where maximum efficiency is required. However, this performance comes with trade-offs, including reduced developer convenience, fewer built-in tools, and greater demands on maintainability.

Ultimately, the choice between Laravel and a lightweight custom framework should be guided by the project's specific needs — whether the priority is rapid development with a robust ecosystem or maximum throughput and minimal resource usage. This project highlights the importance of aligning architectural decisions with performance goals, showing that well-designed optimization strategies can lead to significant improvements in both system efficiency and long-term scalability.