Contribution ID: **93** Type: **Poster**

Performance Benchmarking: one of the System Design's pillars

In the rapidly evolving landscape of web development, the performance of backend technologies is a critical factor influencing scalability, efficiency, and user experience. This research aims to present a comprehensive performance comparison of Node.js, Rust, Go, and Python —four prominent technologies widely adopted in web application development. Through a series of systematic tests, we evaluated each technology's capability to handle concurrent connections, process I/O operations, and manage CPU-intensive tasks within a web server context.

Our methodology involved setting up a standardized testing environment for each technology, focusing on key performance metrics such as requests per second (RPS), latency, and CPU/memory utilization under varying loads. The tests were designed to simulate real-world scenarios, including serving static content, executing database operations, and performing computational tasks.

Node.js, with its non-blocking I/O model, demonstrated excellent performance in handling I/O-bound operations, particularly in scenarios with high concurrency levels. However, its single-threaded nature posed limitations in CPU-bound tasks, despite the potential for scalability offered by its cluster module.

Rust, known for its speed and memory safety, excelled in CPU-intensive tests, showcasing its ability to leverage system resources efficiently. Its asynchronous runtime further enabled impressive handling of I/O-bound operations, positioning it as a robust choice for high-performance web applications.

Go's simplicity and built-in concurrency model, based on goroutines and channels, allowed it to perform remarkably well across all tests. It balanced CPU and I/O operations adeptly, making it a versatile option for a wide range of web applications.

Python, while not matching the performance of the other three technologies in raw throughput and latency, stood out for its developer productivity and vast ecosystem. Its asynchronous frameworks, such as asyncio, provided significant performance improvements for I/O-bound tasks, although it lagged in CPU-bound processing.

In conclusion, our research underscores the importance of choosing the right technology based on the specific needs of a web application. While Rust and Go offer superior performance for CPU-intensive and high-concurrency applications, Node.js remains a strong contender for I/O-bound scenarios. Python, with its ease of use and extensive libraries, is well-suited for projects where development speed is paramount. This benchmarking exercise serves as a guide for developers and architects in making informed decisions, balancing performance with other factors such as development efficiency, ecosystem maturity, and maintainability.

Primary author: KHOKHRIAKOV, Igor (DESY)

Presenter: KHOKHRIAKOV, Igor (DESY)

Session Classification: Poster