The basis of our analysis is based on three different server setups: one utilizing a process that handles connections, server_proc.c, another utilizing threads, server_thread.c, and a third that combines threads with an additional web cache layer , namely server_cached.c. A controlled environment ensures a fair comparison, and the client simulator is designed to simulate web requests under different conditions.

We have two parts: the evaluation without cache, and then the evaluation with cache. Initially, process- and thread-based servers were scrutinized under a large number of simulated web requests. The cache-enabled server then undergoes a similar test and initial run to populate the cache. Comparison metrics include response time and resource usage. Comparative research between processes and threads in server implementations has some trade-offs. Processes running in isolated memory spaces provide stability and security at the cost of increased overhead due to context switches and memory usage. Threads sharing memory within the same process provide a leaner alternative with lower overhead, which may translate into faster response times in high-concurrency scenarios.

As observed in the third server setup, the introduction of caching marks a significant performance enhancement. By storing frequently accessed content in memory, the server reduces the need for repetitive disk I/O operations, resulting in significantly faster response times and reduced CPU and memory footprint.