ABSTRAK

Backend Node.js proses tunggal memiliki keterbatasan karena *event loop* yang bersifat *single-threaded* hanya memanfaatkan satu inti CPU, sehingga rentan mengalami penurunan performa pada beban tinggi. Penelitian ini membandingkan dua arsitektur *backend* pada sistem deteksi penyakit urin U-Detect, yaitu arsitektur *baseline* (V1) berupa Node.js/Express proses tunggal, dan arsitektur usulan (V2) yang menerapkan NGINX sebagai *reverse proxy* dan PM2 sebagai *process manager* dengan distribusi beban ke beberapa proses paralel. Verifikasi fungsional dilakukan melalui pengujian *black-box* dan integrasi, sedangkan pengujian performa menggunakan k6 dengan pola *constant arrival rate* pada rentang 100 hingga 500 RPS, melalui *isolated repeat* 35 *run* per titik beban dengan metrik P50, P95, P99, *error rate*, dan *throughput* serta pemanfaatan CPU sebagai indikator saturasi. Hasil menunjukkan kedua arsitektur identik secara fungsional dan memiliki batas beban deterministik stabil yang sama, yaitu 250 RPS. Manfaat arsitektur usulan terletak pada kestabilan di atas batas tersebut. *Baseline* mengalami kegagalan *timeout* sejak 300 RPS, sedangkan arsitektur usulan baru pada 350 RPS. Pada 400 RPS, *baseline* hanya mempertahankan 46 persen *run* stabil dengan latensi median 227 ms, sementara arsitektur usulan 94 persen dengan 28 ms. Saturasi total bergeser dari 450 menjadi 500 RPS. Ekspektasi penelitian terpenuhi, dengan peningkatan kapasitas moderat dan manfaat utama berupa kestabilan serta prediktabilitas pada beban tinggi.

Kata kunci : NGINX, PM2, *reverse proxy*, Node.js, pengujian beban, kapasitas sistem

ABSTRACT

A single-process Node.js backend is limited because its single-threaded event loop uses only one CPU core, making it prone to degradation under high load. This study compares two backend architectures on the U-Detect urinary disease detection system: a baseline (V1) of a single-process Node.js/Express, and a proposed architecture (V2) applying NGINX as a reverse proxy and PM2 as a process manager with load distribution across parallel processes. Functional verification used black-box and integration testing; performance testing used k6 with a constant arrival rate across 100 to 500 RPS, through isolated repeats of 35 runs per load point, measuring P50, P95, P99, error rate, and throughput and CPU utilization as an indicator of saturation.. Both architectures proved functionally identical and shared the same deterministically stable load limit of 250 RPS. The benefit of the proposed architecture lies in stability beyond that limit. The baseline experienced timeout failures from 300 RPS, the proposal only from 350 RPS. At 400 RPS, the baseline retained only 46 percent stable runs with a median latency of 227 ms, while the proposal retained 94 percent at 28 ms. Saturation shifted from 450 to 500 RPS. The expectation is confirmed, with a modest capacity gain and a primary benefit in stability under high load.

Keywords : NGINX, PM2, *reverse proxy*, Node.js, *load testing*, *system capacity*