BAB 5. PENUTUP

5.1 Kesimpulan

Penelitian ini mengimplementasikan dan membandingkan dua arsitektur *backend* pada sistem U-Detect, yaitu arsitektur *baseline* (V1) berupa Node.js/Express proses tunggal per layanan, dan arsitektur usulan (V2) yang menerapkan NGINX sebagai *reverse proxy* dan PM2 sebagai *process manager*. Berdasarkan hasil pengujian dan analisis, dapat ditarik kesimpulan sebagai berikut.

Pertama, penerapan NGINX dan PM2 pada V2 tidak mengubah perilaku fungsional sistem. Seluruh kasus uji *black-box* dan integrasi menghasilkan respons yang identik pada kedua arsitektur, sehingga perbedaan performa yang terukur dapat dikaitkan pada perbedaan arsitektur *deployment*, bukan pada perbedaan logika aplikasi.

Kedua, pada beban di dalam kapasitas, kedua arsitektur menunjukkan performa yang setara. Pada titik pembanding 250 RPS, yang merupakan beban tertinggi yang deterministik stabil pada keduanya, latensi *median* V2 justru sedikit lebih tinggi dari V1 dengan selisih hanya 1 hingga 2 ms. Selisih ini konsisten dengan penambahan satu hop perantara oleh NGINX, dan berada jauh di bawah ambang yang dapat dirasakan pengguna, sehingga tidak signifikan secara operasional.

Ketiga, penerapan NGINX dan PM2 tidak memperluas batas kapasitas deterministik stabil sistem. Beban tertinggi yang dapat dilayani dengan seluruh 35 *run* stabil sama pada kedua arsitektur, yaitu 250 RPS. Temuan ini menunjukkan bahwa manfaat arsitektur usulan tidak terletak pada peningkatan kapasitas puncak.

Keempat, manfaat utama penerapan NGINX dan PM2 adalah peningkatan kestabilan dan prediktabilitas pada beban mendekati kapasitas. Pada 400 RPS, V1 kehilangan determinismenya dengan hanya 46 persen *run* stabil dan latensi *median* 228 ms, sementara V2 mempertahankan 94 persen run stabil dengan latensi *median* 28 ms, yaitu sekitar delapan kali lebih rendah. Peningkatan ini dapat dihubungkan dengan distribusi beban ke beberapa *instans* proses melalui PM2 dan penggandaan kapasitas *inferensi* melalui dua *worker machine learning* paralel.

Kelima, kedua arsitektur menunjukkan karakteristik kegagalan yang berbeda. V1 mulai mengalami kegagalan *timeout* sejak 300 RPS, sedangkan V2 pada beban yang sama hanya mengalami degradasi ringan tanpa *timeout* dan baru mengalami *timeout* pertamanya pada 350 RPS. Pada 450 RPS, V1 mengalami saturasi total pada seluruh *run* dengan *error rate* 18,73 persen, sementara V2 masih menyisakan sebagian *run* pada kondisi degradasi dengan *error rate* 7,12 persen. V2 baru mengalami saturasi total pada 500 RPS. Dengan demikian, penerapan NGINX dan PM2 menggeser seluruh ambang kegagalan satu tingkat beban lebih tinggi.

Keenam, analisis pemanfaatan sumber daya komputasi menjelaskan mekanisme di balik perbedaan tersebut. Pada V1, layanan *prediction* sebagai proses tunggal terbatas pada satu inti CPU dan berhenti pada pemanfaatan sekitar 97 persen satu inti, sehingga pemanfaatan CPU

total *server* tertahan pada kisaran 73 persen dan saturasi total pada 450 RPS tetap terjadi meskipun sekitar seperempat kapasitas CPU masih menganggur. Pada V2, penggandaan *instans* layanan *prediction* dan *worker* Python memungkinkan beban jalur inferensi tersebar ke beberapa inti sehingga pemanfaatan CPU total dapat mendekati 98 persen, dan saturasi total baru terjadi pada 500 RPS ketika seluruh inti benar-benar jenuh. Pada kedua arsitektur pemanfaatan memori tetap rendah, sehingga *bottleneck* kapasitas bersumber dari CPU, bukan dari memori.

Ketujuh, ekspektasi penelitian terpenuhi, meskipun tidak dalam bentuk yang semula diperkirakan. Penerapan NGINX dan PM2 tidak meningkatkan kapasitas puncak maupun mempercepat *respons* pada beban rendah. Peningkatan kinerja yang terukur justru bermanifestasi sebagai perbaikan kestabilan, prediktabilitas, dan sifat kegagalan sistem pada beban mendekati serta melampaui kapasitas. Dengan demikian, manfaat arsitektur usulan terletak pada keandalan operasional, bukan pada *throughput* maksimum.

5.2 Keterbatasan Penelitian

Berdasarkan Hasil penelitian ini perlu dipahami dalam konteks keterbatasan lingkungan eksperimen yang digunakan, yang memengaruhi sejauh mana temuan dapat digeneralisasi ke kondisi lain.

Pertama, dari sisi spesifikasi perangkat keras, seluruh pengujian dilakukan pada satu VPS dengan konfigurasi tetap, yaitu 4 vCPU, 8 GB RAM, dan penyimpanan HDD. Angka kapasitas absolut yang diperoleh dalam penelitian ini, seperti batas deterministik stabil 250 RPS dan titik saturasi total 450 RPS pada V1 serta 500 RPS pada V2, bersifat spesifik terhadap konfigurasi tersebut. Pada perangkat keras dengan jumlah inti CPU yang berbeda, jenis penyimpanan yang lebih cepat seperti SSD, atau kapasitas memori yang berbeda, nilai kapasitas absolut kemungkinan akan bergeser. Dengan demikian, yang lebih dapat digeneralisasi dari penelitian ini bukanlah angka RPS absolutnya, melainkan pola relatif berupa perbaikan kestabilan dan pergeseran ambang kegagalan pada arsitektur usulan dibandingkan *baseline*.

Kedua, dari sisi konfigurasi jaringan, pengujian dilakukan dengan generator beban dan *server* aplikasi yang berada pada lingkungan jaringan internal yang sama, sehingga faktor latensi jaringan publik, variasi kondisi internet, dan jarak geografis antara klien dan *server* tidak tercakup dalam pengukuran. Latensi yang terukur dalam penelitian ini karena itu lebih merepresentasikan waktu pemrosesan di sisi *server*, bukan latensi ujung ke ujung yang dialami pengguna nyata melalui internet publik.

Ketiga, dari sisi pola beban, pengujian menggunakan pola *constant arrival rate* dengan satu jenis permintaan utama pada *endpoint* /api/predict. Pola beban produksi nyata umumnya lebih bervariasi, mencakup fluktuasi laju permintaan, campuran berbagai jenis *endpoint*, dan distribusi ukuran *payload* yang beragam. Karena itu, perilaku sistem pada kondisi beban yang lebih kompleks dan tidak seragam berada di luar cakupan pengujian ini.

Keempat, dari sisi cakupan arsitektur, penerapan NGINX dalam penelitian ini dibatasi pada fungsi *reverse proxy* dan *load balancing* terhadap beberapa instans pada satu *server* tunggal. Skenario *load balancing* lintas beberapa node *server*, mekanisme *caching* pada NGINX, serta penyetelan lanjutan seperti kompresi *respons* dan parameter *keepalive* tidak termasuk dalam ruang lingkup yang diuji. Demikian pula, analisis dilakukan menggunakan perbandingan nilai deskriptif tanpa uji statistik formal, sehingga signifikansi statistik dari perbedaan antararsitektur tidak diukur secara kuantitatif.

Keterbatasan-keterbatasan tersebut tidak mengurangi validitas temuan dalam konteks lingkungan uji yang digunakan, karena seluruh perbandingan dilakukan pada kondisi yang setara antara kedua arsitektur. Namun, keterbatasan ini menjadi dasar bagi sejumlah arah pengembangan yang diuraikan pada subbab berikutnya.

5.3 Saran

Berdasarkan keterbatasan dan temuan dalam penelitian ini, beberapa saran dapat diajukan untuk pengembangan selanjutnya.

Pertama, pengujian dapat diperluas pada variasi perangkat keras yang berbeda, seperti penggunaan penyimpanan SSD dibandingkan HDD serta jumlah inti CPU yang lebih banyak, untuk mengamati sejauh mana kapasitas kedua arsitektur berubah seiring perubahan sumber daya. Hal ini juga akan memperjelas batas hingga mana penambahan *worker* memberikan manfaat sebelum jumlah proses melampaui jumlah inti yang tersedia.

Kedua, metrik pengujian dapat diperluas dengan menambahkan persentil yang lebih ekstrem seperti P99.9 untuk menangkap ekor distribusi yang lebih jauh, serta menerapkan uji statistik formal terhadap perbedaan antararsitektur, sehingga signifikansi perbedaan dapat diukur secara kuantitatif dan tidak hanya berdasarkan perbandingan nilai deskriptif.

Ketiga, pengujian dapat dilakukan dengan model *machine learning* yang lebih berat atau dengan beban inferensi yang lebih besar, untuk mengamati apakah keunggulan distribusi kapasitas inferensi pada V2 menjadi semakin signifikan ketika inferensi menjadi *bottleneck* utama sistem.

Keempat, arsitektur dapat dikembangkan lebih lanjut dengan mekanisme penyesuaian jumlah *worker* secara dinamis atau *autoscaling* berdasarkan beban aktual, sehingga sistem dapat menyesuaikan kapasitasnya secara otomatis tanpa konfigurasi manual.

Kelima, pengujian dapat menggunakan pola beban yang lebih menyerupai kondisi produksi nyata, seperti pola kedatangan permintaan yang berfluktuasi dan campuran jenis permintaan yang beragam, untuk melengkapi hasil pengujian dengan pola beban konstan yang digunakan dalam penelitian ini.

Keenam, optimisasi lebih lanjut pada konfigurasi NGINX dapat dieksplorasi, seperti penyetelan kompresi pada respons *backend* dan penyetelan parameter *keepalive*, untuk mengamati dampaknya terhadap performa pada berbagai kondisi beban.