JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 8
No. 2
Agustus
2022

Implementasi dan Analisa Performa Berbagai *Platform Web server* pada Lingkungan *Multicore*

Kharisma Monika Dian Pertiwi[#1], Rizky Fenaldo Maulana[*2], Ardian Yusuf Wicaksono[#3]

[#]*Informatika, Institut Teknologi Telkom Surabaya*
Jl. Ketintang No.156, Ketintang, Kec. Gayungan, Kota Surabaya
[1]kharismamonika@ittelkom-sby.ac.id
[3]ardian@ittelkom-sby.ac.id

[*]*Institut Teknologi Telkom Surabaya*
Jl. Ketintang No.156, Ketintang, Kec. Gayungan, Kota Surabaya
[2]aldo@ittelkom-sby.ac.id

Abstrak*—Perkembangan teknologi informasi dan komunikasi (TIK) sangat pesat. Saat ini hampir semua organisasi menggunakan sistem informasi berbasis website untuk mempermudah pekerjaannya. Perkembangan TIK tersebut juga didukung dengan perkembangan perangkat keras dan perangkat lunak. Salah satu isu utama dalam sistem berbasis website adalah kualitas layanan (performa). Faktor yang dapat mempengaruhi performa adalah aspek perangkat lunak dan perangkat keras. Salah satu perangkat keras yang berkembang adalah CPU dengan teknologi *multicore*. Teknologi *multicore*, mampu menjalankan proses secara paralel. Sehingga diharapkan dengan teknologi *multicore*, performa layanan menjadi lebih cepat. Selain dari aspek perangkat keras, performa layanan juga dipengaruhi aspek perangkat lunak yang berjalan dibawahnya, salah satunya adalah *web server*. Saat ini berkembang banyak *web server*. Kualitas berbagai *web server* perlu menjadi perhatian, agar layanan tetap baik. Uji coba terhadap teknologi *multicore* dan berbagai jenis *web server* perlu dilakukan untuk mengetahui efek terhadap kualitas layanan. Oleh karena itu penelitian ini dilakukan untuk mengetahui kualitas layanan berbagai macam *web server* dalam lingkungan *multicore*. Pengujian dilakukan dengan simulasi permintaan dengan variasi jumlah *core* dan konkurensi. Hasil pengujian menunjukkan bahwa penambahan *core* mempengaruhi jumlah permintaan yang dapat dilayani tiap detik, waktu respons *server* lebih cepat dan mengurangi beban masing-masing *core*. Hasil menunjukkan bahwa rata-rata peningkatan jumlah permintaan yang dapat dilayani sekitar 4000 hingga 5000.

Kata kunci*— *Multicore*, Performa, *Web server

I. PENDAHULUAN

Sistem Informasi berbasis website merupakan hal yang sudah sering didengar. Perkembangan sistem informasi berbasis website seiring dengan perkembangan teknologi informasi dan komunikasi. Aplikasi berbasis website berjalan diatas *web server*. Kemampuan *web server* juga didukung dengan perangkat keras yang ada di bawahnya. Salah satu perangkat keras yang dapat mempengaruhi performa *web server* adalah CPU. CPU memiliki bagian penting yang disebut dengan *core*. *Core* adalah unit dasar yang membaca dan menjalankan instruksi dalam program komputer. CPU *single core* hanya memiliki satu *core* yang mengeksekusi instruksi tunggal pada suatu waktu. CPU *multicore* memiliki inti pemrosesan lebih dari satu pada satu chip, sehingga dapat meningkatkan kinerja [1].

Selain faktor dari perangkat keras *web server*, faktor perangkat lunak *web server* juga dapat mempengaruhi kinerja *web server*. Contoh perangkat lunak *web server* adalah Apache, NodeJS, NginX, Go dan Elixir.

Isu utama dari layanan denga teknologi berbasis website adalaga kualitas layanan atau QoS (*Quality of Service*). Pada penelitian [2], disebutkan bahwa kemampuan kinerja layanan aplikasi berbasis website direpresentasikan dengan kecepatan *web server* dalam melayani menyelesaikan permintaan dari klien. Aspek yang dapat mempengaruhi layanan sistem informasi berbasis website adalah *platform web server*. Hal tersebut didukung oleh Toyotaro dalam penelitiannya [3], implementasi layanan web pada *web server* yang berbeda *platform* akan menghasilkan kinerja yang berbeda. Oleh karena itu penelitian ini melakukan uji coba untuk mengukur kinerja terhadap *web server* yang berbeda *platform*.

Performa *web server* menjadi pertimbangan utama bagi penyedia layanan berbasis internet dalam melayani permintaan dari klien. Semakin baik performa dari *web server*, maka semakin kecil biaya yang dikeluarkan untuk melayani permintaan klien. Untuk meningkatkan performa dari *web server*, penyedia layanan dapat menambah *core* dari prosesor agar komputasi dapat dijalankan secara paralel. Secara umum, semakin banyak *core* pada *server*, maka komputasi dapat dilakukan dengan lebih cepat. Hal ini karena beban kerja dibagi dengan banyaknya *core* yang tersedia. Untuk menentukan *web server* yang lebih

Submitted 30-03-2022; Revised 18-04-2022; Accepted 24-07-2022

cepat/lebih baik, maka perlu dilakukan pengujian yang membandingkan *web server* di lingkungan yang sama.

Penelitian ini melakukan uji coba untuk mengetahui performa *web server* dalam melayani klien pada lingkungan *multicore*. Tujuan dari penelitian ini adalah untuk mengetahui performa *web server* ketika jumlah *core* pada *server* berbeda-beda. Berdasarkan studi literatur yang telah dilakukan, belum terdapat penelitian yang membandingkan berbagai macam *web server* pada lingkungan *multicore*. *Web server* yang akan dibandingkan performanya adalah NginX, nodeJS, Golang dan Elixir. Penulis memilih 4 *web server* tersebut karena *web server* tersebut merupakan *web server* yang popular pada akhir dekade ini. Diharapkan dengan penelitian ini dapat menjadi referensi bagi para pengembang aplikasi terdistribusi dalam meningkatkan layanan aplikasi berbasis website.

II. PENELITIAN TERKAIT

Pada tahun 2015 [4] Yogiswara melakukan pengukuran kinerja tiga jenis *platform web server* yang berbeda yaitu Apache *Web server*, NginX *Web server* dan IIS7 *web server*. Penelitian tersebut mengevaluasi nilai *latency* dari berbagai jenis *web server* tersebut. *Latency* adalah waktu yang diperlukan mulai adanya permintaan dari klien hingga *server* mengirimkan respon dari permintaan tersebut. Berdasarkan hasil penelitian didapatkan penggunaan *web server* NginX dengan menggunakan metode rest memiliki nilai *latency* terbaik ketika diimplementasikan pada transaksi penambahan, perubahan, pemilihan dan penghapusan data.

Abdul Aziz dkk [5] menganalisis kinerja *web server* Apache dan NginX untuk pengembangan *server hosting* pada institusi. Uji coba tersebut diimplementasikan pada jaringan enterprise Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta. Hasil uji coba tersebut *server web* dengan menggunakan Apache lebih unggul dibandingkan dengan *server web* yang menggunakan Nginx. Kinerja Apache unggul dalam *transfer* data, *connection*, dan permintaan data, sehingga memudahkan klien dalam mendapatkan informasi yang dibutuhkan.

Van Nam Nguyen [6] melakukan analisa secara empiris dari beberapa *web server* seperti Apache, NodeJS dan NginX untuk mencari tahu *trade-off* antara desain perangkat lunak yang berbeda. Penelitiannya bermanfaat untuk mengatasi hambatan kinerja dan masalah konsumsi sumber daya. Hasil percobaan menunjukkan bahwa Nginx dapat mencapai tingkat konkurensi hingga 42% dan hingga 200% lebih tinggi daripada Node.js dan Apache. Selain itu penelitiannya juga juga menunjukkan bahwa Nginx bisa mendapatkan *throughput* 16% lebih tinggi daripada Apache dalam jangka panjang dengan volume beban yang tinggi. *Response time* pada Nginx juga dua kali lebih kecil dari Apache.

Dr. Amita Goel dkk pada penelitiannya [7] melakukan uji coba perbandingan *web server* yang berbeda. Uji coba ini membandingkan Apache, GlassFish, IBM HTTP *server* dan Nginx berdasarkan *support*, *cost*, dan kemampuan. Uji coba tersebut menyimpulkan bahwa Apache adalah *server web* paling populer karena berfungsi pada hampir semua jenis Sistem Operasi dan berfungsi pada *platform* yang mendukung C dan Java. Nginx adalah alternatif yang sangat populer karena sangat cepat dan sangat ringan. Juga hampir setengah dari situs *web* menggunakan *server web* Apache.

Rahmad Dawood dkk. [8] melakukan pengujian kelayakan dan kinerja aplikasi *server web* yang dijalankan pada *Raspberry* Pi. Pengujian dilakukan terhadap tiga aplikasi *server web* yang paling populer saat ini, yaitu: Apache, Nginx, dan Lighttpd. Dengan memakai parameter kinerja: maximum permintaan dan *reply time*. Hasil pengujian menunjukkan bahwa Raspberry Pi layak untuk menjalankan semua aplikasi *server web* yang diuji namun kinerja terbaik diberikan oleh Nginx yang diikuti oleh Lighttpd dan Apache.

Prakash P dkk [9] menyelidiki kinerja dua *server web* dalam skenario yang berbeda yaitu Apache - process-based *server web* dan Nginx - *asymmetric multi-process event-driven architecture* (AMPED). Hasil uji coba menunjukkan bahwa *server* Nginx mengungguli *server* Apache dalam hal responsif dan skalabilitas, sementara Apache unggul dalam hal efisiensi. Hasil ini bermanfaat bagi mereka yang ingin meng-*host* situs *web* di *server* pribadi *virtual* (VPS).

Dalam penelitian Qi Fan dkk. [10] melihat tingkat responsif *server web* yang berbeda arsitektur (asinkron dan berbasis *thread*) dibawah beban kerja konkurensi yang tinggi. Melalui pengujian menggunakan *Apache Bench*, ditunjukkan bahwa *server web* dengan arsitektur asinkron dapat mencapai ketahanan yang lebih baik dalam menangani beban dengan konkurensi tinggi dibandingkan *server web* berbasis *thread*. Hasil analisis menunjukkan bahwa *server* berbasis *thread* rentan terhadap beban kerja dengan konkurensi yang tinggi karena ukuran antreannya yang terbatas dalam permintaan bersamaan yang tinggi, hal ini menyebabkan meluapnya antrean dan *response time* yang lama karena transmisi ulang TCP.

Pada penelitian ini, penulis melakukan uji coba dengan membandingkan kinerja *web server* pada lingkungan *multicore*. *Web server* yang akan dibandingkan adalah Go, NginX, NodeJS dan Elixir. Pengujian dilakukan dengan menyimulasikan permintaan dari pengguna. Pengukuran performa dilihat dari jumlah permintaan yang dapat dikerjakan oleh *web server* tiap detik, rata-rata waktu respons dan data persentase penggunaan CPU.

III. TINJAUAN PUSTAKA

Server adalah komputer atau perangkat di jaringan yang mengelola sumber daya jaringan. Ada berbagai jenis *server* seperti *server* dial-up, *server* printer dan *server web* [11]. Fungsi *web server* adalah menyimpan proses dan mengirimkan halaman situs web yang diminta pengguna. *Web server* menyimpan *file* halaman web seperti HTML, *file* CSS, *template* dan *file-file* lain yang penting [7]. Aplikasi *web server* merupakan perangkat lunak yang

berjalan di sisi *server* dan bertugas untuk menerima permintaan dari *web browser*, menerjemahkan permintaan tersebut dan mengembalikan hasil permintaan ke *web browser*. Saat ini tersedia berbagai jenis *platform web server*. Penelitan ini menggunakan *platform web server* : NginX, NodeJS, Golang dan Elixir.

A. *NginX*

Nginx merupakan perangkat lunak *open source* dan gratis yang berfungsi sebagai *server* web. Selain itu, NginX juga berfungsi sebagai *reverse proxy* untuk protokol TCP, UDP, HTTP, HTTPS, SMTP, POP3 dan IMAP serta *load balancer* dan *cache* HTTP [7]. NGinX dapat dengan cepat memberikan konten statis. NginX dibangun secara modular sehingga mampu mendukung berbagai fitur seperti *Load Balancing*, *Reverse proxy*, *Virtual Host* berbasis nama dan IP, Fast CGI, akses langsung ke *cache*, SSL, flash video streaming dan sejumlah fitur-fitur standar lainnya. Nginx dapat dijalankan dan tersedia untuk *platform* Unix, Linux, MacOS X, solaris dan Microsoft Windows [12]. NginX menjadi pilihan berkat kinerjanya yang tinggi, stabilitas, kekayaan akan fitur, mudah dikonfigurasi dan hemat sumber daya [13].

Secara umum *web server* NginX menerapkan model *event-based*. Model *event-based* hanya menggunakan sebuah *thread* untuk memproses permintaan dan mengelola event dengan *multiplexing*. Setiap koneksi diproses secara sangat efisien dalam sejumlah proses *single thread* yang disebut dengan *workers*. Dalam NginX, *workers* dapat menangani ribuan koneksi bersamaan dan permintaan per detik.

B. *NodeJS*

NodeJS adalah *single-threaded server-side* yang bekerja pada lingkungan javascript yang dikembangkan oleh Ryan Dahl pada tahun 2009 [12]. NodeJS dapat bertindak sebagai layanan *server web* karena memiliki *library* untuk komunikasi HTTP. Dengan *event-based architecture* yang *scalable*, Node.js dapat mencapai tingkat konkurensi yang tinggi [6]. Banyak modul utama di NodeJS ditulis dalam *Javascript* dan dieksekusi oleh Google V8 *Javascript* Engine. NodeJS banyak digunakan untuk aplikasi *realtime*. Chitra dkk [14] dalam penelitiannya menyatakan bahwa NodeJS memiliki sifat yang ringan dan efisien, sehingga bagus untuk mengembangkan layanan web.

C. *Golang*

Golang, atau singkatnya disebut dengan Go, merupakan Bahasa pemrograman untuk tujuan umum yang dikembangkan oleh Google pada November 2009. Go dapat digunakan untuk memprogram perangkat lunak dan infrastruktur. Go merupakan Bahasa pemrograman yang sederhana dan pragmatis. Pada halaman situs Go [15], dituliskan bahwa Go merupakan Bahasa pemrograman *open source* yang membuatnya mudah untuk membangun perangkat lunak yang sederhana, andal, dan efisien. Go menyediakan model pemrograman sederhana untuk membangun perangkat lunak yang efisien dengan dukungan konkurensi, selain itu go juga dapat digunakan untuk membangun *server* jaringan, alat infrastruktur, DevOps, aplikasi perangkat seluler, Internet of Things (IoT) dan *machine learning* [16].

D. *Elixir*

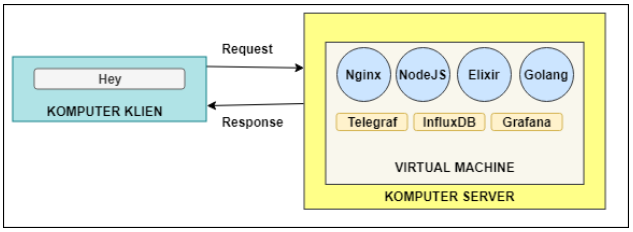
Elixir adalah bahasa yang dibangun untuk dijalankan di BEAM (Erlang *Virtual Machine*). Tujuannya adalah untuk meningkatkan semua abstraksi Erlang, sambil menambahkan fitur baru dari banyak bahasa pemrograman. Meskipun relatif baru, setelah memulai pengembangannya pada 2011, Elixir telah digunakan oleh sejumlah perusahaan untuk membangun aplikasi web [17].

Elixir merupakan Bahasa pemrograman berbasis fungsional. Elixir dirancang agar efektif dalam membangun aplikasi [18].

IV. PERANCANGAN SISTEM

Pada bagian ini dijelaskan mengenai rancangan penelitian yang akan dilakukan. Perancangan yang dilakukan adalah arsitektur sistem, lingkungan sistem, *web server* yang akan dibandingkan, skenario pengujian dan metrik yang akan dianalisis.

A. *Arsitektur dan Lingkungan Sistem*



Gambar. 1 Arsitektur sistem

Arsitektur jaringan yang digunakan dalam penelitian ini adalah *client-server*. Secara umum, arsitektur sistem dalam penelitian ini ditunjukkan pada Gambar. 1. Klien mengirimkan *request* ke *server* lalu *server* memberikan respons. Komunikasi antara komputer klien dan *server* menggunakan protokol HTTP. Dalam uji coba ini, komputer klien dan *server* terhubung dalam satu sub net jaringan LAN. Protokol jaringan yang digunakan dalam penelitian ini adalah HTTP/1.1. Hal tersebut agar klien dapat mengirimkan banyak permintaan dalam satu koneksi HTTP.

Dalam penelitian ini, *server* dibangun dalam lingkungan *virtual machine* (VM). Hal ini karena dengan menggunakan VM dapat dilakukan konfigurasi terhadap jumlah *core* yang akan digunakan. Selanjutnya dalam VM tersebut di *install* 4 *web server* yang berbeda. Penjelasan detail mengenai komputer klien dan *server* akan dijelaskan dalam sub-bab berikut :

1) *Komputer Klien:* Penelitian ini menggunakan sebuah komputer yang akan digunakan sebagai klien. Komputer tersebut akan melakukan *generate traffic* dan

monitoring hasil. *Tools* yang digunakan untuk *generate traffic* adalah Hey [17]. *Tools* ini memiliki fitur yang sama dengan Apache Bench, tetapi dapat mengirimkan beban menggunakan HTTP/1.1. Spesifikasi komputer klien yang digunakan dapat dilihat pada Tabel I.

TABEL I
SPESIFIKASI KOMPUTER KLIEN

Komponen	Spesifikasi
CPU	Intel(R) *Core*(TM) i3-3240 CPU @ 3.40GHz
Sistem Operasi	Ubuntu 18.04.2 LTS
Memori	8 GB
Kecepatan Ethernet	100 Mb/s

2) *Komputer Server:* Penelitian menggunakan sebuah komputer yang akan digunakan sebagai *server*. Spesifikasi komputer yang digunakan untuk *server* dapat dilihat pada Tabel II. Komputer *host* yang digunakan dalam penelitian ini yaitu komputer yang menggunakan prosesor Intel(R) *Core*(TM) i5-7400 CPU @ 3.00GHz yang memiliki 4 *core*. Komputer ini memiliki RAM sejumlah 8 GB dan Ethernet dengan kecepatan 100 Mb/s. *Virtual machine* yang digunakan adalah *Virtual*Box versi 5.2.18 dengan sistem operasi Ubuntu *Server* 16.04 dengan konfigurasi jumlah *core* 1 hingga 4 *core*. Uji coba ini menggunakan pengaturan *execution* cap sejumlah 100%, dalam artian, VM dapat menggunakan CPU yang terdapat pada sistem *host* secara penuh. Penulis menggunakan Telegraf, InfluxDB [18], dan Grafana untuk pemantauan sumber daya CPU. Telegraf digunakan untuk mengumpulkan data penggunaan CPU. Telegraf mengirimkan data ke InfluxDB dan Grafana menampilkan data yang disimpan di CollectDB menggunakan browser. InfluxDB juga menyediakan antarmuka berupa kueri SQL untuk keperluan automasi.

TABEL II
SPESIFIKASI LINGKUNGAN *HOST* DAN *VIRTUAL MACHINE SERVER*

Komponen	Spesifikasi
CPU	Intel(R) *Core*(TM) i5-7400 CPU @ 3.00GHz, 4 *Core*
Sistem Operasi *Host*	Ubuntu 18.04.2 LTS
Sistem Operasi VM	Ubuntu 16.04.6 LTS
Memori *Host*	8 GB
Memori VM	4 GB
Hypervisor	*Virtual*Box

3) *Web server:* Dalam penelitian ini menggunakan 4 *platform web server* : NginX, NodeJS, Go dan Elixir. 4 *web server* tersebut diinstall dalam sebuah VM diatas komputer *server*. NginX yang digunakan adalah Nginx versi 1.10.3 dan PHP versi 7.2.18. Jumlah *worker* yang digunakan adalah menyesuaikan dengan jumlah *core* yang digunakan. NodeJS yang digunakan adalah NodeJS 12.2.0 dengan modul Express XX. Agar bisa multi-*thread*ing, penulis menambahkan modul node-cluster dengan jumlah *worker* sesuai dengan jumlah *core* yang digunakan. Go yang digunakan yaitu Go versi 1.12.5. Elixir yang digunakan yaitu Elixir 1.8.1 dikompilasi dengan Erlang/OTP 20.

B. Skenario Pengujian

Parameter pengujian dapat dilihat pada TABEL III. Pengujian ini menggunakan variasi jumlah *core* 1 hingga 4 *core*. Pada masing-masing *core*, pengujian dilakukan dengan variasi konkurensi dari 1 hingga 1000. Masing-masing pengujian dilakukan selama 30 detik dengan mengirimkan halaman situs berupa html dengan ukuran 135 bytes sebanyak-banyaknya. Dalam halaman tersebut ter-dapat satu variabel dinamis yang berubah-ubah sesuai teks yang dikirim oleh klien.

TABEL III
PAREMETER PENGUJIAN

Variabel	Nilai
CPU *Core*	1, 2, 3, 4
Ukuran Halaman	135 *bytes*
Waktu Pengujian	30 detik
Protokol	HTTP/1.1
Konkurensi	1, 10, 50, 100, 1000

C. Metrik Analisis

Setelah dilakukan pengujian, akan dilakukan evaluasi untuk mengetahui kinerja *server* web. Metrik yang akan dianalisis pada uji coba ini adalah persentase penggunaan CPU, jumlah permintaan tiap detik dan waktu respons *server*. Parameter-parameter tersebut dianalisis untuk dapat membandingkan performa *server* web. Penjelasan untuk masing-masing metrik analisis adalah sebagai berikut:

1) *Persentase Penggunaan CPU:* Dalam uji coba ini, penggunaan CPU adalah persentase beban kerja CPU pada *server*. Dalam hal ini adalah kinerja *virtual machine*.

2) *Jumlah Permintaan Tiap Detik:* Permintaan tiap detik adalah jumlah permintaan HTTP yang mampu ditangani oleh *server* setiap detik dari klien.

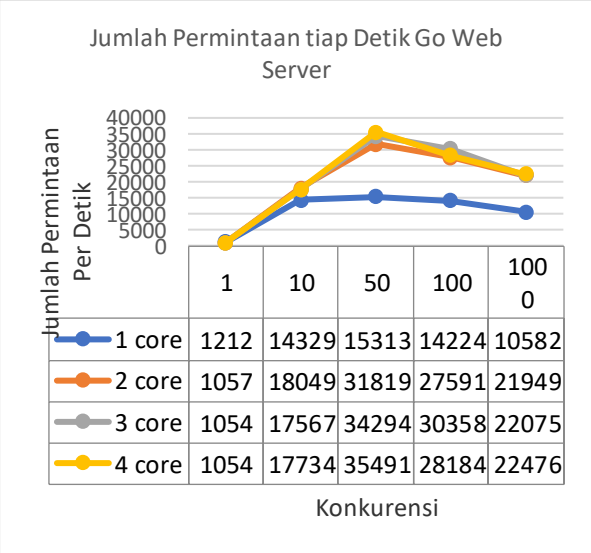
3) *Rata-rata Waktu Respons:* Rata-rata waktu respons adalah rata-rata waktu yang digunakan dalam proses pengiriman 1 paket data. Waktu mulai dihitung saat klien mengirimkan permintaan hingga paket diterima oleh klien.

V. HASIL DAN PEMBAHASAN

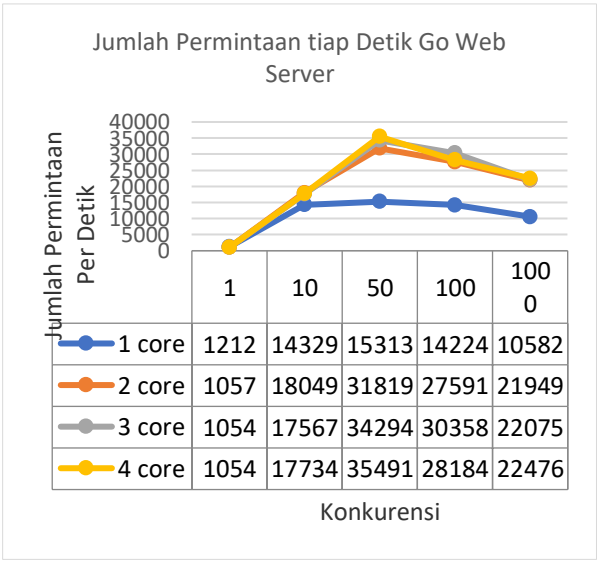
Pada bab ini dibahas hasil evaluasi dan analisa terhadap *web server* sesuai dengan metrik analisis.

A. Jumlah Permintaan Tiap Detik

Hasil evaluasi jumlah permintaan yang dapat dilayani oleh *web server* Go dapat dilihat pada

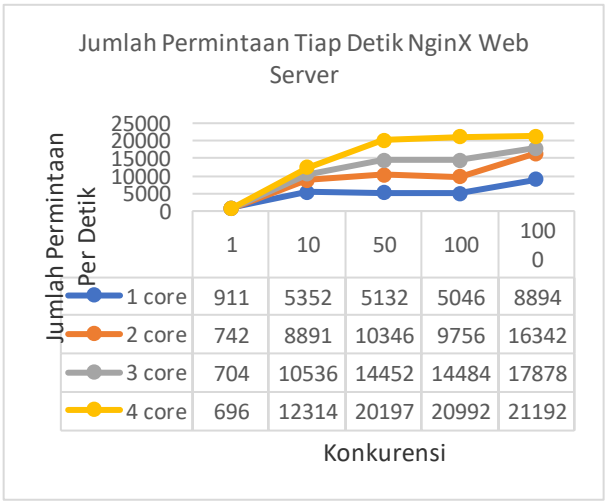


Gambar. 2. Berdasarkan gambar tersebut dapat dilihat bahwa penambahan jumlah *core* tidak menunjukkan perbedaan yang signifikan terhadap jumlah permintaan yang dapat dilayani. Dari hasil pengujian tersebut performa *web server* Go maksimal pada tingkat konkurensi 50. Namun performa menurun pada tingkat konkurensi 100 dan 1000. Hal ini logis karena jumlah permintaan sebanyak 100 dan 1000 yang datang pada waktu yang bersamaan. Ketika jumlah permintaan banyak, sumber daya pada *server* harus dibagi.

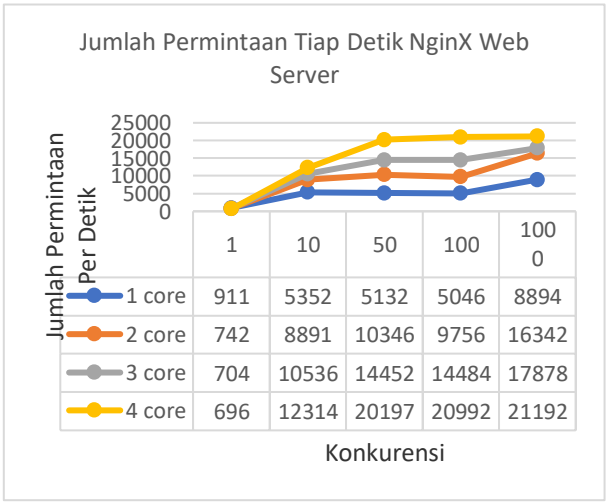


Gambar. 2 Jumlah permintaan tiap detik Go *Web server*

Hasil evaluasi jumlah permintaan yang dapat dilayani tiap detik pada *web server* NginX dapat dilihat pada

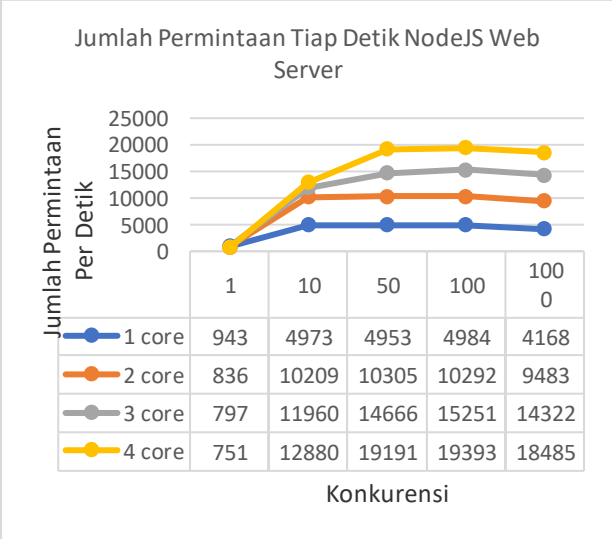


Gambar. 3. Pada *web server* NginX terlihat bahwa dengan menambah jumlah *core* maka jumlah permintaan yang dapat dilayani juga semakin banyak. Pada tingkat konkurensi 50,100 dan 1000, performa *web server* NginX cukup stabil dalam melayani jumlah permintaan.

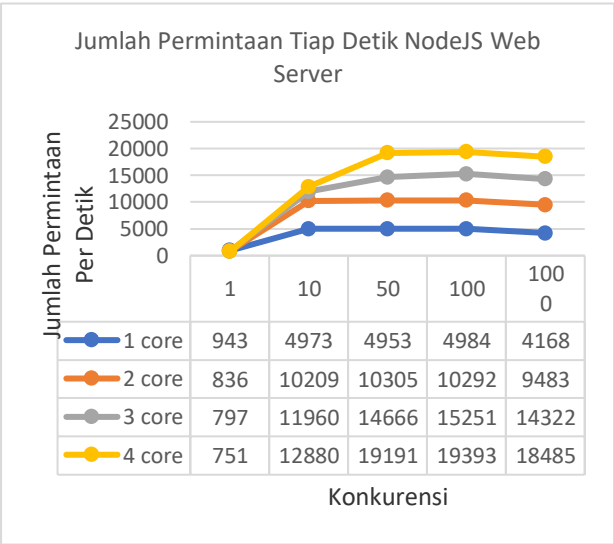


Gambar. 3 Jumlah tiap permintaan tiap detik NginX *Web server*

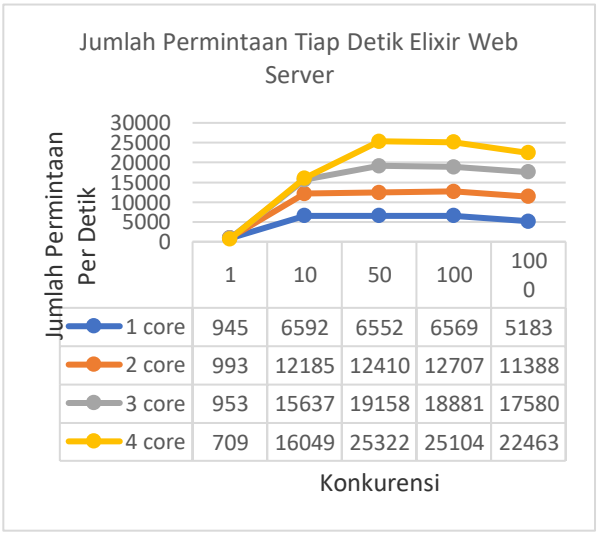
Jumlah permintaan per detik pada NodeJS dapat dilihat pada



Gambar. 4 dan permintaan per detik pada Elixir dapat dilihat pada Gambar. 5. Kenaikan jumlah permintaan terjadi secara seragam pada NodeJS dan Elixir sesuai dengan jumlah *core*. Kenaikan tingkat konkurensi juga tidak berdampak signifikan pada performa kedua *server* web tersebut.



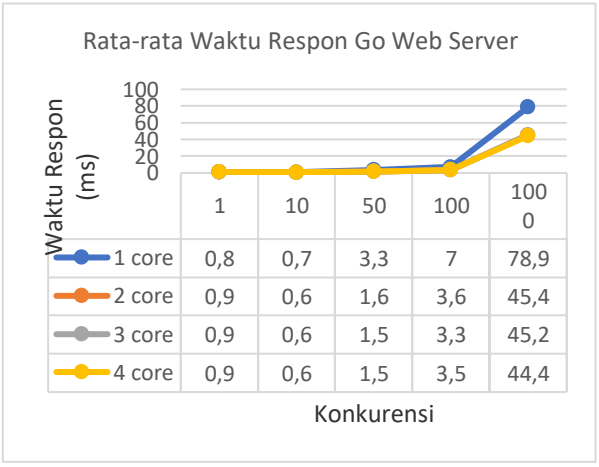
Gambar. 4 Jumlah permintaan tiap detik NodeJS *Web server*



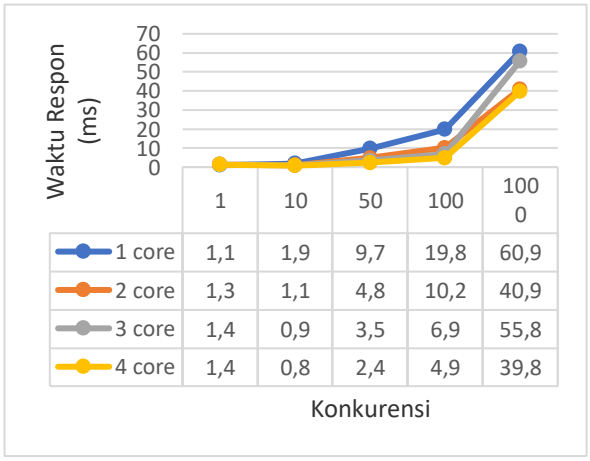
Gambar. 5 Jumlah permintaan tiap detik Elixir *Web server*

B. Rata-rata Waktu Respon

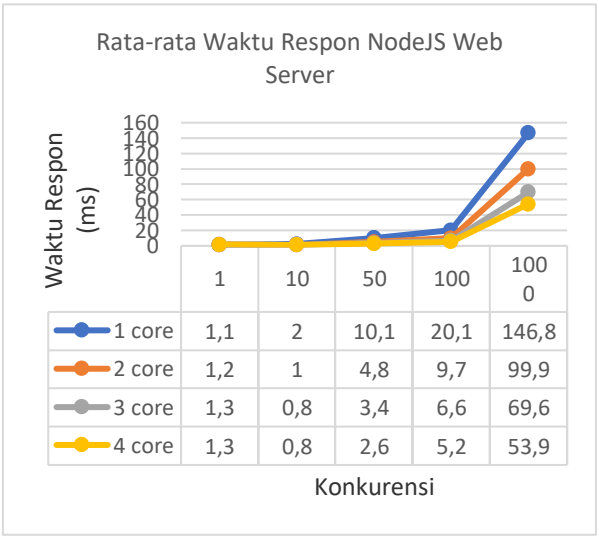
Waktu respons adalah waktu yang dibutuhkan pada saat permintaan datang hingga *server* memberikan balasan. Hasil evaluasi waktu respons Go *web server* dapat dilihat pada Gambar. 6. Dari gambar tersebut terlihat bahwa waktu respons stabil hingga tingkat konkurensi 100. Waktu respons naik pada saat konkurensi 1000, berada di sekitar 45 ms, kecuali pada lingkungan 1 *core*.



Gambar. 6 Rata-rata waktu respon Go *Web server*



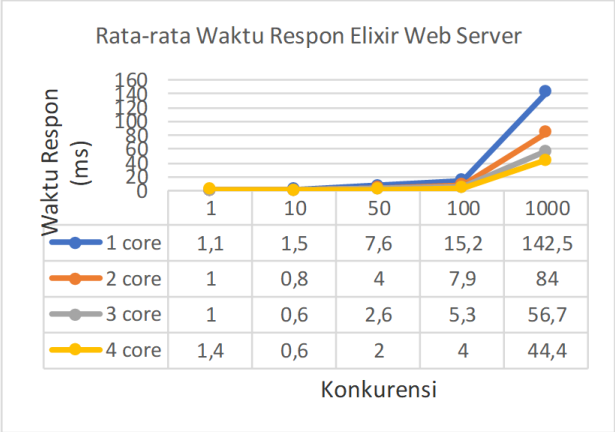
Gambar. 7 Rata-rata waktu respon NginX *Web server*



Gambar. 8 Rata-rata waktu respon NodeJS *Web server*

Hasil evaluasi rata-rata waktu respon pada NginX *web server* ditunjukkan pada Gambar. 7. Dari gambar tersebut, dapat terlihat bahwa rata-rata waktu respon stabil hingga konkurensi 100 pada semua variasi *core*. Hasil evaluasi menunjukkan bahwa pada lingkungan 1 *core* dengan tingkat konkurensi 50 dan 100 terjadi rata-rata waktu respon yang tinggi dibandingkan dengan yang lain. Selain itu juga terlihat bahwa rata-rata waktu respon *web server* di lingkungan 4 *core* cenderung lebih rendah dibandingkan dengan yang lain. Berdasarkan evaluasi tersebut dapat dikatakan bahwa penambahan jumlah *core* mampu menurunkan jumlah waktu yang digunakan untuk respon. Dengan waktu respon yang rendah, diharapkan dapat meningkatkan performa layanan.

Hasil evaluasi rata-rata waktu respon *web server* NodeJS ditunjukkan pada Gambar. 8 dan hasil evaluasi rata-rata waktu respon ditunjukkan pada Gambar. 9. Dari kedua gambar tersebut menunjukkan bahwa waktu respon Node.js dan Elixir mengalami penurunan yang seragam ketika terjadi penambahan jumlah *core*. Hal tersebut terjadi karena Node.js dan Elixir dapat mendistribusikan beban pada masing-masing *core* dengan baik sehingga waktu tunggu untuk eksekusi pada masing-masing *core* semakin berkurang.

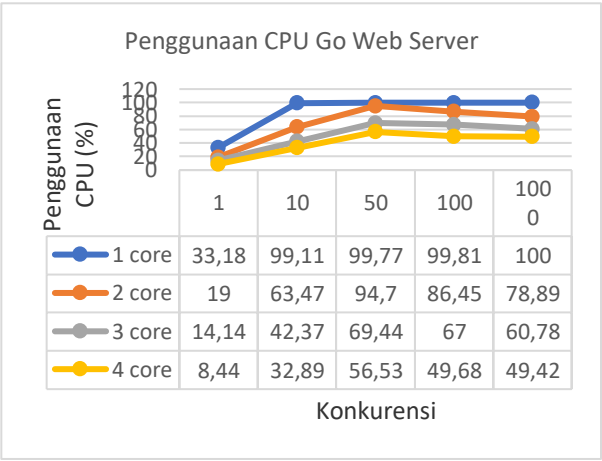


Gambar. 9 Rata-rata waktu respon Elixir *Web server*

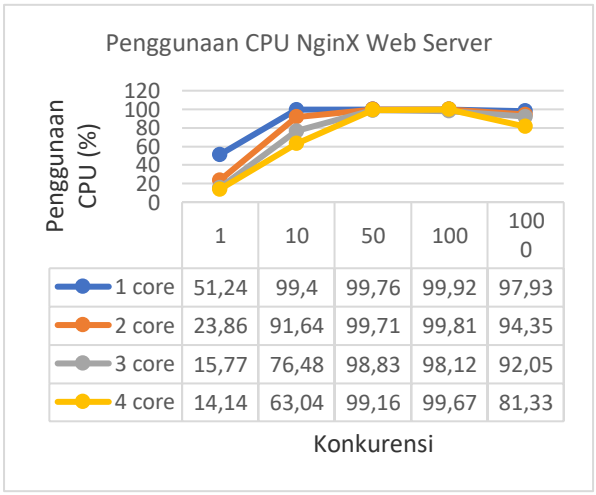
C. Persentase Penggunaan CPU

Secara umum, penambahan jumlah *core* pada *server* menjadikan penggunaan total dari CPU menjadi berkurang karena beban terdistribusi pada masing-masing *core*. Pada tingkat konkurensi 1, rata-rata penggunaan CPU hanya sekitar 40% karena terdapat *latency* pengiriman sehingga CPU tidak digunakan dalam beberapa mikrodetik. Kemudian penggunaan CPU mencapai puncak ketika tingkat konkurensi sebesar 50 ke atas, kecuali Go.

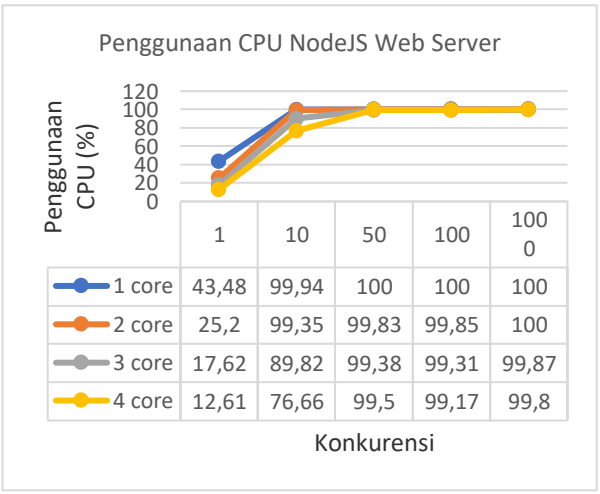
Hasil evaluasi persentase penggunaan CPU pada Go *web server* ditunjukkan pada Gambar. 10. Dari gambar terlihat bahwa semakin banyak *core* maka semakin rendah persentase penggunaan CPU. Selain itu terlihat bahwa persentase penggunaan CPU meningkat lalu relatif stabil seiring dengan peningkatan jumlah konkurensi.



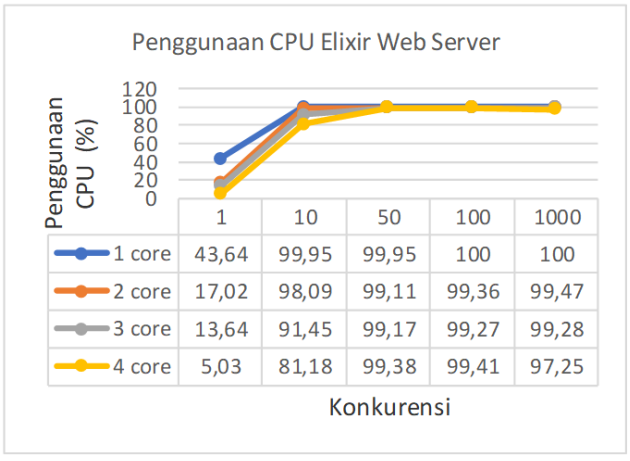
Gambar. 10 Persentase penggunaan CPU Go *Web server*



Gambar. 11 Persentase penggunaan CPU NginX *Web server*



Gambar. 12 Persentase penggunaan CPU NodeJS *Web server*



Gambar. 13 Persentase penggunaan CPU Elixir *Web server*

Hasil evaluasi persentase penggunaan CPU pada NginX, NodeJS dan Elixir *web server* secara berturut-turut ditunjukkan pada Gambar. 11, Gambar. 12, Gambar. 13. Persentase Penggunaan CPU pada semua *server* relatif seragam. Dari gambar terlihat bahwa semakin banyak *core* maka semakin rendah persentase penggunaan CPU. Selain itu terlihat bahwa persentase penggunaan CPU meningkat lalu relatif stabil seiiring dengan peningkatan jumlah konkurensi.

VI. KESIMPULAN

Pada penelitian ini, penulis melakukan uji coba membandingkan performa *web server* pada lingkungan *multicore*. Pengujian beban dilakukan dengan variasi tingkat konkurensi. Hasil pengujian menunjukkan bahwa dengan menambah jumlah *core web server*, mampu meningkatkan performa *web server* dalam melayani permintaan tiap detik. Pada NginX *web server* rata-rata peningkatan jumlah permintaan yang dapat dilayani 4000 hingga 5000. Selain itu dengan menambahkan *core* dapat mempercepat waktu respon *server* dan dapat mengurangi beban setiap *core server*. Hasil pengujian menunjukkan bahwa penambahan jumlah *core* mampu mempersingkat waktu hingga 50%. Di sisi lain, berapa pun jumlah *core* yang terdapat dalam *server*, terdapat kemampuan maksimal *server web* dalam mendistribusikan beban ke masing-masing *core*. Apabila beban dalam hal konkurensi melebihi kemampuan maksimal *server web* tersebut, maka performa mulai menurun.

REFERENSI

[1] N. Dilawar, M. Zakarya, dan I. U. Rahman, "A review of power efficient *load balancing* algorithms for *multicore* sistems," *World Appl. Sci. J.*, vol. 27, no. 9, hal. 1175–1182, 2013.

[2] M. Conti dan M. Kumar, "Quality of service in web services," vol. 12, no. 1, hal. 3550–3550, 2005.

[3] T. Suzumura, S. Trent, M. Tatsubori, A. Tozawa, dan T. Onodera, "Performance comparison of Web service engines in PHP, Java, and C," *Proc. IEEE Int. Conf. Web Serv. ICWS 2008*, hal. 385–392, 2008.

[4] R. P. Lunak, "Rekayasa Perangkat Lunak Kinerja Web Service pada *Web server* Apache , Ngin-X dan IIS-7," hal. 175–179, 2015.

[5] A. Aziz dan T. Tampati, "Analisis *Web server* untuk Pengembangan *Hosting Server* Institusi: Pembandingan Kinerja *Web server* Apache dengan Nginx," *Multinetics*, vol. 1, no. 2, hal. 12, 2018.

[6] V. N. Nguyen, "Comparative Performance Evaluation of *Web server*s," vol. 31, no. 3, hal. 28–34, 2017.

[7] A. Goel, N. Bansal, dan S. Gupta, "Comparison of Different *Web server*s," *Imp. J. Interdiscip. Res.*, vol. 2, no. 12, hal. 2454–1362, 2016.

[8] R. Dawood, S. F. Qiana, dan S. Muchallil, "Kelayakan Raspberry Pi sebagai *Web server*: Perbandingan Kinerja Nginx, Apache, dan Lighttpd pada *Platform* Raspberry Pi," *J. Rekayasa Elektr.*, vol. 11, no. 1, hal. 25–29, 2015.

[9] P. Prakash, R. Biju, dan M. Kamath, "Performance analysis of process driven and event driven *web server*s," *Proc. 2015 IEEE 9th Int. Conf. Intell. Syst. Control. ISCO 2015*, hal. 1–7, 2015.

[10] Q. Fan dan Q. Wang, "Performance comparison of *web server*s with different architectures: A case study using high concurrency work*load*," *Proc. - 3rd Work. Hot Top. Web Syst. Technol. HotWeb 2015*, no. Section III, hal. 37–42, 2016.

[11] Minakshi dan H. Sharma, "Web Technology Http *Web server* or Web Services," *Int. J. Res.*, vol. 1, no. 10, hal. 1585–1590, 2014.

[12] A. Low, J. Siu, I. Ho, dan G. Liu, "Introduction to Node.Js," in *Proceedings of 24th Annual International Conference on Komputer Science and Software Engineering*, 2014, hal. 283–284.

[13] X. Chi, B. Liu, Q. Niu, dan Q. Wu, "Web *load* balance and *cache* optimization design based nginx under high-concurrency environment," *Proc. - 2012 3rd Int. Conf. Digit. Manuf. Autom. ICDMA 2012*, hal. 1029–1032, 2012.

[14] L. P. Chitra dan R. Satapathy, "Performance comparison and evaluation of Node.js and traditional *web server* (IIS)," *2017 Int. Conf. Algorithms, Methodol. Model. Appl. Emerg. Technol. ICAMMAET 2017*, vol. 2017-Janua, hal. 1–4, 2017.

[15] "The Go Programming Language." [Daring]. Tersedia pada: https://golang.org/.

[16] S. Varghese, "Web Development with Go," *Web Dev. with Go*, 2015.

[17] G. Fedrecheski, L. C. P. Costa, dan M. K. Zuffo, "Elixir programming language evaluation for IoT," *Proc. Int. Symp. Consum. Electron. ISCE*, hal. 105–106, 2016.

[18] H. Takase, K. Matsui, Y. Ueno, M. Mori, Y. Hisae, dan S. Yamazaki, "Work-in-Progress: A concept of a hardware design environment with the functional language Elixir," in *2019 International Conference on Hardware/Software Codesign and Sistem Synthesis (CODES+ISSS)*, 2019, hal. 1–2.