Analisa Web Server Untuk Kebutuhan Open Journal System Menggunakan Secure Tunnel

Web Server Analysis for Open Journal System Needs Using Secure Tunnel

Yudhi Arta[1], Rizky Wandri[2], Anggi Hanafiah[3], Bima Kristian Pranoto[4], M Rizki Fadhilah[5]
[1,2]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Riau
e-mail: [1]**yudhiarta@eng.uir.ac.id** , [2]rizkywandri@eng.uir.ac.id , [3]anggihanafiah@eng.uir.ac.id , [4]bimakristian@student.uir.ac.id , [5]mrizkifadhilah@eng.uir.ac.id

Abstrak

Jurnal digital (e-journal) melalui Open Journal System (OJS) menjadi salah satu sarana dalam mempublikasikan hasil penelitian pada lingkup yang lebih luas.Server adalah tempat untuk menyimpan konten website atau sering juga disebut dengan istilah hosting. Tanpa adanya server, maka sebuah website tidak bisa diakses. Open Journal System diakses oleh banyak pengunjung dan memiliki traffic yang tinggi, sehingga dibutuhkan sebuah web server yang andal dan optimal, Penggunaan web server yang tepat, membantu dalam publikasi jurnal ilmiah dan kurangnya informasi web server yang memiliki ketahanan dengan beban request yang tinggi, juga meliputi banyaknya jumlah user yang mengakses web server tersebut dalam hitungan hari hingga perbulan. Banyak web server yang ada pada saat ini, contohnya yang populer adalah Apache dan Nginx. Dua web server ini merupakan web server yang banyak digunakan saat ini oleh banyak website di seluruh dunia. Penelitian ini dilakukan untuk menguji dan menganalisa kinerja dari web server untuk kebutuhan OJS. Untuk metode pengujiannya adalah penerapan metode stress test. Metode ini berguna untuk melihat throughput, response time, sent, received, dan error. Pengujian selanjutnya pada sejumlah user, dan pengujian waktu yang dibutuhkan untuk mengakses sebuah web server.Pengujiannya menggunakan user yang dimulai dari 500 user, 750 user dan 1000 user. Dari pengujian yang dilakukan terhadap beberapa parameter, website Open Journal System yang menggunakan web server Apache bekerja lebih optimal daripada website Open Journal Sytstem yang menggunakan web server Nginx sehingga saat diakses menggunakan secure tunnel, web Open Journal System (OJS) gagal memuat file header dan footer sehingga tampilan website jadi kurang menarik dan tidak user friendly.

Kata kunci—Open Journal System, Server, Apache, Nginx, Traffic

Abstract

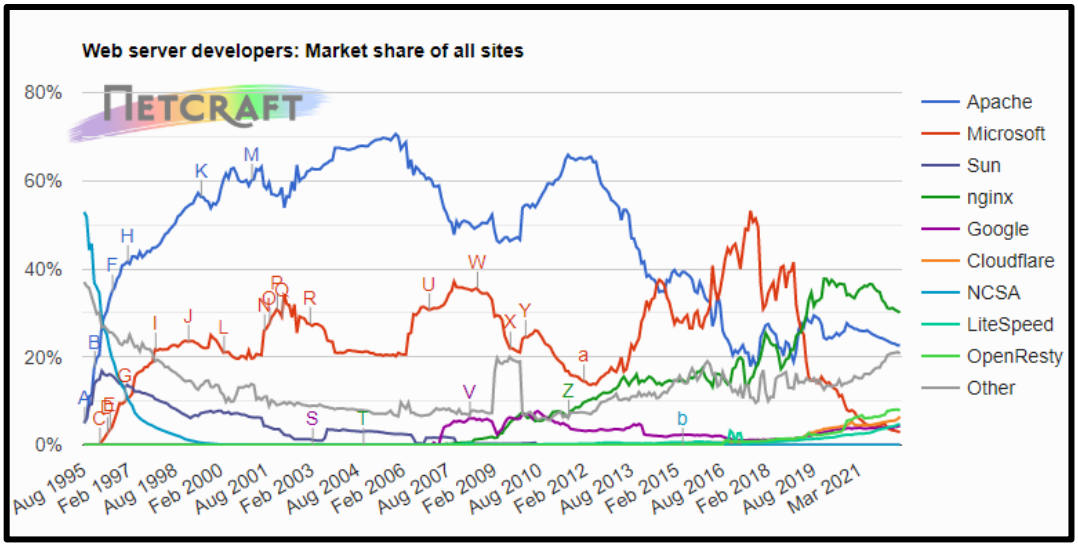
Digital journals (e-journals) through the Open Journal System (OJS) are a means of publishing research results on a broader scope. Servers are places for storing website content or often also referred to as hosting. Without a server, a website cannot be accessed. The Open Journal System is accessed by many visitors and has high traffic, so a reliable and optimal web server is needed. The use of the right web server helps in publishing scientific journals and the lack of information on a web server that has resilience with high request loads, also includes the large number of users accessing the web server in a matter of days to a month. Many web servers exist today, popular examples are Apache and Nginx. These two web servers are web servers that are widely used today by many websites around the world. This research was conducted to test and analyze the performance of web servers for OJS needs. For the test method is the application of the stress test method. This method is useful for viewing throughput, response time, sent,

received, and error. Further testing on a number of users, and testing the time needed to access a web server. The test uses users starting from 500 users, 750 users and 1000 users. From the tests carried out on several parameters, the Open Journal System website that uses the Apache web server works more optimally than the Open Journal System website that uses the Nginx web server so that when accessed using a secure tunnel, the Open Journal System (OJS) web fails to load header and footer files. so that the appearance of the website becomes less attractive and not user friendly.

Keywords—Open Journal System, Server, Apache, Nginx, Traffic

1 PENDAHULUAN

Open Journal System (OJS) merupakan perangkat lunak yang bersifat open source. OJS merupakan sarana publikasi karya ilmiah dan merupakan salah satu aplikasi website yang banyak digunakan. Sebagai salah satu website yang banyak dikunjungi oleh pembaca atau memiliki traffic yang tinggi, OJS harus memiliki server yang mumpuni. Berdasarkan survei yang dilakukan oleh situs Netcraft pada bulan Juli 2022, Nginx menjadi web server yang banyak digunakan saat ini. Untuk lebih jelasnya dapat dilihat pada gambar 1 di bawah ini.



Gambar 1 Informasi Web Server Netcraft [1]

Bagi beberapa pengguna web server saat ini, Apache dan Nginx merupakan web server yang selalu menjadi pilihan utama. Berdasarkan survei penggunaan web server yang dilakukan oleh Netcraft terhadap 1.139.467.659 situs di 271.728.559 domain unik dan 12.341.172 komputer yang terhubung ke website menyatakan bahwa web server Nginx digunakan sebanyak 343.354.785 situs website atau sekitar 30,13%, sedangkan web server Apache digunakan sebanyak 258.219.193 situs website atau sekitar 22,66%. Dari jumlah pengguna, web server Nginx unggul dari web server Apache.

Berdasarkan latar belakang tersebut maka dapat diidentifikasi masalah sebagai berikut bahwa Open Journal System diakses oleh banyak pengunjung dan memiliki traffic yang tinggi, sehingga dibutuhkan sebuah web server yang andal dan optimal, Penggunaan web server yang tepat, membantu dalam publikasi jurnal ilmiah dan kurangnya informasi web server yang memiliki ketahanan dengan beban request yang tinggi, juga meliputi banyaknya jumlah user yang mengakses web server tersebut dalam hitungan hari hingga perbulan.

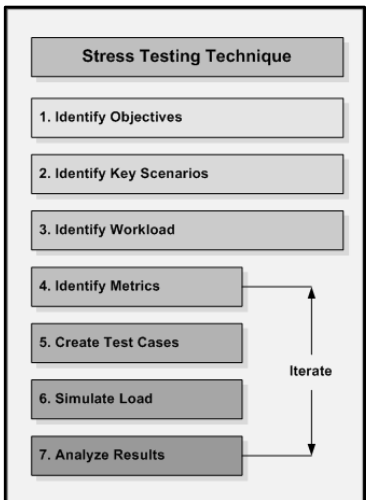
Berdasarkan penelitian yang dilakukan oleh Rasma Bayu Krisnandar dalam penelitian "Analisis Perbandingan Kinerja Web Server Nginx, Apache, & LIGHTTPD Dengan Metode

Stress Test" menyatakan bahwa web server Nginx mempunyai kelebihan dalam kecepatan menanggapi request dari klien serta lebih hemat dalam penggunaan RAM, namun mempunyai kelemahan pada jumlah transfer data yang sedikit lebih kecil serta penggunaan daya CPU yang tinggi. Web Server Apache punya kelebihan dalam jumlah transfer data yang besar serta penggunaan daya CPU yang kecil, namun memiliki kelemahan yakni lambat dalam menanggapi request dari klien serta penggunaan daya RAM yang tinggi [2]. Penelitian yang dilakukan oleh Andhika Dwitama Putra dalam penelitiannya "Analisis Kinerja Dan Konsumsi Sumber Daya Aplikasi Web Server Pada Platform Raspberry Pi" menyatakan bahwa web server Nginx memiliki kinerja dan konsumsi daya yang lebih baik jika dibandingkan dengan web server Apache dalam melayani halaman statis [3]. Penelitian yang dilakukan oleh Intan Ferina Irza "Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux.com)" menyatakan bahwa web server Apache memiliki throughput yang lebih unggul daripada web server Nginx, di mana bandwith yang dihasilkan jauh lebih baik sehingga mampu menampung banyak data. Web server Nginx lebih unggul dalam mengkoneksikan klien ke server dan server ke klien di mana semakin kecil koneksi yang ditunjukan semakin bagus kinerja dari sebuah web server. Web server Apache dan Nginx memiliki kinerja yang hampir sama dalam menangani *request*, tergantung dari kemampuan klien yang mengakses web server itu. Dari segi *reply section*, web server Nginx lebih baik daripada web server Apache karena *reply time* nya lebih kecil, sehingga semakin kecil waktu *reply time* semakin cepat data yang bisa ditransfer. Web server Apache dan web server Nginx memiliki kinerja yang sama baiknya karena tidak terjadinya kesalahan selama pengujian berlangsung[4]–[6].

2 METODE PENELITIAN

2.1 Stress Testing

Stress Testing adalah metode pengujian kinerja yang berfokus pada ketahanan, ketersediaan dan keandalan dari aplikasi pada kondisi yang ekstrem. Kondisi ekstrem yang dimaksud di sini adalah *heavy loads, high concurrency, atau limited computational resources*. Tujan dari stres testing ialah untuk mengidentifikasi masalah aplikasi yang muncul. *Stress testing* yang baik juga membantu dalam menemukan bug terkait sinkronisasi, *interlock problems, priority problems*, dan *resource loss bugs*. Stress test biasanya melibatkan satu atau lebih simulasi dari *production scenarios* yang menggunakan variasi dari *stressful conditions*[5]–[8]. Secara umum, terdapat 7 tahapan dalam melakukan stres testing pada gambar 2 dibawah ini :



Gambar 2 Tahapan pada Stres Testing

Langkah 1 - Identifikasi Test Objectives
Tes objectives ini adalah menguji beberapa objektiritas dari sebuah aplikasi. Identifikasi bertujuan membuat beberapa pertanyaan yang anntinya akan kita gunakan sebagai bahan ouput dari penelitian ini. Contoh beberapa pertanyaan yang diajukan diantaranya adalah :

1. Bagaimana pengujiandari server apache dan nginx dalam menangani bebna server yang besar ?

Langkah 2 - Identifikasi Key Scenario
Identifikasi ini digunakan untuk mengidentifikasi masalah potensial. Dalam mendapatkan hadil yang terbaik, test tetap harus fokus pada mekanisme skenario yang mencakup *overall* dari sebuah aplikasi[9], [10] Untuk mendapatkan mekanisme skenario yang terbaik adalah sebagai berikut :

1. Percobaan untuk tetap melakukan skenario yang akan mempengaruhi performa aplikasi dan lainnya. Dalam hal ini biasanya seperti *scenario intensive locking* and *synchronization, disk-intensive input/output* (I/O) *operations dan long transactions*.

Langkah 3 - Identifikasi Workload
Identifikasi workload adalah analisa yang dilakukan untuk melihat seberapa handal dan jauh bagaimana sebuah web server dalam menangani beberapa permintaan yang melebihi batas maksimum, sehingga peneliti dapat mencari dan menemukan berbagai solusi dalam skema pengujian yang akan dilakukan.

Langkah 4 - Identifikasi Metrik
Yang akan dikumpulkan terkait performa dari aplikasi ini adalah identifikasi ukuran/metrik. Peneliti nantinya mengumpulkan beberapa informasi dari beberapa sebuah server sehingga dalam penggunaan pemakaian RAM (*Random Access Memory*) dan CPU (*Central Processing Unit*) dan melayani beberapa request[11]–[13].

Langkah 5 - Membuat Tes Cases
Membuat test yang sudah didefinisikan untuk melakukan percobaan tes dengan disertakan hasil yang diharapkan. Dalam tes ini dibutuhkan tes design untuk memberikan hasil yang terkait pada *expected results*.

Langkah 6 - Simulasi Load
Untuk simulasi ini menggunakan *test tools* untuk setiap tes dan catatan hasil dari data metrik[14], [15]. Adapun langkahnya sebagai berikut :

1. Memvalidasi setiap tes dan harus sesuai dengan yang didesain.
2. Memastikan uji coba environment yang terkonfigurasi.
3. Melakukan tes sebelum menjalankan beberapa tes lainnya untuk memastikan skrip bekerja dengan benar dan baik.
4. Mereset system dan melakukan tes.

Langkah 7 - Analisa hasil
Dalam hasil Analisa ini yang dijadikan Analisa adalah metric dari yang sudah dicatat. Jika dalam pengujian hasil yang diindikasikan bahwa performance yang diminta belum dapat dicapai, maka peneliti melakukan analisa serta memperbaiki issue. Melakukan beberapa review design dan review kode untuk mempermudah peneliti dalam memperbaiki issue tersebut.

Skenario Pengujian

Dalam analisa ini melakukan perbandingan kinerja Apache dan Nginx untuk kebutuhan Open Journal System (OJS) dengan menggunakan aplikasi Apache JMeter. Pengujan kinerja yang dilakukan melalui beberapa langkah yang nantinya akan menjadi prosedur dari penelitian ini. Adapun beberapa tahapan yang akan dilakukan adalah sebagai berikut :

1. Skema Rancangan Tahapan Pengujian

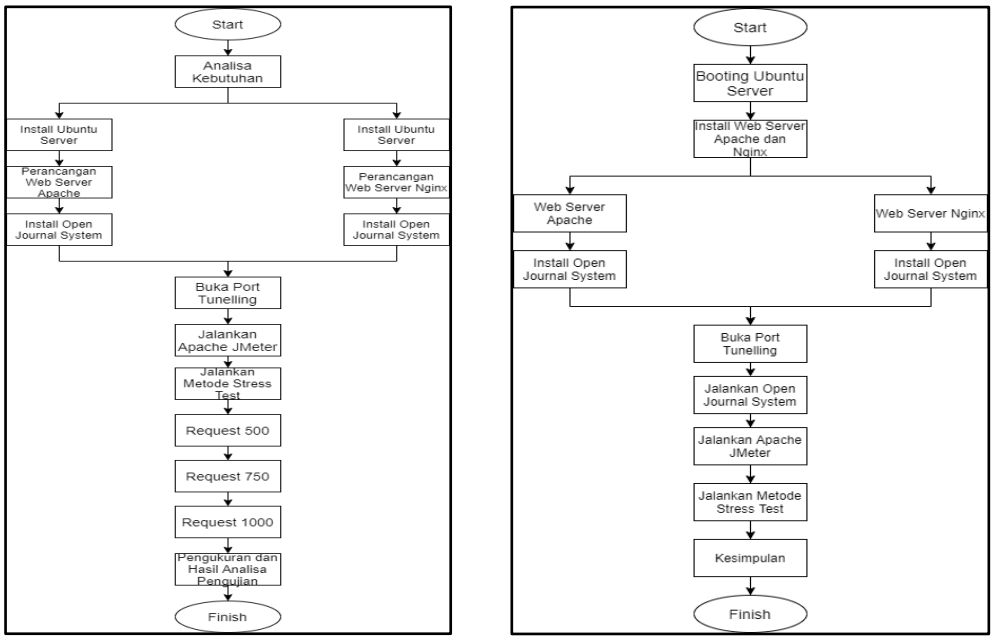
Pengukuran ini dilakukan dengan cara mengambil nilai average dari beberapa pengujian. Pengujian yang dilakukan pada website Open Journal System (OJS) beserta dengan beban yang diberikan yakni 500, 750, dan 1000 request.

Pembahasan dalam hasil penelitian ini dan pengujian yang diperoleh dapat disajikan dalam bentuk uraian teoritik, baik secara kualitatif atau kuantitatif. Beberapa hasil percobaannya sebaiknya dapat ditampilkan dalam berupa grafik atau tabel.

2. Skema Rancangan Alur Pengujian pada web server

Pada gambar di bawah ini menjelaskan skema pengujian dari sebuah web server. Kemudian, tahapan instalasi software Open Journal System (OJS) pada web server yang akan diuji. Akses Ngrok untuk membuka port tunelling, lalu jalankan Open Journal System (OJS).

Selanjutnya Apache JMeter diinstalasi pada sisi client serta menjalankan metode pengujiannya. Dan selanjutnya membuat kesimpulan dari hasil pengujian untuk mendapatkan perbandingan dari web server Apache dan Nginx. Untuk 2 skema diatas dapat lihat pada gambar 3 dibawah ini :

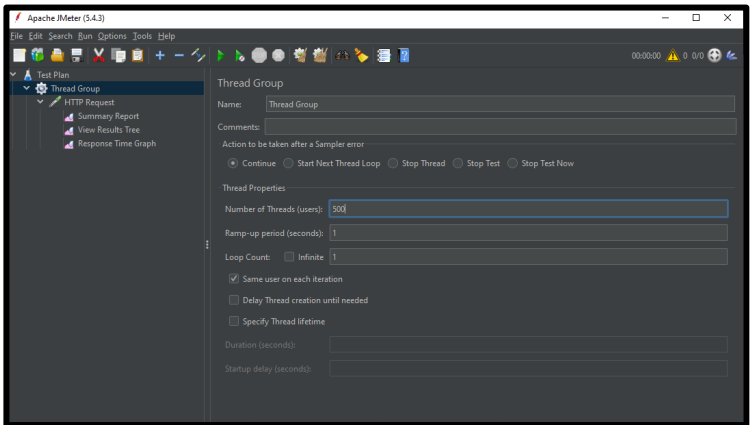


(a) (b)

Gambar 3 (a) Skema Tahapan Penelitian (b) Skema Pengujian Web Server

3 HASIL DAN PEMBAHASAN

Banyak hal yang bisa dilakukan pada aplikasi Apache JMeter. Salah satunya adalah penerapan metode stress test. Metode ini berguna untuk melihat throughput, response time, sent, received, dan error. Pengujian selanjutnya pada sejumlah user, dan pengujian waktu yang dibutuhkan untuk mengakses sebuah web server. Pada gambar 4 di bawah ini merupakan tampilan aplikasi Apache JMeter untuk pengujian 500 user dalam 1 second/detik.



Gambar 4 Apache JMeter dengan 500 user dalam 1 detik

Untuk metode pengujiannya, kemudian peneliti melakukan pengujian dengan jumlah yang sama namun dengan pengujian waktu berbeda. Waktu yang digunakan adalah 5 detik. Peneliti selesai dengan melakukan pengujian terhadap 500 *user* dalam waktu 1 hingga 5 detik, lakukan pengujian terhadap 750 *user* dan 1000 *user* dalam waktu 1 detik. Kemudian peneliti melakukan pengujian seperti sebelumnya dengan waktu berbeda. Dan kini waktu yang digunakan adalah 5 detik.

Pengujian Kinerja Web server

Pengujian untuk kinerja web server ini dilakukan dengan metode secara bergantian dan mendahulukan pengujian terhadap kinerja web server Apache, dilanjutkan kemudian melakukan pengujian terhadap web server Nginx dengan metode stress test pada Apache JMeter.

Pengujian web server dilakukan dengan mengakses halaman index dari website Open Journal System (OJS) menggunakan secure tunnel. Di bawah ini adalah tampilan dari halaman index website Open Journal System (OJS) yang diakses menggunakan secure tunnel

Gambar 5 Halaman index OJS yang diakses menggunakan secure tunnel

Pada gambar 5 di atas, dapat dilihat bahwa halaman index dari webiste Open Journal System (OJS) yang diakses menggunakan secure tunnel tidak terdapat header dan footer. Hal ini disebabkan karena secure tunnel gagal memuat header dan footer dari Open Journal System (OJS).

Hasil dari pengujian website Open Journal System (OJS) untuk web server Apache disajikan dalam tabel 1 di bawah ini yang meliputi *throughput, parameter error, sent*, dan *received.*

Tabel 1 Hasil pengujian OJS pada web server Apache

Jumlah Pengujian	Error (%)	Throughput (Sec)	Received (KB/Sec)	Sent (KB/Sec)
500 *users* 1 *second*	2,6	5,2	39,24	3,84
500 *users* 5 *second*	2	5,2	39,3	3,85
750 *users* 1 *second*	2,53	8,8	65,81	6,44
750 *users* 5 *second*	2,8	10,9	82,18	8,04
1000 *users* 1 *second*	5	11,5	85,21	8,3
1000 *users* 5 *second*	17,3	10,7	74,85	7,23

Hasil dari pengujian website Open Journal System (OJS) untuk web server Nginx disajikan dalam tabel 2 di bawah ini yang meliputi throughput, parameter error, sent, dan received.

Tabel 2 Hasil pengujian OJS pada web server Nginx

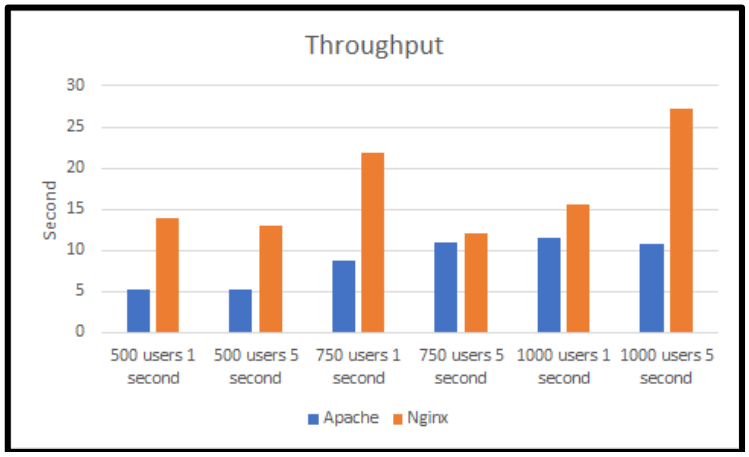
Jumlah Pengujian	Error (%)	Throughput (Sec)	Received (KB/Sec)	Sent (KB/Sec)
500 *users* 1 *second*	28,2	13,9	83,69	8,57
500 *users* 5 *second*	23	13	80,9	8,42
750 *users* 1 *second*	49,07	21,8	93,44	11,79
750 *users* 5 *second*	49,2	12	51,66	6,51
1000 *users* 1 *second*	61,9	15,5	55,61	7,33
1000 *users* 5 *second*	61,7	27,3	93,2	13,53

Hasil Pengujian Web Server

Hasil pengujian dari Apache dan Nginx pada website Open Journal System yang diakses melalui secure tunnel meliputi beberapa parameter yakni throughput, response time, sent, received, eror, dan log. Hasil dari pengujian kinerja tersebut adalah sebagai berikut:

1. Throughput

Throughput menghitung jumlah permintaan yang akan diproses dalam satuan waktu yang telah ditentukan oleh setiap server. Waktu tersebut akan dihitung mengikuti pengujian pertama hingga pengujian terakhir. Dengan catatan semakin besar nilai dari throughput, maka semakin bagus pula kinerja dari sebuah web server.

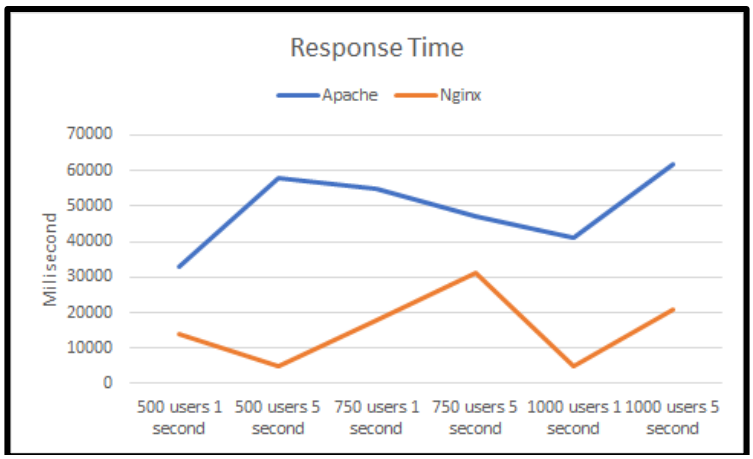


Gambar 6 Grafik hasil pengujian throughput

Dari gambar 6 di atas hasil dari pengujian throughput, web server Nginx mendapatkan nilai yang lebih baik daripada server Apache. Itu artinya, web server Nginx memiliki jumlah transfer data yang lebih banyak ketimbang web server Apache.

2. Response Time

Response time merupakan waktu yang diberikan oleh interface, ketika client melakukan beberapa request atau mengirim beberapa permintaan ke server. Pada aplikasi ini sudah tersedia beberapa fitur untuk melihat hasil dari response time yang berguna untuk menghitung waktu response. Dan tampilan grafik untuk melihat hasil dari waktu respon sebuah web server. Di bawah ini adalah grafik response time dari server Apache dan Nginx.

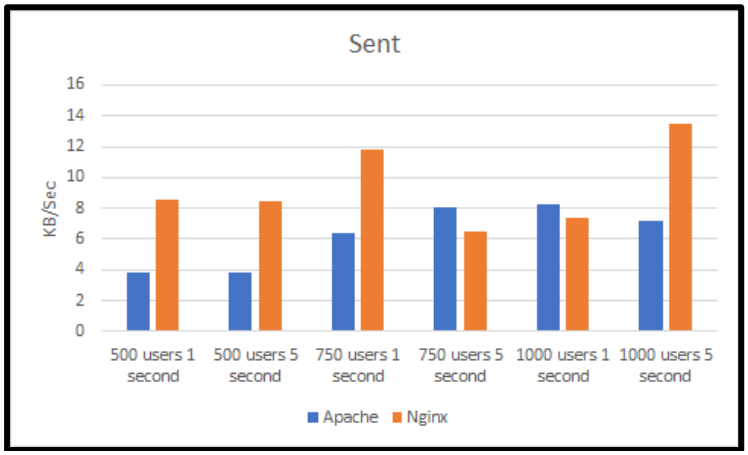


Gambar 7 Grafik Response Time Apache Dan Nginx

Pada gambar 7 di atas, Nginx mendapatkan nilai yang lebih baik pada response time daripada Apache. Artinya, web Nginx dalam menanggapi interaksi lebih responsif antara server dengan client.

3. Sent

Sent merupakan pengujian yang dikirim dari sebuah server hingga menuju client. Jika semakin kecil nilai hasilnya, maka kinerja dari sebuah web server semakin baik. Di bawah ini adalah grafik hasil pengujian parameter sent pada web server Apache dan Nginx.

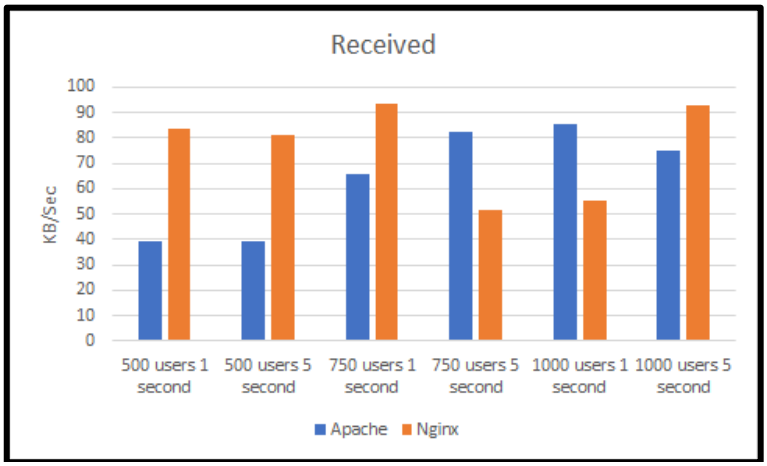


Gambar 8 Grafik Hasil Pengujian Sent

Berdasarkan gambar 8 di atas, didapatkan bahwa nilai paramater sent pada web server Apache lebih kecil jika dibandingan dengan nilai parameter sent pada web server Nginx. Nilai paramater sent pada web server Apache lebih besar daripada nilai parameter sent pada web server Nginx terjadi pada pengujian sampel 750 users dalam waktu 5 seconds dan pengujian sampel 1000 users dalam waktu 1 second. Pada pengujian sampel dan waktu yang lainnya, nilai parameter sent pada web server Apache lebih kecil jika dibandingkan dengan hasil dari web server Nginx. Maka, jika melihat dari parameter ini bahwa web server Apache memiliki kinerja yang lebih baik dibandingkan dengan web server Nginx.

4. Received

Received merupakan pengujian yang diterima dari sisi client hingga menuju server. Jika semakin kecil nilainya, maka kinerja dari sebuah web server semakin baik. Di bawah ini adalah grafik hasil pengujian parameter received pada web server Apache dan Nginx.



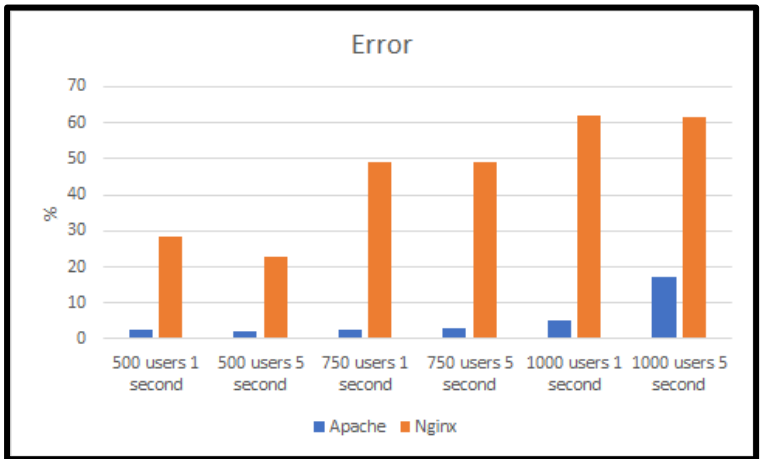
Gambar 9 Grafik Hasil Pengujian Received

Berdasarkan gambar 9 di atas, didapatkan bahwa nilai paramater received pada web server Apache lebih kecil jika dibandingkan dengan nilai parameter received pada web server Nginx. Nilai paramater received pada web server Apache lebih besar daripada nilai parameter received pada web server Nginx terjadi pada pengujian sampel 750 users dalam waktu 5 seconds dan pengujian sampel 1000 users dalam waktu 1 second.

Pada pengujian sampel dan waktu yang lainnya, nilai parameter received pada web server Apache lebih kecil jika dibandingkan dengan web server Nginx. Maka, jika melihat dari parameter ini bahwa web server Apache memiliki kinerja yang lebih baik dibandingkan dengan web server Nginx.

5. Error

Error merupakan pengujian saat client mengakses sebuah halaman web. Jika nilainya besar pada error pada web server, maka akan semakin tidak baik kinerja dari web server tersebut. Di bawah ini adalah grafik hasil pengujian parameter error pada web server Apache dan Nginx.



Gambar 10 Grafik Hasil Pengujian Error

Pada gambar 10 di atas, didapatkan bahwa nilai paramater error pada web server Apache lebih kecil jika dibandingkan dengan nilai parameter error pada web server Nginx. Maka, jika melihat dari parameter ini bahwa web server Apache memiliki kinerja yang lebih baik dibandingkan dengan web server Nginx.

Pembahasan

Dalam pembahasan ini, berdasarkan dari pengujian dilakukan peneliti pada Apache dan Nginx, maka akan didapatkan hasil sebagai berikut:

1. Dalam pemilihan sebuah web server yang dapat menangani sejumlah permintaan yang tinggi, baik dari satu waktu serta lebih, maka responsif saat dalam menanggapi interaksi antara client dengan server, web server Nginx bisa dijadikan pilihan yang tepat. Sebab, web server Nginx unggul dari web server Apache dalam pengujian throughput dan response time.
2. Dalam pemilihan sebuah web server dalam memanajemen bandwith dan pengiriman data dari user, Apache menjadi pilihan yang tepat untuk dipilih. Karena web server Apache unggul dari web server Nginx dalam pengujian sent, received, dan error.
3. Dampak yang dihasilkan dari penelitian ini bisa membantu para administrator server dalam mementukan web server apa yang akan dipakai untuk memenuhi kebutuhan dalam menangani OJS sehingga dapat berpengaruh positif atas kinerja server tersebut.

4 KESIMPULAN

Berdasarkan dari hasil pengujian yang sudah dilakukan oleh peneliti, maka dapat diambil kesimpulan :

1. Untuk pengujian *throughput*, web server Nginx bisa dikatakan lebih baik dari Apache. Web Server Nginx juga dapat melakukan manajemen dari beberapa user yang dapat kita lihat pada gambar 4, sehingga jumlah request bandwith pada sebuah server yang diproses dapat diproses lebih baik.
2. Saat pengujian response time, web server Nginx juga lebih baik dibandingkan web server Apache. Web server Nginx lebih responsif jika sebuah user melakukan interaksi dengan server.
3. Saat mencoba pengujian yang lain seperti sent, untuk Apache lebih baik dari Nginx. Dikarenakan ketika apache mengelola sebuah pengiriman data yang direquest dapat mengelolanya lebih baiki dibandingkan Nginx.
4. Keterbatasan user dalam melakukan pengujian berdasarkan user yang di uji menjadi salah indicator penting dalam penelitian ini.

5 UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Islam Riau yang telah memberi dukungan financial terhadap penelitian ini.

DAFTAR PUSTAKA

[1] Netcraft, “July 2022 Web Server Survey | Netcraft News.” [Online]. Available: https://news.netcraft.com/archives/2022/07/28/july-2022-web-server-survey.html. [Accessed: 03-Oct-2022].

[2] R. Kisnandar, “Analisis Perbandingan Kinerja Web Server Nginx, Apache, Dan Lighttpd Dengan Metode Stress Test.” STMIK AKAKOM Yogyakarta, 2019.

[3] A. D. Putra, W. Yahya, and A. Bhawiyuga, “Analisis Kinerja Dan Konsumsi Sumber Daya Aplikasi Web Server Pada Platform Raspberry Pi,” *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2019.

[4] I. F. Irza, Z. Zulhendra, and E. Efrizon, “Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux. com),” *Voteteknika (Vocational Tek. Elektron. dan Inform.*, vol. 5, no. 2, 2017.

[5] I. Satwika and K. N. Semadi, “Perbandingan Performansi Web Server Apache Dan Nginx Dengan Menggunakan Ipv6,” *SCAN-Jurnal Teknol. Inf. dan Komun.*, vol. 15, no. 1, pp. 10–15, 2020.

[6] A. Ridwan, “Analisis Perbandingan Performa Apache Web Server Dan Nginx Menggunakan Apache Jmeter,” *J. Teknoif Tek. Inform. Inst. Teknol. Padang*, vol. 8, no. 2, pp. 87–92, 2020.

[7] D. DeJonghe, *Nginx CookBook*. O’Reilly Media, 2020.

[8] S. D. Riskiono and D. Pasha, “Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E-Learning,” *InComTech J. Telekomun. Dan Komput.*, vol. 10, no. 3, pp. 135–144, 2020.

[9] A. Tsakyridis, T. Alexoudi, A. Miliou, N. Pleros, and C. Vagionas, "10 Gb/s optical random access memory (RAM) cell," *Opt. Lett.*, vol. 44, no. 7, pp. 1821–1824, 2019.

[10] X. Chen *et al.*, "Sky-RAM: Skyrmionic random access memory," *IEEE Electron Device Lett.*, vol. 40, no. 5, pp. 722–725, 2019.

[11] N. Harki, A. Ahmed, and L. Haji, "CPU scheduling techniques: A review on novel approaches strategy and performance assessment," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 48–55, 2020.

[12] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 463–479.

[13] F. Zhang, L. Yang, S. Zhang, B. He, W. Lu, and X. Du, "{FineStream}:{Fine-Grained}{Window-Based} Stream Processing on {CPU-GPU} Integrated Architectures," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 633–647.

[14] J. C. Phillips *et al.*, "Scalable molecular dynamics on CPU and GPU architectures with NAMD," *J. Chem. Phys.*, vol. 153, no. 4, p. 44130, 2020.

[15] A. J. Sanchez-Fernandez *et al.*, "Asynchronous processing for latent fingerprint identification on heterogeneous CPU-GPU systems," *IEEE Access*, vol. 8, pp. 124236–124253, 2020.