OPTIMASI *APACHE WEB SERVER* MENGGUNAKAN *VARNISH WEB CACHE DAN REVERSE PROXY NGINX*

Skripsi

disusun sebagai salah satu syarat

untuk memperoleh gelar Sarjana Komputer

Program Studi Teknik Informatika

Oleh

Anindya Putri Arunawati
4611416030

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2020

PERNYATAAN

Saya menyatakan bahwa skripsi saya yang berjudul "Optimasi *Apache Web Server* Menggunakan *Varnish Web Cache Dan Reverse Proxy Nginx*" disusun atas dasar penelitian saya dengan arahan dosen pembimbing. Sumber informasi atau kutipan yang berasal dari karya yang diterbitkan telah disebutkan dalam teks dan dicantumkan dalam daftar pustaka di bagian akhir skripsi ini. Dan saya menyatakan bahwa skripsi ini bebas plagiat dan apabila di kemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya bersedia menerima sanksi sesuai ketentuan perundang-undangan.

Semarang, 11 Februari 2020

Anindya Putri Arunawati
4611416030

PERSETUJUAN PEMBIMBING

Nama : Anindya Putri Arunawati

NIM : 4611416030

Program Studi : Teknik Informatika S1

Judul Skripsi : Optimasi *Apache Web Server* Menggunakan *Varnish Web Cache* dan *Reverse Proxy Nginx*

Skripsi ini telah disetujui oleh pembimbing untuk diajukan ke sidang panitia ujian skripsi Program Studi Teknik Informatika FMIPA UNNES.

Semarang, 11 Februari 2020

Pembimbing

Much Aziz Muslim, S.Kom., M.Kom.
NIP 197404202008121001

PENGESAHAN

Skripsi yang berjudul

Optimasi *Apache Web Server* Menggunakan *Varnish Web Cache* dan *Reverse Proxy Nginx*

disusun oleh

Anindya Putri Arunawati

4611416030

Telah dipertahankan di hadapan sidang Panitia Ujian Skripsi FMIPA UNNES pada tanggal 24 Februari 2020.

Panitia:

Ketua

Dr. Sugianto, M.Si.
NIP 196102191993031001

Sekretaris

Dr. Alamsyah, S.Si., M.Kom.
NIP 197405172006041001

Penguji 1

Anggyi Trisnawan Putra, S.Si., M.Si.
NIP 198707062014041003

Penguji 2

Aji Purwinarko, S.Si., M.Cs.
NIP 198509102015041001

Anggota Penguji/
Pembimbing

Much Aziz Muslim, S.Kom., M.Kom.
NIP 197404202008121001

MOTTO DAN PERSEMBAHAN

MOTTO

- Jika yang dapat kau lakukan hanyalah merangkak, maka mulailah merangkak (Rumi)
- Semakin dia yakin kepada Allah, semakin sedikit dia mengeluh dalam hidupnya (ust. Hanan Attaki)
- Jangan pernah lelah untuk berdoa dan percayakan semuanya pada Allah, Allah yang mengatur segalanya (Anindya)

PERSEMBAHAN

Skripsi ini saya persembahkan kepada:

- Kedua Orang Tua saya Bapak Achmad Daserun dan Ibu Watini yang telah mencurahkan keringatnya untuk membiayai pendidikan saya, yang selalu memberikan kasih sayang, doa, dan dukungannnya.
- Adik saya, Hilal Hubdin yang telah memberikan dukungan serta doa yang terus dipanjatkan.
- Teman-teman saya di jurusan Ilmu Komputer, Fakultas MIPA, serta teman-teman di Universitas Negeri Semarang.
- Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu hingga terselesaikannya penulisan skripsi ini.
- Almamater, Universitas Negeri Semarang.

PRAKATA

Puji syukur penulis panjatkan kepada Allah *Subhanahu wa ta'ala* atas berkat rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi yang berjudul "**Optimasi *Apache Web Server* Menggunakan *Varnish Web Cache Dan Reverse Proxy Nginx***".

Penulis menyadari bahwa penulisan skripsi ini tidak akan selesai tanpa adanya dukungan serta bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Prof. Dr. Fathur Rokhman, M.Hum., Rektor Universitas Negeri Semarang.
2. Dr. Sugianto M.Si., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Dr. Alamsyah S.Si., M.Kom., Ketua Jurusan Ilmu Komputer FMIPA Universitas Negeri Semarang yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.
4. Much Aziz Muslim, S.Kom., M.Kom., Dosen Pembimbing yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.
5. Bapak dan Ibu Dosen Jurusan Ilmu Komputer yang telah memberikan bekal kepada penulis dalam penyusunan skripsi ini.
6. Kedua Orang Tua saya Bapak Achmad Daserun dan Ibu Watini yang telah mencurahkan keringatnya untuk membiayai pendidikan saya, yang selalu memberikan kasih sayang, doa, dan dukungannnya.

7. Adik saya, Hilal Hubdin yang telah memberikan dukungan serta doa yang terus dipanjatkan.
8. Teman-teman seperjuangan saya, Tanzilal Mustaqim, Ilham Esa Tiffani, Anisa Falasari, Diah Alifia E, Apriani S, Novia Puji, Hansa Nuha dan Sulistiana sebagai teman diskusi yang banyak memberikan dukungan dan motivasi.
9. Teman-teman saya di jurusan Ilmu Komputer, terutama teman-teman ilkom angkatan 2016 dan teman-teman kontrakan saya yang telah memberikan semangat dan dukungannya.
10. Sahabat-Sahabat saya, Ahmad Dzubayyan, Yustica Septyaningtyas, Ahmad Bakrie, Setiawan Rifki dan Agustin Triningsih sebagai teman diskusi yang banyak memberikan dukungan dan motivasi
11. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu persatu, terimakasih atas bantuannya.

Semoga skripsi ini dapat memberikan manfaat bagi pembaca di masa yang akan datang.

Semarang, 11 Februari 2020
Penulis

Anindya Putri Arunawati
4611416030

ABSTRAK

Anindya Putri Arunawati. 2020. Optimasi *Apache Web Sever* Menggunakan *Varnish Web Cache* dan Reverse Proxy Nginx. Skripsi, Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Much Aziz Muslim, S.Kom., M.Kom.

Kata kunci: *Optimasi, Apache, Varnish, Nginx, Apache Benchmark.*

Web server adalah suatu *software* yang berfungsi untuk melayani permintaan dari browser kemudian mengirimkan kembali hasil permintaan dalam bentuk halaman situs *web*. *Apache* merupakan *web server* yang paling banyak digunakan, sejalan dengan itu pula beban kerja *apache web server* yang memberikan *service* di internet mengalami kenaikan yang cukup signifikan karenanya pada saat permintaan yang masuk berjumlah besar *apache* menjadi lamban. Penelitian ini berfokus pada bagaimana mengoptimasi kinerja dari *apache web server* menggunakan *reverse proxy nginx* dan *web cache varnish*. Pengujian performa *web server* dilakukan dengan mengirim 3 macam uji yaitu dengan mengirimkan beban 100 *request*, 500 *request* dan 1000 *request* yang diukur menggunakan *apache benchmark*. Berdasarkan hasil pengujian memberikan hasil *benchmarking* bahwa dari sisi *respon time* meningkat sebanyak 96%. Pada pengujian dengan uji coba *full traffic* dengan 1000 *request* menunjukkan bahwa respon time dari yang sebelumnya 31,201 ms menjadi 1,144 ms

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR LAMPIRAN

Lampiran Halaman

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Dunia internet di era globalisasi sekarang berkembang dengan pesat, dikuti dengan banyaknya pengguna internet yang semakin meningkat jumlahnya. Ketika internet menjangkau informasi global, layanan *web* dalam internet menjadi sistem penyebaran data berskala global (Nanda *et al*, 2015: 41). Penlitian sebelumnya menyatakan bahwa lalu lintas data yang terjadi pada internet sebanyak lebih dari 75% didominasi oleh data *Hypertext Transfer Protocol* (HTTP).

Informasi dapat di akses dengan berbagai cara salah satunya melalui situs *web* (Kamil *et al*, 2019: 1349). Jumlah situs *web* yang diakses semakin meningkat, kemampuan untuk mengakses dan menyediakan informasi secara cepat dan akurat menjadi sangat esensial bagi sebuah organisasi, baik yang berupa organisasi komersial (perusahaan), perguruan tinggi, lembaga pemerintahan, maupun individual (pribadi), akibatnya perusahaan dan organisasi banyak yang bergantung pada kinerja *web server* (Luthfi *et al*, 2018: 1, Chandra, 2019: 48).

Sebagai salah satu layanan informasi yang mempunyai jumlah akses yang tinggi maka perlu dibangun *website* yang mampu menangani permintaan *(request*) dari banyak pengguna dengan baik (*reliable*) (Kurniawan & Widiyanto, 2016: 109). Banyaknya pengguna *website* mengakibatkan *latensi* dari pengguna, pemanfaatan *bandwidth*, dan beban *server* mejadi meningkat karena jumlah pengguna *website* di internet yang semakin meningkat (Nanda *et al*, 2015: 41).

Ada hubungan dua arah untuk mengakses *web*, yaitu *client* (yang meminta sumber daya *server*) dan *server* atau penyedia layanan (Irza, Zulhendra, & Efrizon, 2017). *Web server* adalah *file server* yang terhubung ke *client* melalui internet (Lijun & Qiuyu, 2014: 230). Penggunaan teknologi informasi saat ini sangat bergantung pada tingkat kinerja *web server* yang tinggi dikarenakan banyaknya jumlah situs *web* yang mempunyai permintaan untuk akses yang akurat, lebih cepat, berkelanjutan & konten *web* yang menarik sangat berpengaruh terhadap kepuasan pengguna (Kunda *et al*, 2017: 43).

Dengan banyaknya beban yang ditanggung oleh *server* akibatnya *server* tidak mampu untuk melayani semua permintaan (*request*) dan menjadi lamban dalam merespon permintaan (*request*) (Adnan & Kusnawi, 2016: 1). Karenanya pengunjung *website* lebih memilih untuk meninggalkan *website* yang dikunjungi jika *loading* (*respon time*) *website* tersebut terlalu lama (Kusumo *et al*, 2014: 1).

Website yang sering diakses oleh *client* memiliki beban proses yang tinggi dalam melayani *request* dan memungkinkan *web server* tidak mampu melayani *request* yang sangat banyak. Hal ini bisa mengakibatkan *web server* mengalami *overload*, lambat, dan akhirnya *server* menjadi *down* (Dwijaya, 2018: 2). Dalam beberapa kasus, *client* tidak tahu jumlah antrian atau *request* pada *website* yang diakses, tapi *client* memilih untuk meninggalkan *website* tersebut karena menunggu (*respon time*) terlalu lama (Dwijaya, 2018: 1). Sebagai penyedia konten *web server* diharapkan selalu dapat memenuhi semua permintaan (*request*) dari pengguna dengan cepat (Putri *et al*, 2018: 17).

Kinerja dari sebuah *web server* yang bagus sangat berpengaruh dalam kualitas layanan yang diberikan. Kualitas *web server* dapat dikatakan baik jika mampu melayani setiap permintaan (*request*) dari pengguna secara cepat dan dapat mengantisipasi kesalahan sekecil mungkin (Kurniawan & Widiyanto, 2016: 109). Hasil survey W3Techs per tahun 2019 mengidentifikasi adanya lebih dari 60 aplikasi *web server* yang saat itu dipakai di Internet. *Survey* ini juga mengidentifikasi sepuluh *web server* yang paling banyak dipakai adalah *Apache* dan *Nginx*. Walaupun *apache* menjadi *web server* yang paling banyak dipakai akan tetapi *apache* memiliki beberapa kekurangan salah satunya adalah *apache* menjadi lamban pada saat permintaan yang masuk berjumlah besar. *Apache* masih menggunakan proses terpisah untuk setiap koneksi, dimana saat *web server apache* menerima koneksi HTTP, maka *apache* akan menciptakan sebuah proses baru untuk menangani setiap koneksi baru disisi lain, nginx menggunakan arsitektur *event driven* dimana setiap proses pekerja dapat menangani ribuan koneksi HTTP secara bersamaan (Data *et al*, 2017: 21).

Berdasarkan uraian latar belakang di atas peneliti berfokus pada penelitian tentang menganalisis performa dari *apache web server* dengan judul **"OPTIMASI *APACHE WEB SERVER* MENGGUNAKAN *VARNISH WEB CACHE DAN REVERSE PROXY NGINX".**

1.2. Rumusan Masalah

Berdasarkan latar belakang diatas, rumusan masalah dalam penelitian ini adalah bagaimana performa *apache web server* setelah dioptimasi menggunakan *varnish web cache* dan *reverse proxy nginx*?

1.3. Batasan Masalah

Pada penelitian ini diperlukan batasan-batasan agar tujuan penelitian dapat tercapai. Adapun batasan masalah yang dibahas pada penelitian ini adalah:

1. *Cache* yang digunakan untuk mengoptimasi *apache web server* adalah *varnish web cache* dan *reverse proxy nginx*.
2. *Tools benchmark* untuk analisis performa adalah *Apache Benchmark.*
3. VPS yang digunakan dalam penelitian ini adalah *Digital Ocean.*
4. Sistem operasi yang digunakan adalah Ubuntu 18.04.
5. CMS yang digunakan adalah *wordpress*.

1.4. Tujuan Penelitian

Untuk mengoptimasi performa dari *apache web server* menggunakan *varnish web cache* dan *reverse proxy nginx* dengan membandingkan hasil sebelum dan sesudah diterapkan *varnish web cache* dan *reverse proxy nginx*.

1.5. Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Dapat mengetahui performa *apache web server* yang telah dioptimasi menggunakan *varnish cache* dan *reverse proxy nginx.*
2. Dapat memberikan wawasan bagi penulis dan pengetahuan bagi dunia pendidikan terkait optimasi performa *apache web server* menggunakan *varnish cache dan reverse proxy nginx*.
3. Dapat digunakan sebagai acuan sebelum membangun sebuah *web server* pada suatu jaringan.

1.6. Sistematika Penulisan

Sistematika penulisan berguna untuk memudahkan dalam memahami jalan pemikiran secara keseluruhan skripsi. Penulisan skripsi ini secara garis besar dibagi menjadi tiga bagian, yaitu sebagai berikut.

1.6.1 Bagian Awal Skripsi

Bagian awal skripsi terdiri dari halaman judul, halaman pengesahan, halaman pernyataan, halaman motto dan persembahan, abstrak, kata pengantar, daftar isi, daftar gambar, daftar table, dan daftar lampiran.

1.6.2 Bagian Isi Skripsi

Bagian isi skripsi terdiri dari lima bab, yaitu sebagai berikut.

1. BAB 1: PENDAHULUAN

   Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat penelitian serta sistematika penulisan skripsi.

2. BAB 2: TINJAUAN PUSTAKA

   Bab ini berisi penjelasan mengenai definisi atau landasan teori maupun pemikiran-pemikiran yang dijadikan kerangka teoritis yang menyangkut masalah dalam skripsi ini serta penelitian terkait.

3. BAB 3: METODE PENELITIAN

   Bab ini berisi penjelasan mengenai studi literatur, identifikasi alat, eksperimen, analisis hasil dan penarikan kesimpulan.

4. BAB 4: HASIL DAN PEMBAHASAN

   Bab ini berisi hasil penelitian berserta pembahasannya.

5. BAB 5: PENUTUP

   Bab ini berisi simpulan dari penulisan skripsi dan saran yang diberikan penulis untuk mengembangkan skripsi ini.

1.6.3 Bagian Akhir Skripsi

Bagian akhir skripsi ini berisi daftar pustaka yang merupakan informasi mengenai buku-buku, sumber-sumber dan referensi yang digunakan penulis serta lampiran-lampiran yang mendukung dalam penulisan skripsi ini.

BAB 2

TINJAUAN PUSTAKA

2.1. Jaringan Komputer

Pada era globalisasi zaman sekarang, kemajuan dan perkembangan dunia teknologi dan internet sangatlah pesat sehingga berdampak di setiap aktivitas perusahaan, instansi atau lainnya, sebagai kebutuhan pokok dalam media komunikasi antar cabang perusahaan kantor pusat atau sebagai media dalam pertukaran data (*sharing*) antar cabang yang dapat diakses melalui jaringan komputer (Akbar *et al*, 2019: 85). Jaringan komputer (*computer networks)* adalah suatu himpunan interkoneksi sejumlah komputer *autonomous.* Jaringan komputer merupakan kumpulan beberapa komputer dan perangkat jaringan komputer seperti *router, switch, hub*, dan sebagainya (Sofana, 2015:3).

Jaringan komputer berkembang dengan sangat pesat, baik di instansi-instansi komersil, dunia akademik, bahkan rumah-rumah penduduk yang membutuhkan akses internet. Internet diakses oleh banyak orang tanpa terkecuali *hacker* dan *cracker*. Dengan alasan tertentu mereka melakukan penyusupan yang dapat merugikan para pemilik *server* dan jaringan komputer (Fadli *et al*, 2017: 11). Untuk meningkatkan pengguna di jaringan maka perlu dibuatnya membuat *server* dan untuk itu perlu dilakukan pengelolaan *server* (administrasi *server)*. Administrasi *server* pada jaringan komputer digunakan pengontrolan atau pengelolaan akses terhadap jaringan dan sumber daya yang terdapat di

dalamnya. Administrasi itu sendiri merupakan suatu hal yang berhubungan dengan pengelolaan, pemberian jasa atau bantuan, dan pelayanan dan *server* merupakan suatu bagian terpenting dari sebuah jaringan yang bertugas untuk menyediakan layanan yang dibutuhkan oleh *client*. Beberapa PC atau pengguna menggunakan *server* untuk bertukar informasi. Terkadang kita membutuhkan berbagai jenis *server* untuk PC yang berbeda pula. Misalnya - *server* FTP, *web server*, *server* mail, *server proxy*, *server* DSN, *firewall*. Semuanya memiliki fungsi yang berbeda dan tanpa mereka tidak mungkin mempertahankan jaringan (Rahman, 2018: iv).

2.2. *Server*

Server merupakan sebuah komputer dengan prosessor yang mempunyai inti lebih dari satu yang dapat menjalankan aplikasi aplikasi dan services secara bersamaan (Putra, 2019: 2). *Server* secara sederhana dapat berupa satu buah komputer untuk beberapa layanan aplikasi, atau jika jaringannya lebih komplek dan rumit, maka *server* dapat diatur hanya untuk memberikan satu atau beberapa layanan saja, sementara layanan yang lain diserahkan kepada *server* yang lain. Untuk *server* yang terdiri dari satu buah komputer yang melayani beberapa layanan biasanya hanya digunakan untuk lingkungan yang lebih kecil misal sekolah, perkantoran, atau usaha kecil dan menengan (UKM) (Suryana, 2018: 1). Berdasarkan fungsinya *server* dapat dibedakan menjadi.

1. *Web server* : *server* yang berfungsi untuk memberikan layanan protokol http, contoh aplikasi *web server* yaitu : *apache*, *Microsoft IIS*, *Tomcat*, *Nginx*, dll

2. *Database server : server* yang berfungsi untuk menyimpan data secara terpusat dan mendistribusikan ke *client* melalui jaringan *wireless* ataupun kabel, Contoh *database server : MySQL, Postgres, MS SQL Server, Oracle, Interbase,* dll
3. FTP *Server* : *Filezilla, FTPd, pro-FTPd, Wu-FTPd, ftpX , Troll-FTPd.*
4. *Mail Server : Mercury, Merak, sendmail, postix.*
5. *Print / File server : Samba Serve*
6. DNS *Server : Server* yang berfungsi menerjemahkan alamat *host* menjadi IP *address,* contoh : Bind.
7. DHCP *Server* : *server* yang bertugas memberikan IP *address* secara otomatis ke komputer *client.*
8. *Proxy server* : aplikasi ini diterapkan untuk membatasi hak akes ke internet ataupun ke suatu *server*. sehingga dapat dibatasi jumlah pengguna ataupun adanya saringan ke media masa, mana saja yang dapat diakses.

Semuanya memiliki fungsi yang berbeda dan tanpa mereka tidak mungkin mempertahankan jaringan (Rahman, 2018: iv).

2.3. *Web Server*

Web server merupakan suatu perangkat lunak yang berjalan di sisi *server* dan bertugas untuk menerima permintaan dari *web browser,* menerjemahkan permintaan tersebut, dan mengembalikan ke *web browser* hasil dari permintaan itu Dalam bentuk sederhana *web server* akan mengirim data HTML kepada permintaan *web browser* sehingga akan terlihat seperti pada umumnya yaitu sebuah tampilan *website.* Fungsi utama *web server* adalah untuk melakukan atau akan *tranfer* berkas

permintaan pengguna melalui protokol komunikasi yang telah ditentukan sedemikian rupa. Halaman *web* yang diminta terdiri dari berkas teks, video, gambar, *file* dan banyak lagi. pemanfaatan *web server* berfungsi untuk men-*transfer* seluruh aspek pemberkasan dalam sebuah halaman *web* termasuk yang di dalam berupa teks, video, gambar atau banyak lagi. *Web* dapat dibangun dengan menggunakan bahasa HTML dan PHP dengan *style* tampilan menggunakan bahasa CSS. *Web* tersebut disimpan pada satu komputer yang disebut *server* (Dawood *et al*, 2014: 26).

Hubungan antara *web server* dan *browser internet* merupakan gabungan atau jaringan komputer yang ada di seluruh dunia. Setelah terhubung secara fisik, *Protocol* TCP/IP (*networking protocol*) memungkinkan semua komputer dapat berkomunikasi satu dengan yg lainnya. Pada saat *browser* meminta data *web page* ke *server* maka instruksi permintaan data oleh *browser* tersebut di kemas di dalam TCP yg merupakan *protocol transport* dan dikirim ke alamat menggunakan *protocol* berikutnya yaitu *Hyper Text Transfer Protocol* (HTTP). Data yg di passing dari *browser* ke *web server* disebut sebagai HTTP *request*. Data yg dikirim dari *server* ke *browser* disebut sebagai HTTP *response*. Jika data yg diminta oleh *browser* tidak ditemukan oleh *web server* maka akan menampilkan *error* yang sering anda lihat di *web* page yaitu e*rror* : 404 *Page Not Found* (Yogiswara, 2015: 176).

Dokumen yang disimpan di *web server* biasanya diakses oleh *client web* menggunakan *web browser*. Untuk cara kerja dari *web* adalah ketika pengguna mengirimkan permintaan dari *browser*, maka akan mendapat respon dari *web server*

atas permintaan tersebut (Yao & Xia, 2016: 2). Ada hubungan dua arah untuk mengakses *web*, yaitu *client* (yang meminta sumber daya *server*) dan *server* atau penyedia layanan (Irza *et al*, 2017: 75). *Web server* adalah *file server* yang terhubung ke *client* melalui internet (Zhang & Zhu, 2014: 229). Beberapa jenis *web server* di antanya adalah:

1. *Apache Web Server / The HTTP Web Server*
2. *Apache Tomcat*
3. *Microsoft windows Server 2008 IIS (Internet Information Services)*
4. *Lighttpd*
5. *Zeus Web Server*
6. *Sun Java System Web Server*
7. *Nginx*

Seperti sebelumnya disinggung fungsi utama dari *web server* adalah untuk melayani permintaan dari *client*. Sebuah kontak *client web server* melalui jaringan, mengirimkan permintaan yang berisi nama *file* yang diperlukan dan kemudian *server* merespon dengan isi *file* jika ada, operasi dasar dari proses ini dapat digambarkan dari Gambar 2.1. (Kunda *et al*, 2017: 43).

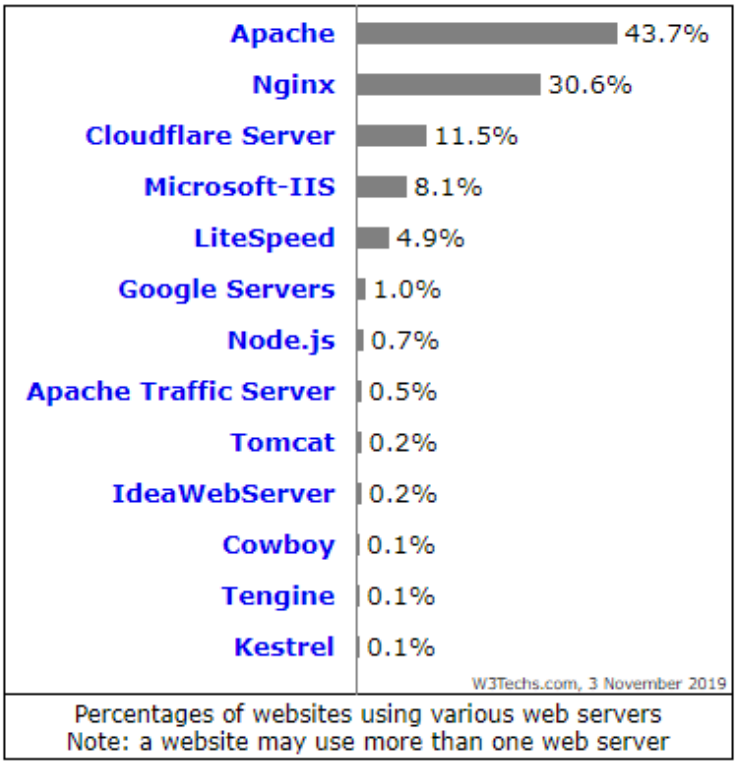
Gambar 2.1 Dasar Fungsi *Web Server*

Pada saat ini, sebagian besar aplikasi berbasis *web*. *Web server* memainkan peran penting dimana *web server* menangani semua permintaan pengguna. Perlambatan di tingkat *web server* akan mempengaruhi pengalaman pengguna. Sebagai kinerja aplikasi *web* semakin semakin penting, karena jumlah pengguna dan pesaing masih meningkat, untuk alasan ini, penelitian terbaru telah secara komprehensif dieksplorasi metode untuk mengevaluasi atau mengoptimalkan kinerja *web server* (Xu *et al*, 2013: 1). Kinerja *web server* mengacu pada kemampuan dari *web server* menanggapi permintaan, yang secara langsung mempengaruhi kepuasan pengguna (Qu *et al,* 2010: 1).

2.4. Apache

Menurut lembaga *survey* Netcraft (2017) *Apache HTTP Server* adalah perangkat lunak *web server open-source*. *Open-source* yang dikembangkan dan dikelola oleh *Apache Software Foundation*. Rilis pada tahun 1995, *apache* menjadi perangkat lunak *web server* yang paling banyak digunakan hingga saat ini. Sebuah survei yang dilakukan oleh Netcraft menunjukkan bahwa pada Februari 2017 *Apache* digunakan di 45,78% situs aktif .

Berdasarkan *survey* dari lembaga *survey* W3Tech *apache web server* juga menempati posisi teratas dengan pengguna sebanyak 43, 7% dan pada urutan kedua terdapat *nginx web server* dengan pengguna sebanyak 30,6% seperti pada Gambar 2.2. Ini membuktikan bahwa sampai saat ini *apache web server* masih menjadi *web server* yang digemari dan banyak digunakan.



Gambar 2.2 *Survey Web Server*

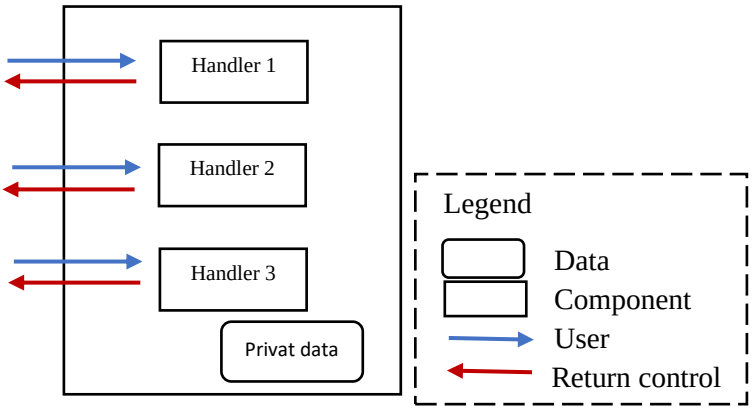
Beberapa dukungan yang terdapat pada *apache web server* yang menjadi alasan *client* untuk menggunakannya dibanding *web server* yang lain diantaranya.

1. Kontrol akses. Kontrol ini dapat dijalankan berdasarkan nama *host*.
2. *CGI (Common Gateway Interface)* Yang paling terkenal untuk digunakan adalah *perl* (*Practical Extraction and Report Language*), didukung oleh *apache* dengan menempatkannya sebagai modul (*mod_perl*)
3. PHP (*Personal Home Page/PHP Hypertext Processor*) Program dengan metode semacam CGI, yang memproses teks dan bekerja di *server*. *Apache* mendukung PHP dengan menempatkannya sebagai salah satu modulnya (mod_php).
4. SSI (*Server Side Includes*)

Keunggulan *apache* dibandingkan dengan *web server* yang lain seperti IIS (*Internet Information Service*) dari *microsoft* adalah kemampuannya untuk mendukung berbagai bahasa *script* paling populer seperti PHP (*Personal Home Page*) dan JSP (*Java Server Pages*). Hal lain yang membuat *apache* lebih diminati adalah sistem lisensinya yang gratis sehingga mengurangi biaya yang perlu dikeluarkan dalam membangun situs *web* dinamis (Emanuel, 2014: 23)

Apache adalah salah satu *web server* yang bertanggung jawab pada *request-response HTTP* dan *logging* informasi secara detail. Secara umum arsitektur apache bekerja dengan model thread-based atau berbasis proses. *Web server* berbasis proses menggunakan proses (*thread*) untuk menerima dan merespon permintaan. Setiap permintaan akan diciptakan sebuah thread yang disimpan dalam sebuah *pool* pada alokasi memori tertentu dan dilakukan proses untuk menjawab, setelah proses dieksekusi kemudian hasil proses dikirimkan kembali ke klien serta dicatat dalam sebuah *log* (Yogiswara, 2015: 176).

Selain itu, a*pache* juga terkenal sebagai suatu *web server* yang kompak, modular, mengikuti standar protokol HTTP, dan tentu saja sangat digemari. Kesimpulan ini bisa didapatkan dari jumlah pengguna yang jauh melebihi para pesaingnya (Irza *et al,* 2017: 76). Proses *handler* apache ditunjukkan pada Gambar 2.2.



Gambar 2.3. Proses *handler* yang dilakukan *apache*

Fungsi dari *handler* melakukan apa yang perlu dilakukan untuk memenuhi permintaan, kemudian mengirimkan proses itu kembali ke komponen HTTP_REQUEST dari inti *apache* agar dikirim ke modul lain untuk diproses atau dikembalikan ke *client* (Irza *et al*, 2017: 76).

2.5. *Reverse proxy*

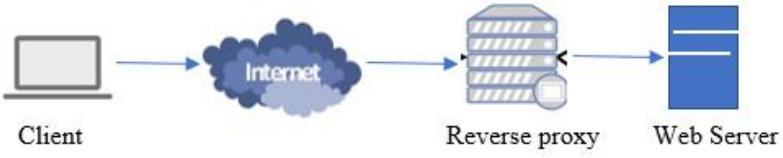
Proxy adalah sebuah sistem komputer atau program aplikasi yang melayani permintaan dari klien dengan meminta layanan ke *server* lain (Ardhian, dkk, 2013). *proxy server* memiliki 3 fungsi utama yaitu :

1. *Connection sharing*: perantara *client* dan *server*.
2. *Filtering* : bekerja pada *layer* aplikasi yang dapat memblokir paket-paket tertentu.
3. *Caching* :mampu menyimpan informasi yang pernah diakses dari *server-server*.

Proxy di bagi menjadi 2 yaitu *forward proxy* dan *reverse proxy*. *Forward proxy* adalah *proxy* yang meneruskan data ke *host* tujuan. *Reverse proxy* adalah sebuah *proxy* yang berada di depan dari *web server*, digunakan sebagai *cache* atau bisa juga sebagai *load balancer*. *Reverse proxy* menjadi perantara *user-user* di internet terhadap akses ke *web-web server* yang berada pada *local area network*, sehingga seolah-olah *user* di internet mengakses langsung *web server* yang dimaksud padahal sesungguhnya *user* di internet mengakses *web-web server* yang terdapat di *local area network* melalui *reverse proxy* tersebut (Suprama & Prismana, 2016: 118).

Reverse proxy adalah sebuah *proxy* yang terletak di *front end web server* yang berfungsi sebagai *cache* (Kusumo *et al*, 2014: 2). Menurut Netcraft (2017) r*everse proxy* digunakan untuk menerima permintaan dari pengguna dan mendistribusikannya ke *server* tertentu di *cluster server* menggunakan algoritma *load balancing*. Kemudian *server reverse proxy forwarding* merespons pengguna. Tetapi jika klien mengirim permintaan yang sama, *reverse proxy* mengirimkan respon yang telah di-*cache.*

Reverse proxy dalam jaringan komputer dapat diperlakukan sebagai *server proxy* khusus (Yuan *et al*, 2013: 652). Proses *reverse proxy* di tunjukkan pada Gambar 2.3.



Gambar 2.3 *Reverse proxy* menerima permintaan dari klien

Cluster server sesuai dengan strategi tertentu. Konten terakhir akan dikembalikan dengan *reverse proxy* seolah-olah itu berasal dari *server*. (Yuan *et al*, 2013). *Reverse proxy* juga dikenal sebagai akselerasi *web server* web dimana ini adalah metode untuk mengurangi *load* pada *web server*, manfaat lain yang bisa didapat adalah peningkatan keamanan. *Reverse proxy* membantu untuk mengurangi waktu tunggu (*respon time*) pengguna. Tapi ada keterbatasan dari konfigurasi *hardware* dan ukuran *cache reverse proxy*. *Reverse proxy* diperlukan untuk membatasi ukuran *cache* atau berakhirnya waktu. *Cache* akan secara otomatis dibersihkan setelah melebihi berakhirnya waktu respon (Data *et al,* 2017: 22).

Dengan jumlah pengguna di *web* yang terus meningkat, situs *web* harus kadangkala menanggung beban berat dan beresiko terhenti di bawah banjir pengunjung. Salah satu solusi untuk masalah ini adalah dengan menggunakan *caching* HTTP *reverse proxy*, yang bertindak sebagai perantara antara aplikasi *web* dan pengguna. Konten dari aplikasi disimpan dan diteruskan, menghindari perlunya aplikasi memproduksinya lagi untuk setiap permintaan (Dely, 2014: 3).

Reverse Proxy berada di garda depan menerima *request* HTTP. *Request* (umumnya diport 80). Di *port* 80 *Reverse Proxy* tidak menggantikan fungsi w*eb server*, melainkan dia akan melanjutkan *request* HTTP tersebut ke *web server* untuk diolah. Sacara sederhana r*everse proxy* adalah resepsionis atau *frontdesk* yang akan menerima pesan dari luar yang kemudian diteruskan kepada pihak yang dituju. Apabila ada respon, pesan tersebut akan disampaikan oleh *frontdesk* kepada si pengirim pesan. Keuntungan yang diperoleh dengan menggunakan *reverse proxy* dan semua tergantung kepada bagaimana tujuan dan *setup*-nya. Meskipun tidak

diperlukan dalam semua kondisi, *reverse proxy* bisa bermanfaat tergantung pada skenario yang diperlukan, berikut ini adalah beberapa manfaat dari penerapan *reverse proxy*

2.6. Nginx

Dimitri Aivaliotis mengemukakan bahwa *Nginx* pertama kali disusun menjadi HTTP. *Nginx* merupakan salah satu *software web server* yang *powerful* dan mempunyai performa tinggi dan *server reverse proxy*, dalam kasus koneksi yang sangat bersamaan, memori, CPU dan sumber daya sistem lainnya memiliki konsumsi sangat rendah, dan operasi yang stabil, selain itu *Nginx* juga didesain untuk *server* dengan sumber daya yang kecil (Chi *et al*, 2012: 1029). *Nginx* menggunakan arsitektur *event driven*. Dalam *nginx*, setiap proses pekerja dapat menangani ribuan koneksi HTTP secara bersamaan. Hasil arsitektur ini adalah *server* ringan, *scalable*, dan kinerja tinggi HTTP (Data *et al*, 2017: 21).

Nginx di rancang untuk mengatasi masalah C10K yang dijelaskan oleh Daniel Kegel, maksud dari C10K adalah merancang *web server* untuk menangani 10.000 koneksi simultan, *Nginx* juga mampu melakukan hal ini melalui mekanisme *event-based connection-handling*, tetapi harus menggunakan sistem operasi yang mendukung mekanisme tersebut (Aivaliotis, 2013: 1).

Nginx dibangun di Rusia untuk memenuhi kebutuhan mesin pencari skala besar Ramble yang tetap memanfaatkannya sampai sekarang. Berkat berbagai kemampuan yang dimilikinya, termasuk kinerja yang tinggi dan fleksibilitas dalam

konfigurasi, *nginx* banyak digunakan untuk mendukung layanan *web* skala besar antara lain *wordpress.com, SourceForge, Hulu, ComputerBase* (Nedelcu, 2010).

Nginx pada awalnya dirancang untuk bertindak sebagai *server* tetapi saat ini *nginx* juga memiliki kemampuan untuk bertindak baik sebagai *reverse proxy* dan *reverse proxy caching. Nginx* akan menyimpan *file cache* pada *disk* di *server*. Jika ada *request* pertama kali ke *website, nginx* akan memberikan/men-*generate* sebuah *hash key* untuk setiap komponen yang akan di *cache* dan menyimpannya di direktori yang ditentukan dalam konfigurasi. Sebetulnya, *nginx* menyimpan *hash* key nya di *memory* dan *file cache* nya di *disk*. Jadi setiap kali ada *request*, *nginx* akan mencari *hash key* yang tersedia di *memory* kemudian mencocokannya dengan informasi hash *key* pada *file cache* yang disimpan di *disk*. *Hash key* itu jadi seperti sebuah shortcut (Chi *et al*, 2012: 1031).

Dari tes ini Chi et al., (2012) menemukan bahwa solusi *cache web* bahwa *nginx* cocok dan efektif dalam lingkungan konkurensi tinggi, tetapi tidak membandingkan efektivitas ini dengan solusi lain yang tersedia.

2.7. Varnish Web Cache

Untuk meningkatkan kecepatan akses pada aplikasi berbasis *web* digunakan *cache*. *Web cache* berada diantara *web server* dan *client*. Ketika ada *request* dari *client*, *cache* akan menyimpan respons, dalam bentuk html, gambar, atau *file*. Selanjutnya apabila ada *request* yang sama dari sisi *client*, maka cache server akan langsung memberikan respons, tidak perlu kembali mengakses *web server* asli. Tujuan utama *web cache* untuk mengurangi *latency* (waktu tunda) dan

mengurangi beban network server utama. Secara umum ada 3 jenis *web cache*, yaitu; *browser cache, proxy cache,* dan *gateway cache* (Kusuma *et al,* 2016: 260)

Cache HTTP *server* (akselerator *web*) biasanya digunakan antara *client* dan *server* HTTP dengan melayani *client* dengan mengirimkan *file* yang disimpan dalam *cache sever* HTTP *(web server*). *Varnish* secara fungsi hampir sama dengan *nginx-cache*, perbedaannya ada pada hal penyimpanan *cache* nya. Jika pada *nginx*, *cache* disimpan ke *disk* di *server*, berbeda dengan *varnish* yang menyimpan *cache* nya di memori. Banyak yang *meng-claim* bahwa *caching* menggunakan *varnish* akan lebih cepat karena *cache* yang disimpan di memori tersebut. Namun karena memang memori memiliki limitasi dari pada *disk* dalam hal ukuran, maka *varnish* harus dikonfigurasi dengan tepat (Xie *et al,* 2012: 1).

Varnish adalah perangkat lunak gratis yang dilisensikan dengan lisensi BSD dua klausa, juga dikenal sebagai lisensi *FreeBSD*. Proyek ini dimulai pada tahun 2005. *Versi* pertamanya adalah dirilis pada September 2006 (*Varnish cache* 1.0), dan versi terbaru *Varnish cache* 3.0.2 dirilis pada Oktober 2011. Fitur utama *Varnish* adalah miliknya kinerja dan fleksibilitas. Melalui bahasa konfigurasi sendiri vcl (*Varnish Configuration Language*) sangat dapat dikonfigurasi, dan pengguna dapat membuat kebijakan bagaimana menangani berbagai skenario lalu lintas (Bakhtiyari, 2012: 27).

Varnish cache adalah sebuah aplikasi *Web accelerator* yang dikenal juga sebagai *Hyper Text Transfer Protocol* (HTTP) *reverse proxy. Varnish* diperkenalkan pertama kali pada tahun 2006 oleh *Poul-Henning Kamp*. *Reverse proxy* adalah sebuah *proxy* yang berada di *front end web server* berfungsi sebagai

cache. *Varnish* bekerja pada *front end* dari sebuah *server HTTP* dan dapat mengkonfigurasi *cache* tiap konten *website*. Prinsip kerja utama *Varnish* adalah menyimpan data halaman *website* di dalam *memory,* sehingga mengurangi beban *web server* memuat halaman yang sama (Kusumo *et al*, 2014: 2).

Varnish Configuration Languange (VCL) merupakan bahasa pemrograman yang digunakan dalam *varnish web cache*. *Syntax* pemprograman pada VCL mirip dengan pemrograman C. Proses *request varnish* dibagi dalam tiga bagian utama yaitu *vcl_recv, vcl_fetch*, dan *vcl_delive.* Varnish mempunyai dua konfigurasi yang berbeda yaitu satu ada di file */etc/varnish/varnish.params* dan satu lagi ada di */etc/varnish/defaul.vcl* (Kusumo *et al*, 2014:2).

2.8. VPS (*Virtual Private Server*)

Virtualisasi *server* menjadi teknologi yang saat ini tumbuh dengan cepat. Terbukti berdasarkan laporan survei diterbitkan oleh spiceworks.com, menyatakan bahwa 76% atau lebih dari tiga perempat responden menggunakan virtualisasi *server* di internet (Soni *et al*, 2019: 18). *Virtual Private Server* lebih dikenal dengan singkatan VPS. Ini adalah sebuah metode untuk mempartisi sebuah komputer *server* menjadi beberapa *server* yang sepertinya *server-server* tersebut berdisi sendiri, seolah-olah sebagai *server* mandiri dan berlaku seperti layaknya sebuah komputer. Meskipun kenyataannya dalam satu komputer terdapat beberapa akun VPS, tetapi satu akun dengan akun lain tidak akan saling mempengaruhi. Hal ini dapat membantu mengimprovisasi tingkat fleksibilitas yang tersedia pada administrator sistem dalam hal pemilihan konfigurasi *software* yang dapat

dijalankan. Selain itu, ini juga dapat memberikan keuntungan yang signifikan dalam hal skalabilitas dari daya pemrosesan (*processing power*), RAM, dan *disk space* dengan biaya yang lebih rendah daripada menggunakan *hardware fisik* tradisional. (Alamsyah & SmitDev, 2009: 31).

Virtualisasi *server* adalah teknik yang digunakan untuk mengkonfigurasi dan mengoperasikan sejumlah *server virtual* dengan membagi sumber daya dari *server* fisik. Ini memberikan sejumlah keuntungan termasuk manajemen sumber daya yang efisien, menghemat ruang, memperkuat keamanan, dan penghematan biaya (Jung *et al,* 2011: 40). Virtualisasi *server* juga mengurangi jumlah *server* fisik dibutuhkan, yaitu dengan memanfaatkan fisik yang keras ruang disk. Ruang dapat dibagi menjadi beberapa bagian yang bias kemudian digunakan oleh mesin *server* virtual. Demikian perusahaan tidak perlu banyak *server* lagi karena dapat ditampung dan digabungkan dalam 1 hingga 2 *server* (Soni *et al,* 2019: 18).

Server virtual ini secara logis bekerja secara terpisah meskipun secara fisik terletak di sama perangkat keras. Ketika beberapa *server virtual* dibuat dan memiliki beban kerja yang berat, *server* dengan sumber daya terbatas harus mampu mengelola penggunaan sumber daya di antara *vitual privat server*. *vitual privat* server (VPS) hosting adalah salah satu layanan *hosting* yang paling banyak digunakan untuk *website*. Tipe *hosting* ini menggunakan teknologi virtualisasi yang menyediakan *resourse dedicated* (pribadi) di *server* meskipun digunakan oleh lebih dari satu *user* (Sabri & Musa, 2013: 1).

2.9. *Apache Benchmark*

ApacheBench adalah alat untuk proses *benchmark apache* HTTP *server* dan di desain untuk memberikan gambaran performa instalasi *apache*. Secara khusus akan menampilkan seberapa banyak *request* per detik yang bisa dilayani oleh *apache* (Kusumo *et al*, 2014:2). Tujuan *benchmark* adalah untuk mengevaluasi dan menggambar perbandingan antara kerangka kerja komputasi aliran terdistribusi yang berbeda. Selain pengukuran kinerja, tolok ukur juga harus memperhatikan sifat yang harus didistribusikan dari kerangka kerja target, oleh karena itu dapat mempertimbangkan kemampuan toleransi kesalahan, ketersediaan, dan skalabilitas (Lu *et al*, 2014: 70).

Perhitungan aliran memiliki banyak fitur unik. Untuk satu hal, selain *throughput*, *latensi* juga merupakan metrik kinerja yang penting. Fitur lain dari perhitungan aliran adalah fitur tersebut terus-menerus menangani data kedatangan. *ApacheBench* adalah alat pengujian beban dasar yang dapat digunakan dengan cepat dan mudah untuk memuat di *server* dan memeriksa kinerja situs di berbagai tingkatan (Rahmel, 2013: 211).

Apache benchmark adalah *tools* yang digunakan untuk mengukur kinerja *server web HTTP*. *Apache benchmark* awalnya dirancang untuk menguji s*erver HTTP Apache*, tetapi *apache benchmark* ini bisa digunakan tidak hanya untuk *apache* bisa juga untuk *web server* lain juga seperti *lighthttpd*, *nginx* atau *miscrosoft iis*. *Apachebench* dengan total koneksi N dibuat dengan konkurensi koneksi C, di mana N dan C ditentukan oleh pengguna. Utilitas *ApacheBench* mencatat *respons*, *throughput*, dan statistik *server* lainnya selama pengujian. Pada dasarnya, masalah

utilitas *apachebench HTTP GET* permintaan untuk konten dalam menetapkan tolok ukur kinerja dari *web server*. Dari hasil pengujian menggunakan *apache benchmark* yang harus diperhatikan adalah berapa hasil dari *requests per second* karena ini menunjukkan seberapa banyak pengunjung yang dilayani dalam satu detik, tapi ingat satu pengunjung bisa membuka beberapa koneksi/permintaan konten sekaligus, dan angka pada *Percentage of the requests served within a certain time (ms),* ini menandakan seluruh permintaan koneksi berhasil dilayankan dalam waktu berapa milidetik. Bagi 1000 untuk mendapatkan 1 detik, jadi kalau dari hasil benchmark diatas paling lama bisa mencapai 3.2 detik. Ini kalau anda matikan parameter header GZIP dan *keep alive* bisa melonjak ke 10 detik lebih (Brunelle *et all*, 2013: 5).

2.10. Penelitian terkait

Penelitian ini dikembangkan dari beberapa referensi yang mempunyai keterkaitan dengan metode dan objek penelitian (Irza, Zulhendra, & Efrizon, 2017; Zhang & Zhu, 2014). Penggunaan referensi ini ditujukan untuk memberikan batasan-batasan terhadap metode dan sistem yang nantinya akan dikembangkan lebih lanjut. Berikut adalah beberapa penelitian yang terkait dengan penelitian yang diusulkan.

Kunda, *et al*., (2017) dalam penelitiannya yang berjudul "*Web Server Performance of Apache and Nginx: A Systematic Literature Review*" bahwa waktu respon, penggunaan CPU dan penggunaan memori bervariasi dengan *web server* yang berbeda tergantung pada model yang digunakan. Namun, ditemukan bahwa

Nginx lebih baik dari *apache* pada banyak metrik yang termasuk waktu respon, penggunaan CPU dan penggunaan memori. Kinerja *Nginx* bawah metrik ini menunjukkan bahwa memori (dalam kasus memori) tidak meningkat dengan meningkatnya permintaan. Disimpulkan bahwa meskipun *nginx* lebih baik dari *apache*, kedua *web server* itu sama-sama kuat, fleksibel dan mampu dan keputusan server yang web untuk mengadopsi sepenuhnya tergantung pada kebutuhan pengguna.

Data, *et al*., (2017) dalam penelitianya yang berjudul "*Optimizing Single Low-End LAMP Server Using NGINX Reverse Proxy Caching*" yang menguji *server* menggunakan 102 koneksi bersamaan, maka kita meningkatkannya ke 408 koneksi bersamaan. Skenario tersebut telah diuji pada *Virtual Private server (VPS) host* di DigitalOcean yang memiliki satu 2,00 GHz CPU dan 512 RAM dengan hasil eksperimen menunjukkan bahwa arsitektur yang diusulkan dapat mengurangi waktu pemuatan halaman hingga 96%, menghemat beban CPU hingga 99% dan menghemat penggunaan memori hingga 28%. Kami menyimpulkan bahwa menggunakan *nginx* sebagai *reverse proxy* pada *server* LAMP adalah solusi alternatif yang baik untuk mengoptimalkan kinerja *web server*.

Luthfi *et al*., (2018) dengan penelitianya yang berjudul "Perbandingan Performa *Reverse Proxy Caching Nginx* dan *Varnish Pada Web Server Apache*" yang membandingkan hasil kedua *server* yaitu *server* dengan *reverse proxy caching* dapat mengungguli *apache web server* tanpa *reverse proxy caching* pada penggunaan CPU dan latensi pada 3 pengujian pertama, sedangkan pada pengujian dengan beban berat *server apache* dengan *reverse proxy caching* memberikan

latensi yang lebih kecil dibandingkan dengan *server apache* tanpa *revese proxy caching*. Hasil *server apache* dengan *reverse proxy caching varnish* memiliki performa yang lebih baik dibandingkan dengan *server apache* dengan *reverse proxy caching Nginx*.

Yang *et al.*, (2000) dalam penelitianya yang berjudul "*Measurement, Analysis and Performance Improvement of the Apache Web Server*" yang mengukur dan menganalisa perilaku dari *server web apache* yang populer pada sistem uniprocessor dan 4-CPU SMP (*symmetric multi-processor*) sistem yang menjalankan sistem operasi IBM AIX. Dengan mengidentifikasi secara kuantitatif kelambatan Performa yang diimplementasikan dalam 6 teknik yang meningkatkan *throughput* dari Apache oleh 61%. Secara rata-rata, *apache* menghabiskan sekitar 20-25% dari total waktu CPU pada kode pengguna, 35-50% pada panggilan sistem kernel dan 25-40% pada penanganan interupsi. Untuk sistem dengan ukuran RAM kecil, kinerja *web server* dibatasi oleh *bandwidth disk*. Untuk sistem dengan ukuran RAM yang cukup besar, tumpukan TCP/IP dan jaringan *Inter-Rupt handler* adalah *bottlenecks* kinerja utama. Teknik ini cukup efektif, dapat meningkatkan *throughput apache* oleh 61%.

BAB 5

PENUTUP

5.1. Simpulan

Hasil penelitian yang didapat dari pengujian menggunkan parameter uji *respon time* dengan beban *request* sebanyak 1000 *request, apache web server* menggunakan *reverse proxy nginx* dan *varnish web cache* memiliki nilai lebih tinggi dengan *respon time* 1,144 (ms), sedangkan *apache web server* tanpa menggunakan *reverse proxy* dan *web cache* membutuhkan waktu 31,201 (ms). Dengan peningkatan *respon time* sebesar 96% *(time per request).* Berdasarkan hasil tersebut dapat diketahui bahwa *apache web server* yang telah dioptimasi menunjukkan performa lebih baik dibandingkan sebelum dioptimasi atau tanpa *reverse proxy* dan *web cache.*

5.2. Saran

Saran yang diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Diharapkan untuk menggunakan metode tambahan agar membuat hasil lebih baik dapat menimalkan penggunaan memori.
2. Diharapkan untuk melakukan pengujian dengan dengan beban lebih dari 1000 permintaan.

DAFTAR PUSTAKA

Adnan, F., & Kusnawi. (2016). Analisis Perbandingan Performa Web Server Apache Dan Nginx Menggunakan Httperf Pada Vps Dengan Sistem Operasi Centos. 1-4.

Akbar, F., Susafa'ati, & Napiah, M. (2019). Metode Point to Point Tunneling ProtocolUntuk Keamanan Jaringan Studi Kasus Kantor Walikota Administrasi Jakarta Barat. *Infortech,* 1(2), 85-91.

Alamsyah, Fahrizal dan SmitDev Community. 2009. E-Business Memb.Bisnis Hosting & Domain. Jakarta: PT Elex Media Komputindo.

Aivaliotis, Dimitri. 2013. Mastering NGINX. Birmingham: Packt Publishing Ltd.

Arman , M. (2016). Analisa Kinerja Web Server E-learning Menggunakan Apache Benchmark dan Httperf. *Jurnal Integrasi*. 8(2), 93-100 .

Aziz, A., & Tampati, T. (2015). Analisis Web Server untuk Pengembangan Hosting Server Institusi: Pembandingan Kinerja Web Server Apache dengan Nginx. *MULTINETICS* , 1(2), 12-20.

Bakhtiyari, Shahab. (2012). Performance Evaluation of the Apache Traffic Server and Varnish Reverse Proxies.Universitas Osloensis. Network and System Administration. University of Oslo

Brunelle, J. F., & Nelson, M. L. (2013). Evaluating the SiteStory Transactional Web Archive With the ApacheBench Tool. 1-12.

Chandra , A. Y. (2019). Analisis Performansi Antara Apache & Nginx Web Server dalam Menangani Client Request. *Jurnal Sistem Dan Informatika (JSI)* , 14(1), 48-56.

Chi, X., Liu, B., Niu, Q., & Wu , Q. (2012). Web Load Balance and Cache Optimization Design based Nginx Under Highconcurrency Environment. *Third International Conference on Digital Manufacturing & Automation* (pp. 1029-1032). China: IEEE.

Data , M., Luthfi, M., & Yahya, W. (2017). Optimizing Single Low-End LAMP Server Using NGINX Reverse Proxy Caching. *International Conference on Sustainable Information Engineering and Technology (SIET)* (pp. 21-23). Malang: IEEE.

Dawood, R., Qiana, S. F., & Muchallil, S. (2014). Kelayakan Raspberry Pi sebagai Web Server: Perbandingan Kinerja Nginx, Apache, dan Lighttpd pada Platform Raspberry Pi. *Jurnal Rekayasa Elektrika*, 11(1), 25-29.

Dely, T. L. (2014). *Caching HTTPA comparative study of caching reverse proxies Varnish and Nginx.* University Of Skovde.

Dwijaya, W. M. (2018). Perbandingan Kinerja Reverse Proxy Antara Squid Dan Varnish Sebagai Web Cache. *Institutional Repositories & Scintific Journal*. 1-4.

Emanuel, A. W. (2014). Instalasi Apache Web Server, MySQL Database, dan PHP pada Sistem Operasi Fedora Core 5. *Jurnal Informatika UKM*, 2(3), 23-35.

Fadli, A., Riadi, I., & Aji, S. (2017). Pengembangan Sistem Pengaman Jaringan Komputer Berdasarkan Analisis Forensik Jaringan. *Jurnal Ilmu Teknik Elektro Komputer dan Informatika (JITEKI)*, 3(1),11-19.

Irza, I. F., Zulhendra, & Efrizon. (2017). Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux.com). *VOTEKNIKA*. 5(2), 75-82.

Joubert, P., Kingy, R. B., Neves , R., Russinovichz , M., & Traceyx, J. M. (2001). High-Performance Memory-Based Web Servers: Kernel and User-Space Performance. *USENIX Annual Technical Conference* (pp. 175-187). Boston: USENIX.

Jung, S. J., Bae, Y. M., & Soh, W. (2011). Web Performance Analysis of Open Source Server Virtualization. *International Journal of Multimedia and Ubiquitous Engineering*, 6(4), 45-52.

Kamil, I., Julham, Lubis, M., & Lubis, A. R. (2019). Management maintenance system for remote control based on microcontroller and virtual private serve. *Indonesian Journal of Electrical Engineering and Computer Science*, 16(3),1349-1355

Kunda, D., Chihan, S., & Sinyinda, M. (2017). Web Server Performance of Apache and Nginx: A Systematic Literature Review. *Computer Engineering and Intelligent Systems*, 8(2), 43-52.

Kurniawan, H., & Widiyanto, E. P. (2016). Analisis Peningkatan Performa Akses Website dengan Web Server Stress Tool. *JATISI*, Vol.2(2), 108-119.

Kusumo, E. B., Andjarwirawan, J., & Gunawan, I. (2014). Pemanfaatan Dan Pengujian Aplikasi Varnish Web Cache Untuk Mempercepat Akses Website. *Infra Journal*, 2(2), 1-6.

Lijun, Z., & Qiuyu, Z. (2014). The Overtime Waiting Model for WebServer Performance Evaluation. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery,*, (pp. 229-232).

Luthfi, M., Data, M., & Yahya, W. (2018). Perbandingan Performa Reverse Proxy Caching Nginx dan Varnish Pada Web Server Apache. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (JPTIIK)*, 2(4), 1457-1463.

Lu, R., Wu, G., Xie, B., & Hu, J. (2014). Stream Bench:Towards Benchmarking Modern Distributed Stream Computing Frameworks. *ACM 7th International Conference on Utility and Cloud Computing* (pp. 69-78). IEEE.

Nanda , P., Singh, S., & Saini, G. L. (2015). A Review of Web Caching Techniques and Caching Algorithms for Effective and Improved Caching. *International Journal of Computer Applications*, 128(10), 41-45.

Nedelcu, C. (2010). *Nginx HTTP Server.* Birmingham: Packt Publishing.

Putra , T. W. (2019). Rancang Bangun Pembelajaran Jaringan Server dengan Sistem Server Cloud Virtual (Hypervisor). *TRANSFORMATIKA*, 1-9.

Putri, F. P., Hutama, R. M., & Kusnadi. (2018). Analisis Perbandingan Kinerja Web Server Apache dan Nginx pada VPS dengan Menggunakan HTTPERFuntukSistem Operasi CentOS. *Jurnal Web Informatika Teknologi, ,* 3(1), 16-24.

Qu, Z., Wang, W., & Li , Z. (2010). Web Server Optimization Model Based on Performance Analysis . *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)* (pp. 1-4). IEEE.

Rahman, T. (2018). *Linux Server Configuration and MikroTik Administration.* Dhaka, Bangladesh: Daffodil International University .

Rahmel, D. (2013). Testing a Site with ApacheBench, JMeter, and Selenium. In D. Rahmel, *Advanced Joomla* (pp. 211-247). Apress, Berkeley, CA.

Sabri , A. A., & Musa , M. N. (2013). Memory Sharing Management on Virtual Private Server. *International Conference on ICT for Smart Society* (pp. 1-4). Jakarta: IEEE.

Sofana, I. (2015). *Membangun Jaringan Komputer.* Bandung: INFORMATIKA.

Soni, Prayudi , Y., Bambang , S., Didik, S., & Harun, M. (2019). Server Virtualization Acquisition Using Live Forensics Method. *International*

Conference of CELSciTech 2019 - Science and Technology track (ICCELST-ST 2019) (pp. 18-23). Riau: Atlantis Pers.

Supramana, & Gusti E. P, I. L. (2016). Implementasi Load Balancing Pada Web Server Dengan Menggunakan Apache. *Manajemen Informatika*, 5(2), 117-125.

Suryana, O. (2018). Server *dan* Web Server. Kuningan, Indonesia: Researchget.

Xie, W., Li, Y., Lu, C., & Shen, R. (2012). Optimizing The Resource-updating Period behavior of HTTP cache servers for better scalability of live HTTP streaming systems. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting* (pp. 1-6). Seoul,: IEEE.

Xu, X., Xu, T., Yin, Y., & Wan, J. (2013). Performance evaluation model of Web servers based on response time. *IEEE Conference Anthology* (pp. 1-5). China: IEEE.

Yao , Y., & Xia, J. ((2016)). Analysis and Research on the Performance Optimization of Web Application System in High Concurrency Environment. *Electronic and Automation Control Conference* (pp. 1-6). Chongqing: IEEE.

Yang, Q., Hu, Y., & Nanda, A. (2000). Measurement, Analysis and Performance Improvement of the Apache Web Server. *18th IEEE International Performance, Computing, and Communications Conferenc* (pp. 1-20). Arizona: IEEE.

Yogiswara. (2015). Kinerja Web Service pada Web Server Apache, Ngin-X dan IIS-7. *SEMNASKIT*, 175-179

Yuan, W., Sun, H., Wang, X., & Liu , X. (2013). Towards Efficient Deployment of Cloud Applications Through Dynamic Reverse Proxy Optimization. *IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference* (pp. pp. 651-658). Zhangjiajie: IEEE.

Zhang, L., & Zhu , Q. (2014). The Overtime Waiting Model for WebServer Performance Evaluation. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (pp. 229-232). IEEE.