PEMANFAATAN DAN PENGUJIAN APLIKASI VARNISH WEB CACHE UNTUK MEMPERCEPAT AKSES WEBSITE

Eko Bayu Kusumo[1], Justinus Andjarwirawan[2], Ibnu Gunawan[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail: ekobayu@mail.com, justin@petra.ac.id, ibnu@petra.ac.id

ABSTRAK

Teknologi *Website* saat ini berkembang dengan sangat cepat. Suatu *Website* mulai dilengkapi dengan berbagai animasi agar lebih menarik tetapi animasi ini kadang-kadang memperlambat akses dari sebuah *Website.* Orang akan meninggalkan *Website* tersebut karena *loading*-nya yang lama. Salah satu cara mempercepat akses *Website* adalah dengan memanfaatkan aplikasi *Web cache* Varnish. Varnish akan menyimpan data statis dari sebuah *Website* dalam sebuah *memory*. Data statis ini adalah data yang jarang berubah atau jarang di-*update.*

Suatu *front end* bagi *Web administrator* akan dibangun untuk aplikasi Varnish sehingga konfigurasinya dapat dengan mudah diganti sesuai kebutuhan. Konfigurasi yang akan disederhanakan melalui *front end* aplikasi Varnish ini adalah mendeteksi SQL *injection* dan *Cross Site Scripting* (XSS), blokir *file* dan *folder*, manipulasi HTTP *header*, mendeteksi *file* asli dihapus atau diblokir, dan konfigurasi *error handling*. Aplikasi *front end* ini diimplementasikan dalam sebuah *virtual machine* dengan VirtualBox sebagai *virtual manager*-nya dan Linux Ubuntu server sebagai sistem operasi-nya.

Aplikasi Varnish diuji menggunakan ApacheBench dimana jumlah *request* dari *client* dan *response time* menjadi parameter utama pengujian.

Kata Kunci: Varnish, *Web Cache, Open Source.*

ABSTRACT

Website technology is currently growing very rapidly. A Website started to be equipped with a variety of animations to make it more interesting but sometimes the animation slowing access of a Website. People will leave the Website because of loading time taking too long. One way to speed up a Website access is by utilizing web applications using Varnish cache. Varnish will store static data from a Website in the memory. Static data is data that changes infrequently or rarely updated.

A front end for Web administrators will be built for Varnish application so that the configuration can be easily replaced as needed. The configuration will be simplified through the front end of this Varnish application is detecting SQL injection and Cross Site Scripting (XSS), block files dan folders, HTTP header manipulation, detects the original file is deleted or blocked and error handling configuration. Application front end is implemented in a virtual machine with VirtualBox as its virtual manager and Linux Ubuntu server as its virtual operating system.

Varnish application was tested using ApacheBench where the number of requests from the client and the response time becomes the main parameters of the test.

Keywords: Varnish, Web Cache, Open Source.

1. PENDAHULUAN

Kebutuhan akses suatu *Website* yang cepat sangat dibutuhkan saat ini. Pengunjung *Website* meninggalkan *Website* yang dikunjungi jika *loading Website* tersebut terlalu lama. Tujuan utama mengunjungi suatu *Website* adalah untuk mendapatkan informasi penting yang dibutuhkan. *Website* seringkali terdapat konten tambahan yang kurang mendukung informasi utama *Website.* Konten tambahan ini akan memperlambat akses dari *Website* tersebut. Pada suatu *website* ada konten yang statis dan dinamis. Konten statis adalah konten yang jarang di-*update* atau jarang ada perubahan. Konten dinamis adalah konten yang sering di-*update* atau selalu ada perubahan. Konten statis seperti *header, sidebar* dan *footer* sedangkan konten dinamis seperti konten berita, *gallery* atau *video*. Konten statis ini dapat disimpan pada suatu *memory* sehingga akses data konten dinamis ke *server* lebih cepat. Data yang disimpan dalam *memory* ini akan mempercepat akses konten statis pada *Website* sehingga jika *Website* dikunjungi banyak *user* maka beban *server* mengakses data konten dinamis akan lebih ringan.

Hal lain yang menjadi masalah adalah jika *server* utama tempat penyimpan data berada jauh dari lokasi *user* pengunjung *Website*. Waktu yang dibutuhkan untuk men-*transfer* data yang diperlukan akan lebih lama. Data pada *memory* ini sangat diperlukan untuk mempercepat *transfer* data ini. Contohnya peneliti memiliki *Web server* di Amerika, tetapi pengunjung *Website* peneliti mayoritas berasal dari Indonesia. Akses data akan sedikit lama karena jauhnya *Web server* peneliti. Data pada *memory* juga membantu menyimpan beberapa data yang sering diakses pengunjung *Website.*

Untuk solusi masalah *Web cache*, dapat diatasi dengan menggunakan Varnish *Web cache.* Dengan Varnish, peneliti dapat menyimpan semua informasi data dari *server* utama ke dalam Varnish. Data statis seperti gambar, video dan hal lain yang berhubungan dengan *Website* dapat disimpan di dalam Varnish ini. Web *Admin* juga dapat mengatur *Time To Live* (TTL) dari tiap *fragment Website* dengan menggunakan konsep *Edge Side Includes* (ESI). Konten *Website* yang jarang di-*update* seperti *header* atau *footer* dapat diatur *time to live* misal selama satu bulan. Selama satu bulan sekali, Varnish secara otomatis meng*update Web cache*-nya. Konten seperti berita ataupun iklan, dapat diatur misal selama 1 jam akan di-*update*. Pembagian waktu *update* bermanfaat mengurangi akses data ke

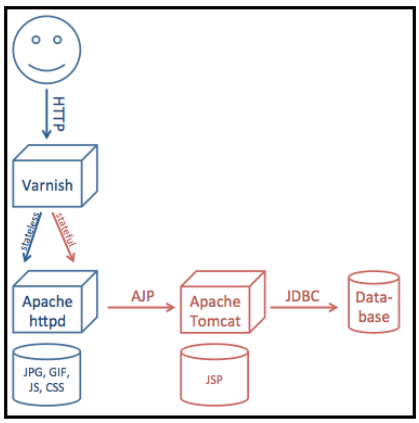
Web server utama dan data yang diterima oleh *user* merupakan data terbaru.

Jika sebuah *Website* diakses banyak pengunjung maka *loading* konten *Website*-nya akan cukup lama. Diperlukan Varnish *Web cache* ini untuk membantu meningkatkan kecepatan akses suatu *Website*. Pada skripsi ini diharapkan dapat membantu permasalahan pada beban *server* dalam mengakses data melalui pemanfaatan dan pengujian aplikasi Warnish *Web cache.*

2. LANDASAN TEORI

2.1 Varnish

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 [4].



Gambar 1. Implementasi Varnish

Sumber: Winkler (2012)

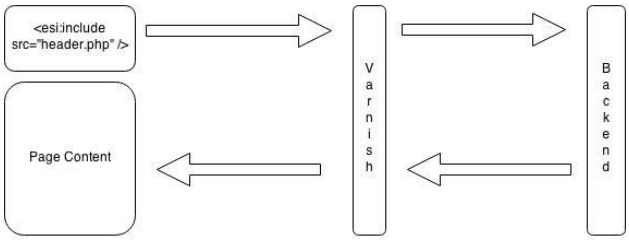
2.2 Varnish *Configuration Languange* (VCL)

VCL merupakan bahasa pemrograman yang digunakan dalam *varnish web cache*. Syntax pemprograman pada VCL mirip dengan pemrograman C. Proses *request varnis*h dibagi dalam tiga bagian utama yaitu *vcl_recv*, *vcl_fetch*, dan *vcl_deliver*.[4].

2.3 *Edge Side Includes* (ESI) Varnish

Edge Side Includes adalah *small markup languange* yang mirip dengan *Server Side Includes* (SSI) untuk memasukkan *fragment* dari suatu halaman *Website*. Kebanyakan *Website* memiliki banyak konten yang dibagi dalam beberapa halaman. Proses *loading* dari tiap halaman *Website* sangat menghabiskan banyak waktu dan ESI akan membuat sebuah *address* dari tiap *fragment Website* agar dapat diatur *cache policy* nya secara individu.

Cara kerja ESI pada Varnish yaitu pertama Varnish mendeteksi ESI dari sebuah halaman *Website* lalu mengecek ESI tersebut di dalam *backend* atau *cache*. ESI diagram dari Varnish dapat dilihat pada Gambar 2:



Gambar 2. Diagram ESI

2.4 Apachebench (ab)

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.[1]

2.5 Gnuplot

Gnuplot adalah sebuah *portable command-line* grafik untuk Linux, MS Windows, OS X, VMS dan banyak *platform* lainnya. Program ini menggunakan dua perintah untuk membuat sebuah *plot* yaitu perintah: `plot` dan `splot`. *Source code* Gnuplot dilindungi hak cipta tetapi bebas didistribusikan (*open source*). Gnuplot pada awalnya diciptakan untuk memungkinkan para ilmuwan dan mahasiswa untuk memvisualisasikan fungsi matematika dan data secara interaktif, tetapi telah berkembang untuk mendukung banyak kegunaan non-interaktif seperti *Website scripting*. Hal ini juga digunakan sebagai *engine* untuk *third-party applications* seperti Octave. Gnuplot telah dikembangkan secara aktif sejak 1986. [2].

2.6 Ubuntu Server

Ubuntu server adalah salah satu varian dari distro Linux Ubuntu dan di desain untuk *server*. Perbedaan paling mendasar adalah Ubuntu server tidak tersedia *Graphic User Interface* (GUI). Perintah–perintah yang dijalankan berjalan di layar hitam atau biasa disebut *console*. Ubuntu server merupakan *software open source* dan disediakan gratis oleh Canonical (perusahaan dibalik Ubuntu). [3]

Spesifikasi yang dibutuhkan Ubuntu *server* minimal prosesor 1 GHz (PIII) dan *Random Access Memory* (RAM) 512MB dengan *hard disk* 5GB. Meskipun *server*, spesifikasi yang dibutuhkan tidak terlalu tinggi. Ubuntu server ada dua macam versi yaitu versi 32 *bit* dan 64 *bit*.

2.6 Hypertext Preprocessor (PHP)

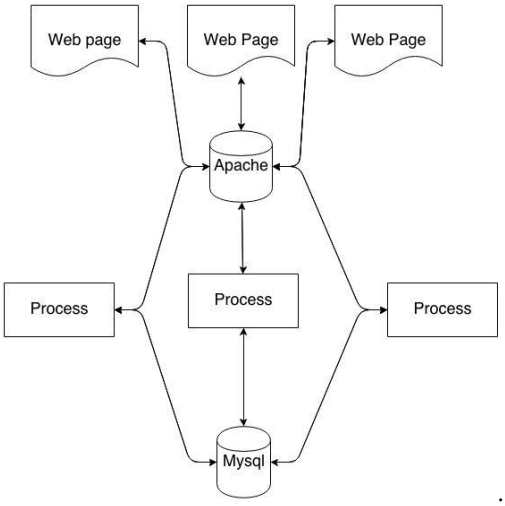
Kepanjangan dari PHP adalah '*Hypertext Preprocessor'*. PHP adalah bahasa *scripting Web* HTML-*embedded*. Kode PHP dapat disisipkan ke dalam HTML halaman *Website*. Ketika sebuah halaman PHP diakses, kode PHP dibaca oleh *serv*er. Output dari fungsi PHP pada halaman biasanya dikembalikan sebagai kode HTML yang dapat dibaca oleh *browser*. Karena kode PHP diubah menjadi HTML sebelum halaman dibuka, pengguna tidak dapat melihat kode PHP pada halaman. Ini membuat halaman PHP cukup aman untuk mengakses *database* dan informasi aman lainnya.[5]

3. DESAIN SISTEM

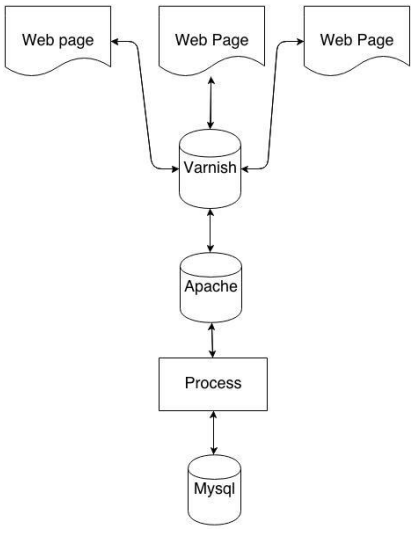
Pada bab ini akan dibahas analisa sistem, diagram dan prosedur *main configuration* Varnish

3. 1 Analisa Sistem

Varnish yang telah ter-*install* dapat diakses melalui *default port* 80. Varnish akan berada pada *front end admin* sehingga *request* data yang keluar masuk akan melalui pengecekan Varnish terlebih dahulu sebelum menuju ke Apache. *Request* data yang *stateless* akan dikelola oleh Varnish sedangkan *request* data yang *stateful* akan dikembalikan ke Apache. Jika ada *request* sebuah halaman *Website* dari *admin* maka *request* tersebut akan dicek oleh Varnish apakah sudah di-*cache*. Jika belum di-*cache* maka *request* tersebut akan diteruskan ke Apache. Jika ada halaman yang sama di-*request* berulang-ulang oleh *admin* yang sama maka akan membebani kinerja Apache. Varnish akan membantu Apache dalam mengatur *request* halaman yang sama dengan sistem menyimpan *cache* di-*memory.* Sistem sebelum dan sesudah menggunakan Varnish dapat dilihat pada Gambar 3 dan Gambar 4.



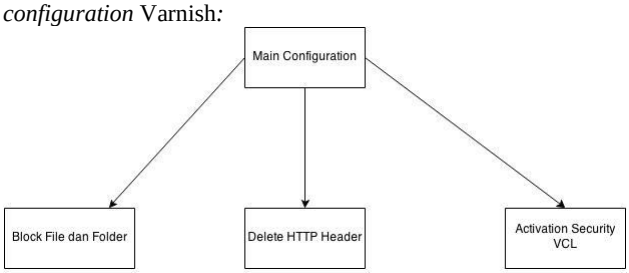
Gambar 3: Sistem Sebelum Menggunakan Varnish



Gambar 4: Sistem Sesudah Menggunakan Varnish

3. 2 Diagram dan Prosedur *Configuration*

Konfigurasi utama untuk *front end admin* pada Varnish dibagi menjadi tiga bagian yaitu *block file* dan *folder, delete* HTTP *header* dan *activation security* VCL. Berikut diagram dari *main configuration* Varnish*:*



Gambar 5: Diagram Main Configuration Varnish

Dari gambar diagram diatas, tampilan utama dari *main configuration* Varnish ada tiga yaitu *block file* dan *folder, delete* HTTP *header* dan *activation security* VCL. Berikut penjelasan dari ketiga menu konfigurasi tersebut:

1) Menu *block file* dan *folder*. Pada menu ini *admin* dapat memilih *file* atau *folder* yang ingin diblokir. *Admin* dapat memilih *file* atau *folder* dari *checkbox button* yang telah disediakan. Setelah menekan tombol *submit* akan muncul notifikasi bahwa *file* atau *folder* pilihan telah berhasil diblokir.

2) Menu *delete* HTTP *header*. Pada menu ini disediakan juga *checkbox button* untuk HTTP *header* yang ingin dihapus oleh *admin. Admin* akan mendapatkan notifikasi bila telah men-*submit* pilihannya.

3) Menu *activation security* VCL. Pada menu ini *admin* dapat memilih *on* atau *off security* pada VCL melalui *radio button* yang disediakan. *Admin* dapat menekan tombol *submit* untuk memproses pilihannya.

4. IMPLEMENTASI SISTEM

Dalam pengujian ini peneliti menggunakan settingan *default* dari Varnish. Berikut detail dari settingan *default* Varnish:

a) Kapasitas *memory* Varnish adalah 256M.
Kapasitas ini digunakan untuk menyimpan *file cache* kedalam *memory* komputer.

b) Minimal *thread* varnish adalah 1 dan maksimal *thread* mencapai 1000.
Minimal dan maksimal *thread* varnish merupakan batas jumlah *user* yang dapat di tangani oleh Varnish.

c) *Idle timeout* jika tidak ada respon dari Varnish adalah 120 detik.
Jika Varnish lama dalam merepson permintaan *user* maka akan muncul *idle timeout.*

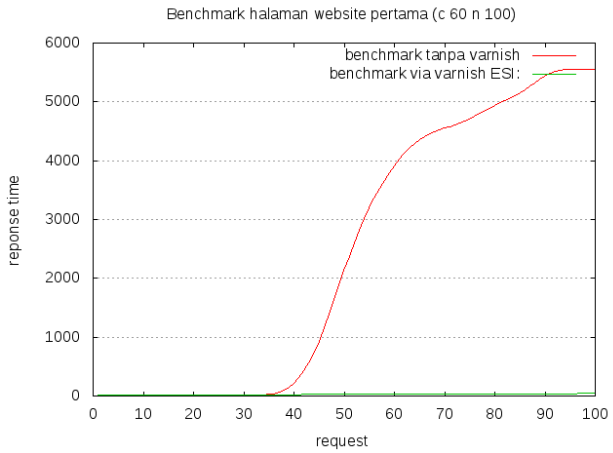
d) Kapasitas *storage cache* Varnish adalah 1G.
Varnish juga memberikan pilihan untuk menyimpan *cache* pada *hard disk* komputer dengan kapasitas 1G.

e) *Default* TTL Varnish adalah 120 detik.
Konten yang kadaluarsa akan di *update* oleh Varnish dengan TTL 120 detik.

f) *Thread Pool* max Varnish adalah 500 dan *thread* min adalah 5.
Varnish akan membuat *pool* dalam *cache*-nya dengan max 1 *pool* menampung 500 *thread/worker* dan min 5 *thread/worker*

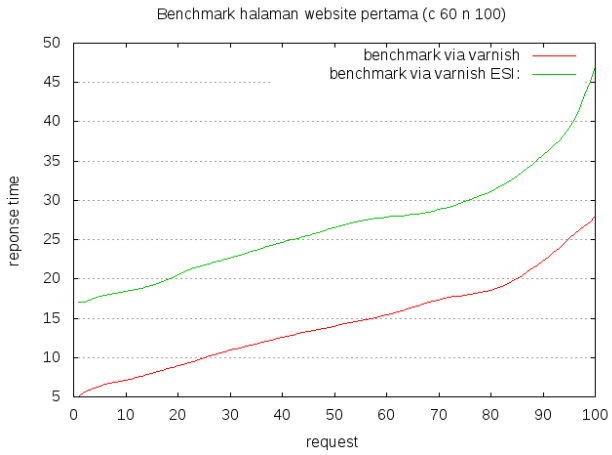
Halaman *Website* pertama memiliki *page size* sebesar 10.843 *bytes*. Pada pengujian ini peneliti membandingkan halaman *Website* pertama yang menggunakan tanpa varnish dengan Varnish ESI dan halaman *website* pertama yang menggunakan Varnish dengan Varnish ESI. Peneliti menggunakan ApacheBench dengan dua nilai interval yang berbeda. Interval pertama *concurrent* 60 *request* 100 dan interval kedua *concurrent* 296 *request* 300.

Peneliti menggunakan Gnuplot untuk visualisasi berupa grafik. Pada Gambar 5 dapat dilihat *response time benchmark* tanpa Varnish mulai berat saat ada 35 *request* yang masuk. Sedangkan *response time benchmark via* Varnish ESI masih stabil pada saat 35 *request* masuk. *Response time* paling lama dari *benchmark* tanpa Varnish adalah 5554ms dan *reponse time* terlama dari *benchmark* dengan Varnish ESI adalah 47ms dengan maksimal 100 *request.*



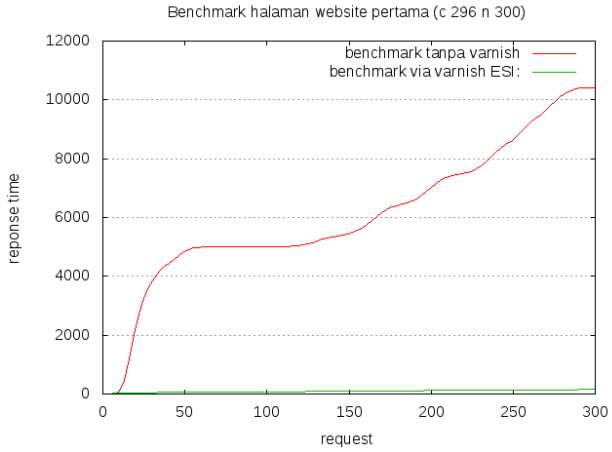
Gambar 5: Grafik Benchmark Website (c 60 n 100)

Peneliti juga mencoba membandingkan *benchmark* halaman *Websi*te pertama yang menggunakan Varnish dengan *benchmark* Varnish ESI. Pada Gambar 6 dapat dilihat *response time* Varnish ESI sedikit lebih lambat daripada Varnish. Pada *request* pertama, *response time* Varnish ESI adalah 17ms sedangkan pada Varnish *response time* nya adalah 5ms.



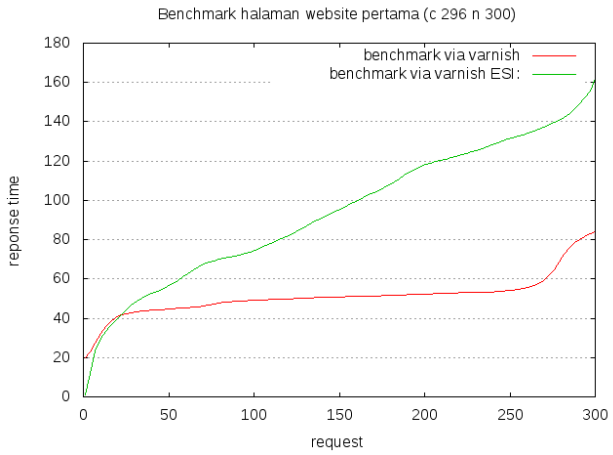
Gambar 6: Grafik Benchmark Website (c 60 n 100)

Pada Gambar 7 dapat dilihat *response time benchmark* tanpa Varnish mulai berat saat ada 20 *request* yang masuk. Sedangkan *response time benchmark via* Varnish ESI masih stabil pada saat 20 *request* masuk. *Response time* paling lama dari *benchmark* tanpa Varnish adalah 10415ms dan *reponse time* terlama dari *benchmark* dengan Varnish ESI adalah 162ms dengan maksimal 300 *request.*



Gambar 7: Grafik Benchmark Website (c 296 n 300)

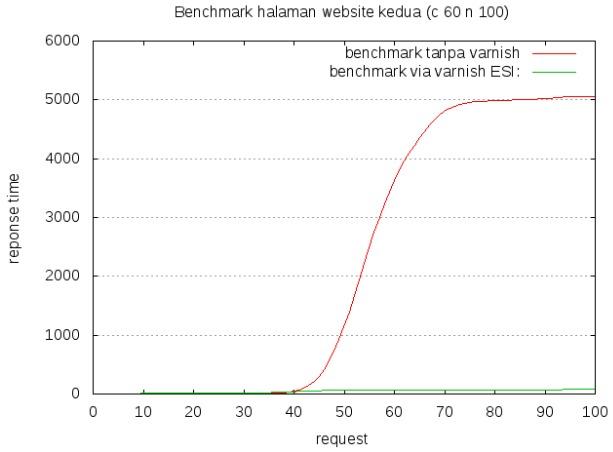
Pada Gambar 8 dapat dilihat *response time* Varnish ESI sedikit lebih lambat daripada Varnish. Pada *benchmark via* Varnish ESI mulai terlihat berat *response time* nya saat ada 30 *request* yang masuk dengan *response time* 40ms. Sedangkan pada *benchmark via* Varnish masih terlihat stabil dengan *response time* 40ms. *Response time* terlama dari *benchmark via* Varnish ESI adalah 162ms dan *response time* terlama *benchmark* Varnish 84ms dengan maksimal 300 *request.*



Gambar 8: Grafik Benchmark Website (c 296 n 300)

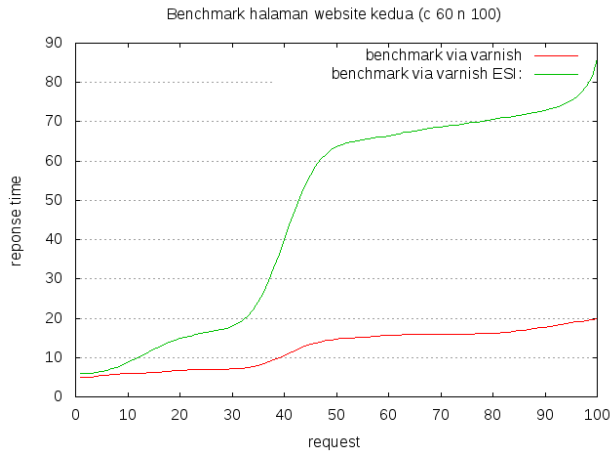
Halaman *Website* kedua memiliki *page size* sebesar 4096 *bytes*. Pada pengujian ini peneliti membandingkan halaman *Website* kedua yang menggunakan tanpa Varnish dengan Varnish ESI dan halaman *Website* kedua yang menggunakan Varnish dengan Varnish ESI. Peneliti menggunakan ApacheBench dengan dua nilai interval yang berbeda. Interval pertama *concurrent* 60 *request* 100 dan interval kedua *concurrent* 296 *request* 300.

Pada Gambar 9 dapat dilihat *response time benchmark* tanpa Varnish mulai berat saat ada 45 *request* yang masuk. Sedangkan *response time benchmark via* Varnish masih stabil pada saat 45 *request* masuk. *Response time* paling lama dari *benchmark* tanpa Varnish adalah 5060ms dan *reponse time* terlama dari *benchmark* dengan Varnish ESI adalah 86ms dengan maksimal 100 *request.*



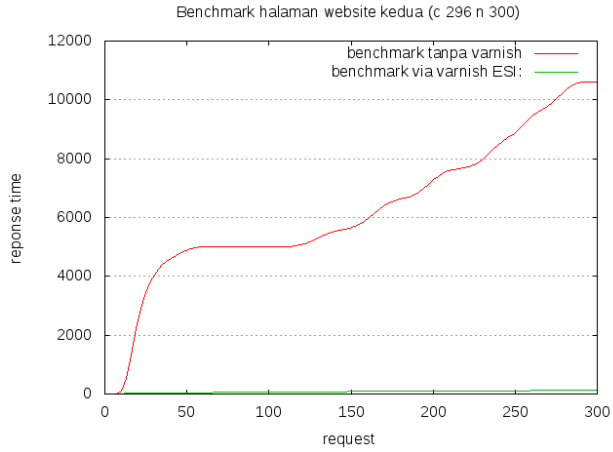
Gambar 9: Grafik Benchmark Website (c 60 n 100)

Pada Gambar 10 dapat dilihat *response time* Varnish ESI sedikit lebih lambat daripada Varnish. *Benchmark* Varnish ESI mulai berat saat 33 *request* masuk bersamaan sedangkan *benchmark* Varnish masih stabil dengan *response time* 8ms. *Response time* terlama Varnish ESI adalah 86ms dan *response time* terlama *benchmark* Varnish adalah 20ms dengan maksimal 100 *request.*



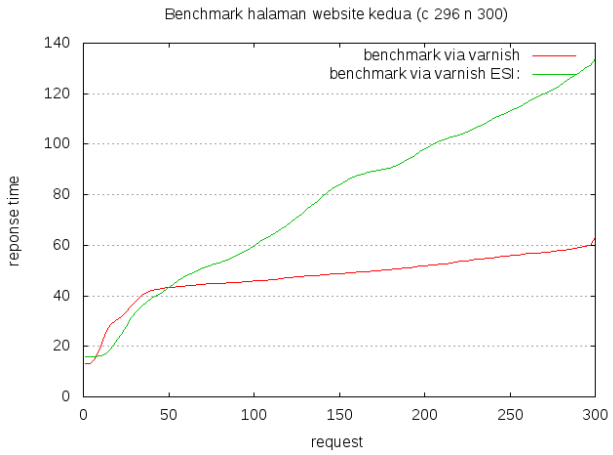
Gambar 10: Grafik Benchmark Website (c 60 n 100)

Pada Gambar 11 dapat dilihat *response time benchmark* tanpa Varnish mulai berat saat ada 30 *request* yang masuk. Sedangkan *response time benchmark via* Varnish ESI masih stabil pada saat 30 *request* masuk. *Response time* paling lama dari *benchmark* tanpa Varnish adalah 10601ms dan *reponse time* terlama dari *benchmark* dengan Varnish ESI adalah 134ms dengan maksimal 300 *request.*



Gambar 11: Grafik Benchmark Website (c 296 n 300)

Pada Gambar 12 dapat dilihat *response time* Varnish ESI sedikit lebih lambat daripada varnish. Pada *benchmark via* Varnish ESI mulai terlihat berat *response time* nya saat ada 50 *request* yang masuk dengan *response time* 40ms. Sedangkan pada *benchmark via* Varnish masih terlihat stabil dengan *response time* 40ms. *Response time* terlama dari *benchmark via* Varnish ESI adalah 134ms dan *response time* terlama *benchmark* Varnish 63ms dengan maksimal 300 *request.*



Gambar 12: Grafik Benchmark Website (c 296 n 300)

5. KESIMPULAN

Dari proses pembuatan konfigurasi Varnish ini sampai tahap pengujiannya, dapat ditarik beberapa kesimpulan yaitu sebagai berikut.

a. Aplikasi Varnish *Web cache* ini sangat bermanfaat bagi *Website* yang tidak interaktif atau *Website* yang statis. Varnish akan menyimpan data *Website* tersebut dalam sebuah *cache* sehingga akses *user* ke *Website* dapat lebih cepat.

b. Kekurangan dari Varnish adalah jika sebuah *Website* dinamis di-*install* aplikasi Varnish ini maka konten yang disajikan ke *user* kemungkinan merupakan konten yang kadaluarsa karena *cache* belum ter-*update* otomatis. Untuk masalah ini peneliti dapat menggunakan fitur ESI pada Varnish untuk mengatur TTL dari bagian *Website* yang dinamis tersebut.

6. DAFTAR PUSTAKA

[1]Apache HTTP Server (2013). *Apache HTTP Server Benchmarking Tool.* http://httpd.apache.org/docs/2.2/programs/ab.html

[2]Gnuplot (2013). *Gnuplot Homepage.* http://gnuplot.info/

[3]Putra, Candra Adi (2012). *Mengenal Ubuntu Server*. http://www.candra.web.id/2012/12/10/mengenal-ubuntu-server/

[4]Winkler, Christian (2012). *Ultra-Performant Dynamic Websites with Varnish.* http://blog.mgm-tp.com/2012/01/varnish-web-cache/

[5]PHP (2014). *PHP Homepage Manual*. http://www.php.net/manual/