SEMINAR NASIONAL INOVASI DAN APLIKASI
TEKNOLOGI DI INDUSTRI (SENIATI) 2016
ISSN : 2085-4218

Evaluasi Performa *Web Server* Menggunakan Varnish HTTP *Reserve Proxy* dan Redis *Database Cache*

Mandahadi Kusuma[1,*], Widyawan[2], Ridi Ferdiana[2]
1 Mahasiswa Pasca Sarjana Jurusan Teknik Elektro dan Teknik Informatika Universitas Gadjah Mada
2 Departemen Teknik Elektro dan Teknologi Informasi Universitas Gadjah Mada
Jln. Grafika 2 Yogyakarta 55281 Indonesia
* E-mail : mandah@ugm.ac.id

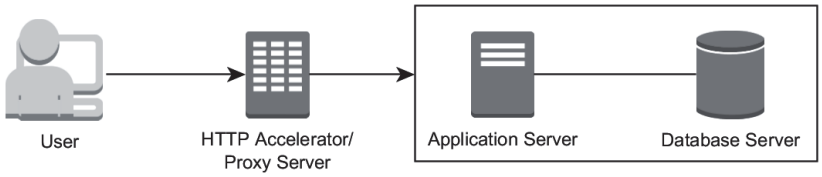
Abstrak. *Cache server* digunakan untuk meringankan beban kerja *webserver* dengan cara menyimpan sementara *content object web* pada memory agar dapat digunakan kembali pada *request* yang sama. Varnish merupakan http reserve proxy *cache* sedangkan Redis-*cache* digunakan sebagai *database cache*. Tujuan penelitian ini untuk evaluasi peningkatan performa *web* yang menggunakan varnish *cache*, redis-*cache*, dan kombinasi diantara keduanya. Penelitian ini menggunakan 4 *server virtual machine* (VM) berbasis VMware pada *server* IBM Blade. Masing-masing *server* berperan sebagai *web server*, *database server*, varnish *server*, dan redis *cache server*. Dilakukan simulasi *benchmark*/ujibeban hingga 10000 *thread/user* selama 600 detik(10 menit), masing-masing *thread* membuka 2 alamat *web* secara berurutan, dari total 500 alamat *web* yang disediakan. Dari hasil simulasi ada peningkatan kecepatan akses (*response time*), penurunan *error rate,* dan penurunan beban kerja *webserver* dan *database server*. Hasil Penelitian menunjukkan bahwa penggunaan varnish dan redis-*cache* pada *webserver* dengan total 20000 *hit*/request alamat web, *error rate* dapat diturunkan dari 10,38% hingga menjadi 2,16% (~3 kali lebih rendah), *response time* menjadi 15-16 kali lebih cepat dibandingkan *webserver* yang tidak menggunakan *cache.*

Kata Kunci: *Cache server*, *Error rate*, Redis, *Response time*, Varnish, *Vebserver*

1. Pendahuluan

Web Cache Secara Umum

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 *response*, dalam bentuk html, gambar, atau *file*. Selanjutnya apabila ada *request* yang sama dari sisi *client*, maka *cache server* akan langsung memberikan *response*, tidak perlu kembali mengakses *web server* asli[1].
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* [2]. Varnish *cache* dan Redis *cache* merupakan *gateway cache*. Varnish *cache* berfungsi sebagai *http reserve proxy*[3] seperti yang diperlihatkan pada gambar 1. Sementara Redis berfungsi sebagai *database cache*, untuk mengurangi beban kerja database server utama. Keunggulan redis-*cache* antara lain mendukung berbagai macam tipe data, memiliki fitur *failover*, dan dapat di*cluster* [4].



*Gambar 13. HTTP Reserve Proxy/*Web *Accelarator*

Keunggulan menggunakan varnish *cache* karena lebih ringan, konfigurasi sangat fleksibel, dan dapat berjalan diatas *multiplatform*[5], Sedangkan redis *cache* berfungsi untuk menyimpan hasil *query* database kedalam *cache* agar *query* yang sama tidak perlu dilakukan berulang kali pada *database* sehingga dapat mengurangi beban pada *server database* dan kecepatan akses meningkat.

ISSN : 2085-4218

Penelitian Terkait

Dari hasil uji performa penggunaan varnish cache pada server fisik [6], kemampuan web server dapat meningkatkan concurrent sessions for 100%, mengurangi response time hingga 90% dan meningkatkan kapasitas transfer data diatas 90%, dan meningkatkan kapasitas transfer data lebih dari 90%. Selain itu varnish juga dapat digunakan untuk meningkatkan keamanan web server untuk mendeteksi serangan SQL injection, Cross Site Scripting (CSS), dan manipulasi HTTP header [7]. Salah satu upaya meningkatkan performa varnish adalah dengan menggunakan mekanisme zero-copy [8], namun penelitian dilakukan masih menggunakan versi kernel linux 2.6.38, sementara kernel linux yang digunakan saat ini sudah versi 3 atau 4

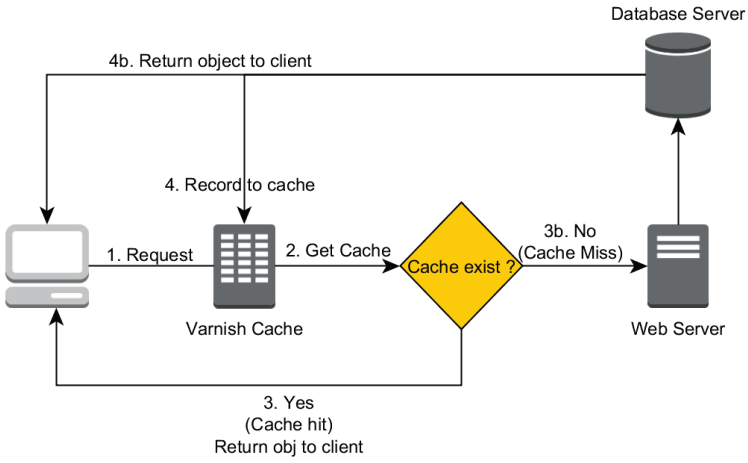
Kelebihan redis adalah memiliki kelebihan throughput yang besar dan rendah latency karena data disimpan didalan memory, selain itu Redis juga dapat dijadikan sebagai system yang terdistribusi [10]. Namun untuk penggunaan machine learning, *scaling up,* dan *bigdata*, lebih baik menggunakan Apache HBase[9], Penggunaan redis cache sebagai cache framework *geolocation* dapat meningkatkan performa hingga 400 kali lebih cepat ketika terjadi *request* dalam jumlah yang masif dibandingkan menggunakan database postgres [11]. Performa redis yang bekerja pada 1 *thread* CPU juga lebih baik dibandingan dengan performa aplikasi memory cache lainnya, yaitu memcached yang bekerja pada lebih dari satu *thread* CPU [12].

Penelitian selanjutnya yang akan dilakukan adalah mengukur performa yang dapat dicapai apabila menggunakan varnish cache dan redis cache secara bersamaan sebagai system cache pada sebuah webserver.

Usulan Penelitian

Penelitian yang dilakukan adalah membandingkan performa penggunaan varnish *cache* dan redis *database cache* pada sebuah layanan *web* multisite berbasis wordpress. Didalam *web* tersebut terdapat subdomain yang dimiliki oleh masing-masing mahasiswa UGM. Pengujian dilakukan dengan melakukan *benchmarking* pada tiga kondisi, yaitu sebagai berikut;

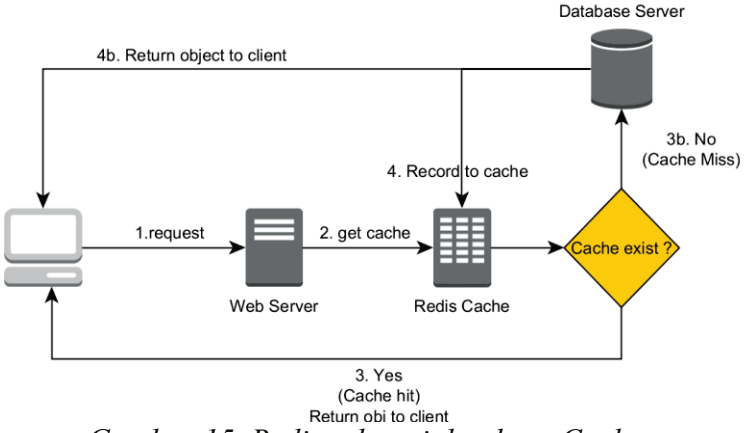
1. *Web* yang menggunakan Varnish sebagai http reserve proxy, seperti pada gambar 2



Gambar 14. Redis sebagai database Cache

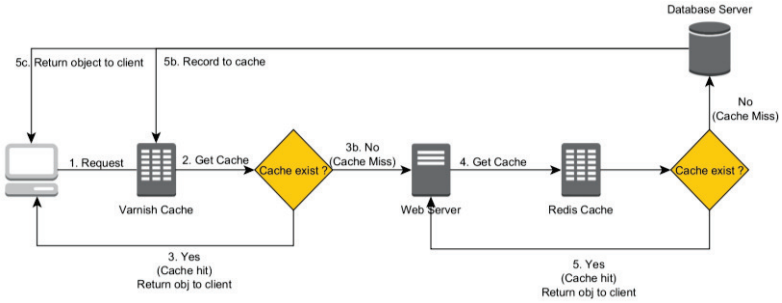
ISSN : 2085-4218

2. *Web* yang menggunakan Redis sebagai *database cache*, seperti pada gambar 3



Gambar 15. Redis sebagai database Cache

3. *Web* yang menggabungkan Varnish dan Redis sebagai http reserve proxy dan *database cache*, seperti pada gambar 4



Gambar 16 Web *server dengan varnish dan redis cache*

Proses *benchmark* menggunakan aplikasi Apache JMeter. Apache JMeter yang berjalan diatas *java virtual machine* dapat mensimulasikan aktivitas *user* menggunakan *web browser* ketika mengunjungi halaman *web*. Komponen *benchmark* yang dibandingkan adalah *response time, error rate, cache hit ratio*, dan *mysql read*.

2. Infrastruktur dan simulasi pengukuran

Infrastruktur

Simulasi ujibeban yang dilakukan menggunakan *Virtual Machine* (VM) VMware yang berjalan diatas *server* IBM Blade. Ada 4 VM yang digunakan, yaitu; VM *webserver*, VM *server database*, VM *Cache Server Varnish* sebagai *http reserve proxy*, dan VM *server* redis sebagai *database cache*. Sistem operasi yang digunakan pada VM adalah Debian versi 7 dan 8. Berikut Spesifikasi teknis pada masing-masing *virtual server* yang digunakan pada penelitian ini;

1. ***Webserver*** **:** Processor 6 core @1.2Ghz, *Memory* 4 GB, menjalankan service Nginx dan hhvm sebagai php compiler
2. ***Database Server*** **:** Processor 6 core @1.2Ghz, *Memory* 4 GB, menggunakan *database* MySql versi 5.6
3. ***Varnish Cache Server*** **:** Processor 2 core @1.2GHz, *memory* 1024 MB
4. ***Redis Cache Server*** **:** Processor 2 core @1.2GHz, *memory* 512 MB

Komponen benchmark

1. *Response time* : Melihat total rata-rata perbedaan kecepatan akses *web* dari sisi client dengan skenario *system cache* yang berbeda. Semakin kecil *response time*, semakin cepat web dapat diakses.
2. *Error Rate* : Jumlah *error* gagal *response* yang akan dialami oleh *client* pada saat *web server* menerima total 20000 request pada masing-masing skenario system cache yang digunakan.

ISSN : 2085-4218

3. *Hit-ratio* : melihat perbandingan antara *hit* dan *miss object cache* pada redis *cache* dan varnish *cache* setiap menitnya (total waktu 10 menit). Semakin besar *hit* pada varnish *cache* maka semakin baik
4. *Mysql Read :* membandingkan rata-rata read *database* perdetik yang dilakukan oleh *web server* setiap menitnya. Semakin sedikit *read* pada *database* mysql maka semakin baik proses *cache* bekerja

Metode Benchmark

Benchmark yang dilakukan menggunakan 500 alamat web aktif yang diambil secara acak dari blog.ugm.ac.id. Disimulasikan ada 10000 *thread/user* yang akan membuka halaman *web* dari daftar alamat web yang telah disediakan, Setiap *thread/user* membuka 2 alamat web/2 *hit* secara berurutan dan berulang secara sehingga total alamat web yang akan dibuka adalah 20000 *hit*. Simulasi dilakukan selama 10 menit/600 detik.

3. Pembahasan Hasil Simulasi Pengukuran

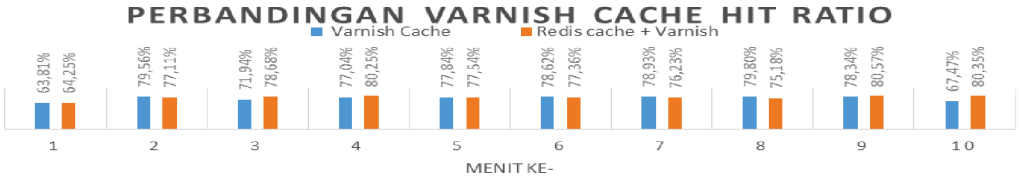
Tabel 3. Error *rate (%) dan* response time *(*miliseconds*)*

Error rate			
Tanpa Cache	**Redis Cache**	**Varnish Cache**	**Redis cache + Varnish**
10,38%	6,03%	2,76%	2,16%

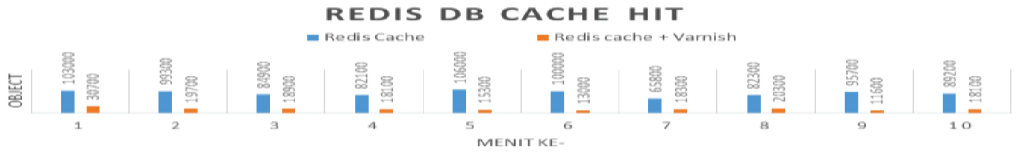
Rerata Response time (ms)			
Tanpa Cache	**Redis Cache**	**Varnish Cache**	**Redis cache + Varnish**
14547	7071	1325	911

Pada simulasi hingga 20000 *hit url*/alamat web, pada semua design akhirnya tetap akan terjadi *error server* gagal memberikan respon kepada *client*. Namun *error* dapat diperkecil dengan menggunakan *cache*. Menggabungkan fungsi varnish *cache* dan redis *cache* pada *webserver* dapat mengurangi *error* hingga 380% Seperti yang ditunjukkan pada table 1. Begitu juga dengan *response time*, ketika menggunakan kombinasi *cache* peningkatan yang dicapai hingga 15 – 16 kali lebih cepat. Namun kecepatan *response time* antara simulasi dengan kondisi nyata akan berbeda, tergantung pada kondisi jaringan yang ada.

Pada gambar 6, adalah data *hit*-ratio varnish *cache*, telihat sedikit peningkatan *hit* antara 2% - 3%, ketika varnish dikombinasikan dengan redis *cache*. Sedangkan pada gambar 5 memperlihatkan *cache hit* pada redis, menurun tajam, antara 70% - 80 %, karena sebagian besar *request* sudah dilayani oleh varnish *cache* dengan baik.

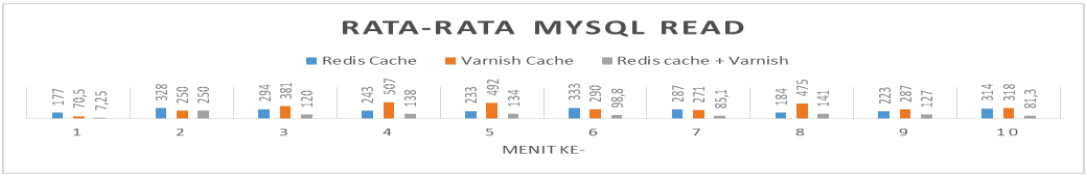


Gambar 17. Perbandingan Varnish Cache hit ratio



*G*ambar 18. Perbandingan Redis cache hit ratio

Penggunaan *cache* juga membantu meringankan kerja *database server*, pada gambar 7 memperlihatkan *database server* jauh lebih ringan, proses read data menurun tajam ketika menggunakan kombinasi varnish dan redis *cache*



Gambar 19. Perbandingan rerata Mysql Read

ISSN : 2085-4218

5. Kesimpulan

Untuk mengurangi beban kerja *webserver* dan *database server*, serta meningkatkan *response time* sebuah *web*, kombinasi *system cache* varnish dan redis *cache* bekerja sangat efektif. Dari *hit* 20000 *url* yang sebelumnya terjadi *error* sebanyak 10,38% dapat diturunkan hingga 2.16% , *error rate* dapat diturunkan dari 10,38% hingga menjadi 2,16% (~3 kali lebih rendah), artinya jumlah *client* yang dapat dilayani juga meningkat karena beban kerja *webserver* sebagian ditangani oleh *server cache, response time* menjadi 15-16 kali lebih cepat dibandingkan *webserver* yang tidak menggunakan *cache.* Juga terlihat peningkatan *hit ratio* 2% - 3% pada varnish *cache* ketika menggunaan redis *cache.*

6. Penelitian Selanjutnya

Saat ini varnish masih menyimpan *cache* didalam memory. Sedangkan pada kondisi nyata, jumlah memory *server* sangat terbatas, menyebabkan penyimpanan *cache* cepat penuh. Usulan penelitian selanjutnya adalah mengganti penyimpanan *cache* didalam RAM dengan penyimpanan didalam NAS (*network attached storage*) berbasis *fiber channel* dengan berbagai metode seperti ISCSI, NFS, dan vmware *attached storage* kemudian melakukan pengamatan performa terhadap kinerja *server cache* dan *webserver* tersebut.

7. Daftar Pustaka

[1] "Caching Tutorial for Web Authors and Webmasters," 10-Apr-2015. [Online]. Available: https://www.mnot.net/cache_docs/#DEFINITION. [Accessed: 10-Apr-2015].
[2] "Use Web Caching to Make Your Web Site Faster | WebReference." [Online]. Available: http://www.webreference.com/authoring/web_caching/index.html. [Accessed: 18-Jan-2016].
[3] "SquidFaq/ReverseProxy - Squid Web Proxy Wiki." [Online]. Available: http://wiki.squid-cache.org/SquidFaq/ReverseProxy. [Accessed: 15-Oct-2015].
[4] "Introduction to Redis – Redis." [Online]. Available: http://redis.io/topics/introduction. [Accessed: 15-Oct-2015].
[5] "Introduction — The Varnish Book." [Online]. Available: https://www.varnish-software.com/book/4.0/chapters/Introduction.html#what-is-varnish. [Accessed: 15-Oct-2015].
[6] P. Brcic, "Improving the performance of physical servers using a proxy servers accelerators," in *Telecommunications Forum (TELFOR), 2013 21st*, 2013, pp. 865–868.
[7] J. Andjarwirawan, I. Gunawan, and E. B. Kusumo, "Varnish Web Cache Application Evaluation," in *Intelligence in the Era of Big Data*, R. Intan, C.-H. Chi, H. N. Palit, and L. W. Santoso, Eds. Springer Berlin Heidelberg, 2015, pp. 404–410.
[8] X. Song, J. Shi, H. Chen, and B. Zang, "Revisiting Software Zero-Copy for Web-caching Applications with Twin Memory Allocation," presented at the Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12), 2012, pp. 355–360.
[9] V. Sharma, J. Carroll, and A. Khune, "Scaling Deep Social Feeds at Pinterest," in *2013 International Conference on Social Computing (SocialCom)*, 2013, pp. 7–12.
[10] Z. Ji, I. Ganchev, M. O'Droma, and T. Ding, "A Distributed Redis Framework for Use in the UCWW," in *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2014, pp. 241–244.
[11] H. Yu, Y. Liu, C. Tian, L. Liu, M. Liu, and Y. Gao, "A cache framework for geographical feature store," in *2012 20th International Conference on Geoinformatics (*GEOINFORMATICS*)*, 2012, pp. 1–4.
[12] D. Stjepanovic, M. Savic, J. Jokic, and S. Maric, "Performance measurements of some aspects of multi-threaded access to key-value stores," in *Telecommunications Forum Telfor (TELFOR), 2015 23rd*, 2015, pp. 831–834.