Available at https://jtsiskom.undip.ac.id (5 November 2019)
DOI:10.14710/jtsiskom.8.1.2020.1-5

Algoritme penggantian cache proxy terdistribusi untuk meningkatkan kinerja server web

Distributed proxy cache replacement algorithm to improve web server performance

Marvin Chandra Wijaya[*)]

Program Studi Sistem Komputer, Fakultas Teknik, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri 65, Bandung, Indonesia 40164

Cara sitasi: M. C. Wijaya, "Algoritme penggantian cache proxy terdistribusi untuk meningkatkan kinerja server web," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 1, pp. 1-5, 2020. doi: 10.14710/jtsiskom.8.1.2020.1-5, [Online].

Abstract – The performance of web processing needs to increase to meet the growth of internet usage, one of which is by using cache on the web proxy server. This study examines the implementation of the proxy cache replacement algorithm to increase cache hits in the proxy server. The study was conducted by creating a clustered or distributed web server system using eight web server nodes. The system was able to provide increased latency by 90 % better and increased throughput of 5.33 times better.

Keywords *- proxy server; distribute proxy cache; web QoS; cache replacement*

Abstrak - Kinerja pemrosesan web perlu meningkat untuk memenuhi pertumbuhan penggunaan internet, salah satunya dengan menggunakan cache pada server proxy web. Penelitian ini mengkaji implementasi algoritme penggantian cache proxy untuk meningkatkan cache hit dalam server proxy. Penelitian dilakukan dengan membuat sistem web server secara cluster atau terdistribusi dengan menggunakan delapan buah node web server. Sistem menghasilkan peningkatan latensi sebesar 90 % lebih baik dan peningkatan throughput sebesar 5,33 kali lebih baik.

Kata Kunci *- pemrosesan web; server proxy terdistribusi; HTTP; web QoS; penggantian cache*

I. PENDAHULUAN

Proxy Cache telah terbukti sebagai solusi penyelesaian masalah yang efisien untuk mengurangi latensi dan mengingkatkan kinerja pemrosesan web [1]. Sebuah *cache proxy* pada server w*eb* dapat melayani banyak pengguna pada waktu yang bersamaan. Pada saat server cache proxy web menerima permintaan dari pengguna, maka server pertama-tama mencari yang objek yang diminta dalam cache-nya [2]. Jika salinan cache tersebut ditemukan, maka proxy mengembalikannya data web tersebut ke penggunanya. Tetapi jika tidak ada, maka akan melakukan relay permintaan ke cache proxy lain yang bekerja sama atau server web tersebut, dan mengembalikan data yang baru ditemukan tersebut kepada pengguna dan menyalin data tersebut di cache-nya sendiri [3]. Cache proxy seringkali ditempatkan dekat dengan penggunanya.

Di sisi lain, pengganti cache proxy bekerja dengan cara yang sama sebagai cache proxy [4]. Perbedaan utamanya adalah pengganti cache biasanya terletak di dekat server web. Tujuan utama dari cache proxy adalah untuk mengurangi latensi akses web, sedangkan tugas dari pengganti cache adalah untuk mengurangi beban kerja server web dan berpotensi mengurangi latensi akses pengguna. Fungsi dari pengganti cache dapat digunakan untuk mereplikasi isi dari server web yang terkait di banyak lokasi web yang berbeda. Beberapa server web dinamis yang lambat, misalnya yang menggunakan PHP atau ASP, menggunakan pengganti cache.

Pengembangan cache proxy saat ini telah menjadi fokus utama dari penelitian *caching* web. Pendekatan berbasis cache proxy membuat latensi minimum dari server web dan jaringan protokol sehingga dapat langsung diterapkan dalam berbagai kondisi dari pengaturan aplikasi tanpa memodifikasi perilaku server. Tidak seperti pada cache browser, cache proxy melayani para pengguna yang berada di subnet yang sama. Hal ini menciptakan banyak peluang untuk mewujudkan potensi peningkatan kinerja dan pengurangan latensi *caching* web tersebut. Dari arsitektur sistem, cache proxy terletak di server proxy yang telah digunakan secara tradisional untuk tujuan lain, misalnya untuk keamanan jaringan, sehingga dapat membuat instalasi dan konfigurasi layanan cache relatif mudah dan lebih transparan [5]. Sebagian besar platform server web menggunakan proxy web yang dilengkapi dengan mekanisme dan hirarki dari cache.

[*)] Penulis korespondensi (Marvin Chandra Wijaya)
Email: marvinchw@gmail.com

Submitted: 18 November 2018; Revised: 26 July 2019; Accepted: 7 August 2019; Published: 31 January 2020

Mekanisme cache yang konsisten telah ditambahkan pada berbagai server proxy cache saat ini [6]. Berbagai macam metode dan algoritme server proxy cache diterapkan dengan menggunakan satu buah server. Untuk mengurangi latensi, digunakan teknik *prefetching* atau pengambil data awal dari pemrosesan web. Peningkatan *cache hit* dilakukan dengan menggunakan berbagai metode yang berbasis pada peningkatan kecepatan *prefetch* menggunakan satu komputer tunggal, di antaranya *historical based performance* [4], modifikasi dari metode LRU [7], modifikasi NMRU [8], *estimating pages* [9], semantik dinamik LFU [10], *low inter-reference recency set* (LIRS) [11], *adaptive replacement cache* (ARC) [12], [13], *clock with adaptive replacement* (CAR), *multi-queue* (MQ) [14], dan Pannier [15].

Peningkatan *cache hit* dapat diperoleh jika terdapat pengulangan-pengulangan permintaan data yang cukup sering. Namun, terdapat beberapa kegagalan atau tidak tercapainya peningkatan *cache hit* jika terdapat data yang acak/random. Sistem terdistribusi dapat digunakan untuk mengatasi hal ini dan dapat berfungsi dengan baik dalam peningkatan *cache hit*. Penelitian ini mengkaji peningkatan kinerja server web menggunakan sistem terdistribusi untuk meningkatkan kinerja server web, meliputi latensi, *throughput*, dan laju *cache hit*. Server proxy cache terdistribusi digunakan dalam perbaikan *cache hit* yang ada serta peningkatan kinerja latensi dan *throughput*-nya.

II. METODE PENELITIAN

A. Problem pada *cache replacement*

Implementasi proxy web didasarkan pada area cache yang terbatas pada media penyimpanan dibandingkan dengan jumlah objek web. Ketika sebuah konten pada area cache mendapat limit dari cache, sebuah aturan untuk penggantian cache diperlukan untuk melakukan penggantian cache yang perlu diperbaharui. Setiap objek web cache berkarateristik berdasarkan "kebekuan" (*staleness*) dari kebutuhan objek web. Kebekuan objek web dipengaruhi oleh pentingnya atau seringnya penggunaan dari objek tersebut berubah. Setiap implementasi proxy cache harus mempertimbangkan kebekuan dari setiap objek cache web yang dipengaruhi oleh popularitas dari objek web (Persamaan 1). Parameter $popularitas_i$ menunjukkan popularitas pada cache i, hit_i hit pada i dan hit_{total} hit keseluruhan.

Popularitas dari *cache replacement* tergantung dari hit pada bagian atau pada blok tertentu (i). Untuk mencari kebekuan dari suatu objek web dapat dihitung menggunakan rasio dari kebekuan dari objek web (Persamaan 2). Parameter c_i menunjukkan waktu objek tersebut dibutuhkan dan l_i waktu objek tersebut terakhir diperbaharui. Frekuensi dinamis dari cache pada objek i diperhitungkan dari rasio kebekuan (Persamaan 3). Parameter af_i menunjukkan banyaknya cache yang diakses sejak waktu terakhir objek web i diakses.

$$popularitas_i = \frac{hit_i}{hit_{total}} \quad (1)$$

$$Kebekuan = \frac{c_i - l_i}{now - c_i} \quad (2)$$

$$df_i = \frac{popularitas_i}{af_i} \quad (3)$$

Pengendalian *cache hit* dan *cache miss* menentukan kualitas dari QoS web. Hal ini menyebabkan web server harus mengambil data objek tersebut pada web server aslinya dan mengirimkan ke pengguna. Pada saat yang sama proxy web melakukan penggantian memori cache proxy dengan data yang baru. Sebuah objek yang dihapus atau digantikan di dalam cache disimbolkan dengan act_i (Persamaan 4). Penggantian cache yang efektif dilakukan dengan memaksimalkan nilai $\sum_{i=1}^{N} (act_i Kebekuan_i df_i)$.

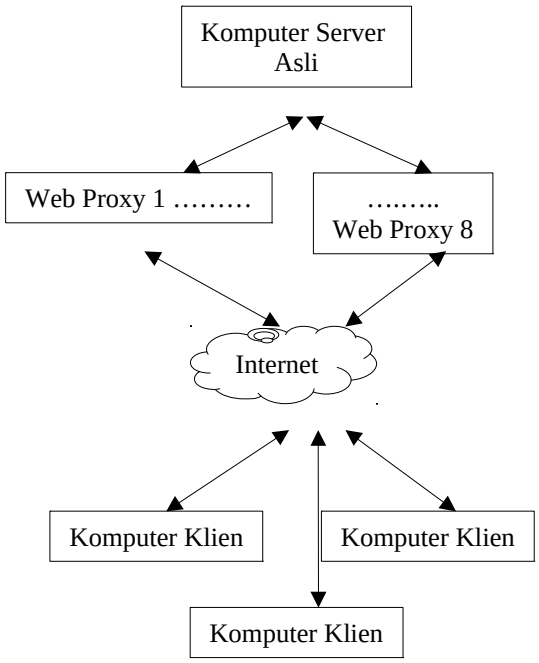
$$act_i = \begin{cases} 0, & jika\, obyek\, dihapus\, dari\, cache \\ 1, & lainnya \end{cases} \quad (4)$$

Tujuan pemecahan dari penggantian cache web adalah untuk mempertahankan cache yang tidak beku sesering mungkin yang diakses oleh objek web yang dibutuhkan oleh pengguna. Sebuah hasil yang maksimal dapat membuat kualitas penggunaan web menjadi lebih baik [14].

B. Sistem web proxy terdistribusi

Sistem yang digunakan dalam kajian ini adalah dengan menjalin kerjasama antara beberapa proxy web (proxy web terdistribusi) seperti pada Gambar 1. Setiap proxy web dapat saling bertukar data. Proxy web tidak selalu harus mengambil objek web dari server aslinya jika terjadi *miss* pada objek web. Hal ini memungkinkan terjadinya perbaikan latensi jika antar proxy web dapat berkomunikasi satu sama lainnya. Arsitektur ini bertujuan memperbaiki proses cache dari objek web. Setiap web proxy mempunyai data yang dapat berbeda atau sama tergantung dari kebekuan dari objek web yang terdapat web cache. Jumlah proxy web yang digunakan sampai delapan proxy.

Sistem antrian untuk pengendalian *cache hit* dirancang untuk menghasilkan sistem antrian proses cache dengan sistem terdistribusi (Algoritme 1). Struktur tabel akses yang digunakan dinyatakan dalam Tabel 1. Tabel akses digunakan untuk membuat suatu urutan pembukaan node-node yang dilakukan. Tabel akses tersebut terdiri dari Usia LRU, Posisi File, Kosong, dan Waktu pertama akses. Setiap blok pada *block replacement* mempunyai metadata sesuai dengan struktur pada tabel akses tersebut. Data pada tabel akses tersebut diproses sesuai algoritme yang telah ditentukan.



Gambar 1. Proses cache pada beberapa web proxy

III. HASIL DAN PEMBAHASAN

Sistem diimplementasikan pada delapan buah node *web-caching* yang terpisah. Delapan buah *web caching* tersebut dilakukan dengan menggunakan suatu sistem tertutup untuk melakukan serangkaian ujicoba permintaan data pada sebuah *browser*. Pengguna yang meminta permohonan data dibuat beberapa node *client* untuk membuat ujicoba tertutup tersebut dapat lebih meniru sistem yang lebih luas lagi. Sistem yang digunakan menggunakan dua buah komputer yang memiliki spesifikasi hampir identik sebagai server proxy web. Selain itu, juga digunakan sejumlah komputer lainnya yang berfungsi sebagai *client* yang melakukan permohonan permintaan data. Semua perhitungan digunakan dan dibandingkan dengan penggunaan satu buah komputer tunggal (yang diambil salah satu dari proxy web tersebut) dan semua hasil percobaan dinyatkaan berupa persentase terhadap kondisi semula (tanpa sistem kluster).

Latensi yang diujicobakan dilakukan untuk jumlah node yang dimulai lebih sedikit. Pada setiap kondisi penambahan jumlah node dilakukan pengukuran latensi. Latensi awal atau 100 % diambil pada saat *node* hanya satu buah saja. Gambar 2 menunjukkan bahwa dengan semakin banyak *node* atau *web proxy* yang terlibat, maka latensi dari kualitas web semakin membaik. Kinerja latensi meingkat seiring dengan penambahan node, seperti dalam [16], [17]. Sistem terdistribusi dengan 8 buah node menghasilkan perbaikan latensi sebesar 90 %. Latensi yang lebih baik berarti setiap perubahan permintaan data acak pada suatu web server dapat direspons lebih cepat oleh web server tersebut [14]. Laju perbaikan latensi ini semakin menurun terhadap penambahan jumlah node. Penambahaan node

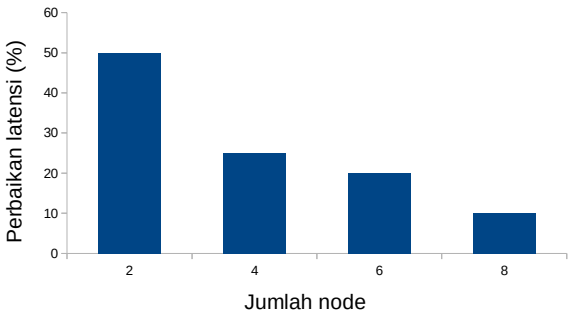
Algoritme 1. Sistem antrian cache dengan 8 proxy

```
Tabel akses dipersiapkan

Ulang (i=batas kumpulan kiri; i<=kumpulan kanan; i++) {
   Jika(tidak dalam pembuangan) {
      Jika(posisi i tidak kosong) {
         Jika(umur dari object i>batas LRU) {
            Hapus objek I dari cache;
            Waktu dari objek ada dalam cache =
               waktu sekarang – tabel.pertama diakses
             Waktu pertukaran sekarang -=ukuran objek
         }
      }
   }
   Jika tidak {
      Jika(i<=batas kumpulan kiri + 8)  {
         Jika(posisi i tidak kosong)  {
            Hapus objek;
            Waktu dari objek ada dalam cache =
               Waktu sekarang-tabel pertama diakses
            Waktu pertukaran sekarang -= ukuran objek
         }
      }
   }
}
```

Tabel 1. Struktur tabel akses

No.	Nama Field	Jenis
1.	Usia_LRU	Numeric
2.	Posisi_File	Numeric
3.	Kosong	Boolean
4.	Waktu_pertama_ akses	Time



Gambar 2. Perbaikan latensi terhadap jumlah node

berikutnya tidak membuat latensi meningkatkan lebih baik secara signifikan.

Throughput yang dihasilkan dari proxy web dengan sistem terdistribusi adalah peningkatan data yang ditransfer dengan peningkatan berbentuk linear (Gambar 3). *Throughput* diperoleh datanya dengan menghitung jumlah data yang dikirimkan oleh proxy web dalam suatu waktu yang sama. Peningkatan *throughput* pada node sebanyak 8 buah adalah 5,33 kali lebih baik dibandingkan dengan node tunggal sebesar 1500 Kbyte. Peningkatan *throughtput* membuat kinerja layanan web meingkat dengan menyediakan jumlah data lebih besar ke pengguna [14].

Penambahan ukuran cache yang tergabung dalam kluster dilakukan dan dilihat laju *cache hit*-nya. Gambar

4 menunjukkan bahwa *cache hit* meningkat pada setiap penambahan ukuran *cache*. Penambahan tersebut tidak linear terhadap ukuran *cache*. Hal ini disebabkan karena masukan data yang diberikan adalah acak / *random* sehingga objek web yang diminta pengguna dapat bervariasi dan menampung berbagai kemungkinan permintaan data dari banyak pengguna. Laju pertambahan *cache hit* semakin menurun seiring pertambahan ukuran cache.

Peningkatan *cache hit* secara langsung meningkatkan kecepatan dari server web. Hal tersebut dapat meningkatkan kualitas web yang langsung dirasakan oleh pengguna. Sistem dengan kluster proxy web yang terdistribusi dapat meningkatkan ukuran *cache* dan membagi tugas pada beberapa server proxy web sehingga meningkatkan ukuran cache dan selanjutnya menaikkan *cache hit* serta mengurangi *cache miss* seperti dalam [8]-[13]. Perbaikan lainnya adalah peningkatan IOPS di media penyimpan flash seperti dalam [15]. Penggunaan algoritme penggantian cache proxy dan sistem terdistribusi meningkatkan kinerja layanan web keseluruhan seperti dalam [14].

IV. KESIMPULAN

Penambahan jumlah node pada cache proxy web dapat meningkatkan kinerja layanan, yaitu meliputi latensi dan *throughput*. Pada sistem kluster dengan 8 buah node didapatkan latensi yang sudah lebih baik yaitu 90 % lebih cepat dan 5,33 kali *throughput* yang yang lebih baik. Penggunaan kluster proxy web dapat meningkatkan kinerja layanan web dibandingkan dengan sistem proxy web tunggal.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih atas pihak Universitas Kristen Maranatha yang telah banyak membantu dalam penyelesaian penelitian ini.

DAFTAR PUSTAKA

[1] A. Loutonen and K. Altis, "World Wide *Web* proxies," *Computer Network and ISDN Systems*, vol. 27, no. 2, pp. 147-154, 1994. doi: 10.1016/0169-7552(94)90128-7

[2] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching proxies: Limitations and potentials," in *Fourth International World Wide Web Conference*, Boston, USA, Dec. 1995, pp. 119 - 133

[3] J. Zhu, G. W. Yang, M. Hu, H-M. Shen, "A site-based *proxy cache*," *Journal of Computer Science and Technology*, vol. 18, no. 2, pp. 258-263, 2003.

[4] A. Vakali, "*Proxy cache* replacement algorithms: a history-based approach," *Journal World Wide Web*, vol. 4, no. 4, pp. 277-297, 2001.

[5] M-K. Liu, F-Y. Wang, and D. D. Zeng, "*Web* caching : a way to improve web QoS," *Journal Computer Science and Technology*, vol. 19, no. 2, pp. 113 -127, 2004.

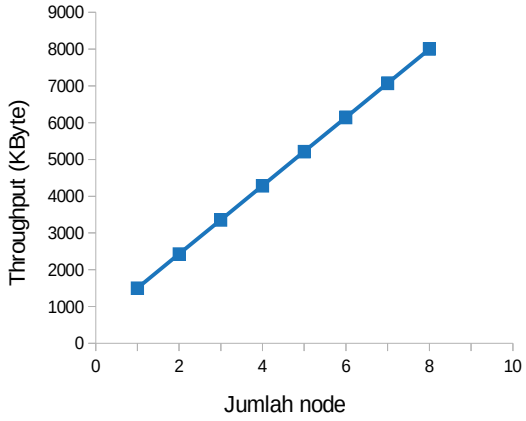
[6] Y. Liu. Y. Wang, and H. Du, "Strong cache consistency on world wide web," in *3rd International Conference on Advanced Computer Theory and Engineering*, Chengdu, China, Aug. 2010, pp. 62-65. doi: 10.1109/ICACTE.2010.5579237

[7] J. Boyar, M. R. Ehmsen, J. S. Kohrt, and K. S. Larsen, "A theoretical comparison of LRU and LRU-K," *Acta Informatica*, vol. 47, no. 7-8, 2010, pp. 359-374. doi: 10.1007/s00236-010-0123-6

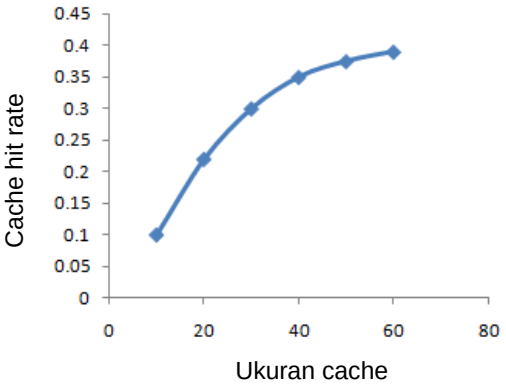
[8] S. Roy, "H-NMRU: an efficient *cache* replacement policy with low area," *International Journal of Parallel Programming*, vol. 48, no. 3-4, pp. 277-287, 2010. doi: 10.1007/s10766-010-0130-9

[9] A. Swami and K. B. Schiefer, "Estimating page fetches for index scans with finite LRU buffers," *The International Journal on Verly Large Data Bases*, vol. 4, no. 4, pp. 675-701. 1995. doi: 10.1007/BF01354879

[10] K. Geetha and N. A. Gounden, "Dynamic semantic lfu policy with victim tracer (DSLV): a customizing technique for client *cache*," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp 725-737, 2017. doi: 10.1007/s13369-016-2287-z



Gambar 3.*Throughput* sistem terhadap jumlah node



Gambar 4.*Cache hit* rate terhadap ukuran cache

[11] R. Rashkovits, "Preference-based content replacement using recency-latency tradeoff," *World Wide Web*, vol. 19, no. 3, pp. 323-350, 2016. doi: 10.1007/s11280-014-0313-

[12] A. P. Negrao, C. Roque, P. Ferreira, and L. Veiga, "An adaptive semantics-aware replacement algorithm for *web* caching," *Journal of Internet Services and Application*, vol. 6, no. 4, 2015. doi: 10.1186/s13174-015-0018-4

[13] A. M. K. Cheng and Z. Zhang, "Improving web server performance with adaptive proxy caching in soft real-time mobile applications," *The journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 47, no. 2, pp. 103-115, 2007. doi: 10.1007/s11265-006-0021-x

[14] N. Xiao, Y-J. Zhao, F. Liu, Z-G. Chen, "Dual queues *cache* replacement algorithm based on sequentiality detection," *Science China Information Sciences*, vol. 55, no. 1, pp. 191-199, 2012. doi: 10.1007/s11432-011-4213-z

[15] C. Li, P. Shilane, F. Douglis, and G. Wallace, "Pannier: a container-based flash *cache* for compound objects," in *16th Annual Middleware Conference*, Vancouver, Canada, Nov. 2015, pp. 50-62. doi: 10.1145/2814576.2814734

[16] K. C. Tsui, J. Liu, and M. J. Kaiser, "Self-organized load balancing in proxy servers: algorithms and performance," *Journal of Intelligent Information Systems*, vol. 20, no. 1, pp. 31-50, 2003. doi: 10.1023/A:1020943021842

[17] I. Ivanisenko, "Methods and Algorithms of Load balancing," *International Journal Information Technologies & Knowledge*, vol. 9, no. 4, pp. 340-375, 2015.