Desain Metode PrefetchCache untuk Peningkatan Kinerja Aplikasi Web

Design of the PrefetchCache Method for Improving Web Application Performance

Huda M. Elmatsani
Badan Pengkajian dan Penerapan Teknologi
E-mail: huda.mohamad@bppt.go.id

Abstrak

Internet dan teknologi web telah menyediakan sarana yang dapat diandalkan untuk berbagi data dan aplikasi. Namun, latensi pemuatan data masih menjadi kendala. Dalam beberapa literatur, metode cache terbukti meningkatkan waktu penyajian halaman web dengan mengurangi proses permintaan pada database. Namun, metode ini tidak mencegah web mengakses server untuk mengambil data cache dalam database, latensi dalam memuat data masih terjadi. Makalah ini menyajikan metode prefetch dengan mengubah data menjadi file data yang diformat JSON sebagai cache, data diperoleh dengan mudah tanpa harus melakukan proses permintaan. Hasil pengujian menunjukkan bahwa aplikasi web menjadi lebih responsif dalam menyajikan data.

Kata kunci: prefetch, cache, latensi, web, kinerja, JSON

Abstract

Internet and web technologies have provided a reliable means of sharing data and applications. Yet, latency on loading data is still a constraint. In some literature, the cache method is proven to increase the time of presentation of web pages by reducing the query process on the database. However, this method does not prevent the web from accessing the server to retrieve cache data in the database, the latency on loading data still occurs. This paper presents a prefetch method by converting data into JSON formatted data files as cache, data obtained easily without having to perform the query process. Test results show that web applications become more responsive in presenting data.

Keywords: prefetch, cache, latency, web, performance, JSON

1. PENDAHULUAN

Internet telah memegang peranan penting dalam kehidupan manusia [1], setidaknya dalam bisnis dan organisasi [2], dan terutama penggunaan teknologi web pada pengembangan e-commerce dan e-business [3], yang mengilhami bagaimana aplikasi web dibangun dan diimplementasikan di jaringan internet. Teknologi web membuat pengembangan aplikasi lebih mudah untuk dilaksanakan dan teknologi internet membuatnya lebih cepat untuk disebarluaskan dalam pemanfaatannya. Selain itu, aplikasi berbasis web menyediakan cara yang menguntungkan untuk mengakses, memodifikasi, dan mengelola data [4]. Keuntungan ini telah membuat banyak perusahaan bermigrasi dari aplikasi desktop ke aplikasi web, terutama dengan kemajuan infrastruktur internet dan kehadiran komputasi awan. Dalam beberapa tahun terakhir, banyak perusahaan mulai memindahkan sistem, aplikasi, layanan, dan data mereka ke lingkungan cloud [5] di mana aplikasi web ditempatkan.

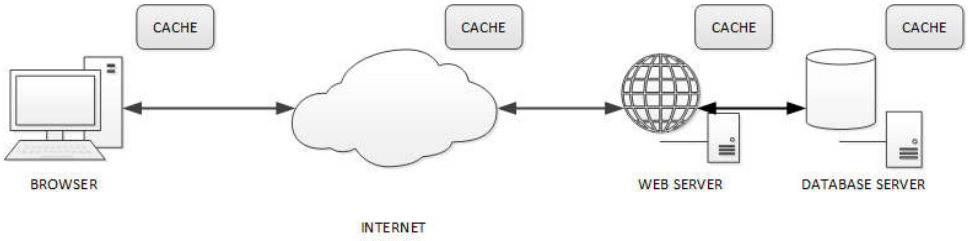
Aplikasi berbasis web adalah perangkat lunak yang dapat digunakan melalui internet. Aplikasi dan basis data terletak di server web dan bukan di komputer desktop lokal. Aplikasi berbasis web adalah cara terbaik untuk memanfaatkan teknologi saat ini untuk meningkatkan produktivitas dan efisiensi bisnis [6]. Namun, karena harus diakses melalui internet, aplikasi web sangat tergantung pada bandwidth jaringan. Tingginya trafik dan terbatasnya bandwidth

internet menyebabkan gangguan pada kinerja web. Hal ini ditunjukkan oleh keterlambatan aplikasi web dalam merespon aktivitas pengguna yang disebabkan oleh keterlambatan aplikasi dalam mengakses server dan mentransfer data yang diminta.

Di sisi lain, salah satu faktor kesuksesan aplikasi web adalah seberapa cepat aplikasi merespons permintaan dan memberikan pengalaman yang memuaskan bagi pengguna. Penelitian yang dilakukan oleh Gomez dan Akamai [7] menggarisbawahi bahwa pengunjung web berharap situs web dapat merespons lebih cepat dari 2 detik, dan jika waktu tunda melebihi 6 detik membuat situs web ditinggalkan oleh pengunjung. Oleh karena itu, dalam mengembangkan aplikasi web, penting untuk mencatat seberapa cepat halaman web merespon interaksi pengguna. Kinerja adalah persyaratan non-fungsional yang penting yang mempengaruhi pengalaman pengguna dan memiliki dampak besar termasuk keunggulan kompetitif, pendapatan online, dan trafik situs web [7]. Situs web yang lebih lambat akan kehilangan pengunjung dan akan kalah dari persaingan.

Ada dua faktor yang mempengaruhi kinerja aplikasi web; platform dan aplikasi web itu sendiri [8]. Platform termasuk server tempat aplikasi berada dan jaringan yang menghubungkan aplikasi ke pengguna. Penyediaan kapasitas server dan besarnya bandwidth diperlukan untuk meningkatkan kinerja, tetapi trafik tinggi membuatnya tidak mencukupi.

Shivakumar [9] menyarankan pedoman untuk meningkatkan kinerja dan beberapa di antaranya adalah pemuatan data secara on-demand, penggunaan ajax, dan caching. Pemuatan data secara on-demand mengurangi konsumsi waktu dan penggunaan memori sehingga mengoptimalkan pengiriman konten. Karena hanya sebagian kecil dari halaman web yang dimuat terlebih dahulu, maka waktu yang dibutuhkan lebih sedikit. Semua ini meningkatkan pengalaman pengguna saat konten yang diminta ditampilkan tanpa waktu tunda [10]. Selanjutanya [9] menyarankan pemuatan data dengan penggunaan teknologi AJAX (Asynchronous Javascript dan XML). Ajax adalah teknologi web yang menyederhanakan penerapan halaman dinamis dan interaktif dengan mengganti bagian konten web dengan cara yang tidak sinkron, Ajax dapat mengganti data atau informasi pada halaman web tanpa harus memuat ulang seluruh tampilan halaman [11], hal tersebut merupakan keunggulan ajax, memberikan peningkatan kinerja bandwidth [12] dengan memisahkan logika presentasi dari data. Seperti disebutkan dalam [9], cara lain untuk meningkatkan kinerja aplikasi web adalah dengan pemanfaatan cache. Caching adalah sebuah metode yang dapat mengurangi kedua beban dengan mengirimkan salinan file atau data lebih dekat ke klien yang menggunakannya [13].



Gambar 1. Cache dapat disimpan pada browser komputer, server web, server database atau proxy

Cache dirancang dan diimplementasikan untuk meningkatkan kinerja situs web atau aplikasi web. Cache berfungsi dengan menyimpan data sementara, jadi untuk halaman web dan objek web seperti gambar, pdf, XML atau file lainnya mudah diakses lagi [14]. Cache dapat disimpan di komputer klien (browser), server web, server database atau pada server proxy. Gambar 1 menggambarkan proses ketika pengguna melakukan permintaan ke server. Untuk mengurangi latensi pada halaman web, browser menggunakan cache berbasis klien yang menyimpan halaman web yang terakhir diakses [15]. Browser memeriksa apakah ada cache untuk permintaan itu, jika tidak, permintaan diteruskan ke server proxy, dan server proxy memproses permintaan itu, hingga permintaan tersebut mencapai server web tujuan.

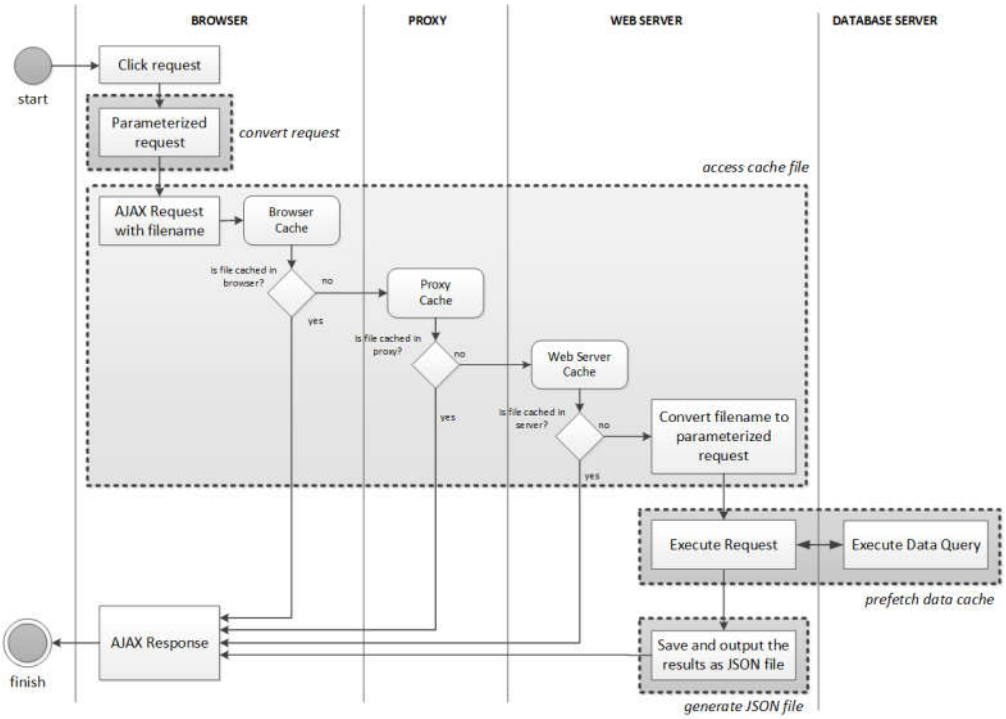
Cache browser memiliki batasan dalam ukuran, dan berdasarkan pada Ali et al [16], server proxy adalah solusi paling baik dalam meningkatkan kinerja web yang menyediakan akses lebih dekat ke objek web yang diminta. Sedekat cache diambil maka secepatnya juga respon aplikasi web. Dari uraian proses cache ini, ada tiga keuntungan yang dapat diperoleh sebagai berikut: aplikasi lebih responsif, mengurangi trafik internet dan mengurangi beban server dalam memproses permintaan [15].

Dalam kasus web meminta data terbaru, proses permintaan membutuhkan waktu lebih lama daripada proses menampilkan objek statis seperti gambar. Salah satu teknik caching di server adalah menerapkan basis data cache. Server basis data cache terletak di tingkat menengah bersama dengan server web dan dibuat untuk melayani permintaan membaca data. Biasanya ini sebagai hasil dari data transaksi (OLTP) yang bukan waktu nyata. Metode ini memberikan hasil yang signifikan dalam meningkatkan kinerja server database. Namun, cache database tidak mengurangi trafik permintaan data dari browser ke server.

Latensi pada akses data dapat dikurangi jika data yang akan diakses dapat diprediksi sehingga pre-fetch dapat dilakukan [17]. Pre-fetching bekerja dengan menyiapkan objek web sebelum objek diminta oleh pengguna. Ini sering digunakan di situs web untuk menampilkan galeri gambar atau peta. Gambar telah disiapkan secara diam-diam sebelum pengguna membukanya, jadi ketika pengguna mengklik tautan gambar, gambar langsung muncul di layar monitor tanpa mengunduh dari server.

Teknik caching dapat dipraktikkan dalam menyajikan data. Data yang sering diminta atau permintaan rumit dan memakan waktu disiapkan dan disimpan dalam basis data cache. Jadi, tidak setiap permintaan data harus mengakses server database, yang pada gilirannya, mengurangi beban server secara signifikan. Namun demikian, data yang diminta ke server masih melalui proses permintaan ke server sehingga latensi masih terjadi saat data diperlukan.

Pre-fetching adalah proses pengaturan konten web yang akan diakses oleh pengguna web, ketika konten diakses, pengguna langsung mendapatkan konten, tidak perlu menunggu lama. Ada dua metode pre-fetching, yaitu pengambilan informasi awal dan prediksi pre-fetching. Informasi pre-fetching digunakan jika sumber daya yang diperlukan sudah diketahui dengan jelas, sedangkan prediksi pre-fetching digunakan jika sumber daya yang diperlukan hanya dapat diperkirakan berdasarkan referensi atau sejarah sebelumnya [18]. Beberapa



Gambar 2. Desain alur kerja metode PrefetchCache – menyiapkan data dan menyimpannya sebagai file JSON

pendekatan telah diperkenalkan untuk menerapkan teknik pre-fetching, seperti penambangan web dan pre-fetching berbasis hyperlink. Berbagai teknik digunakan untuk meningkatkan akurasi prediksi akses pengguna sehingga pre-fetching menjadi lebih efektif [19].

Strategi yang diusulkan dalam penelitian ini menggunakan ajax untuk meminta data dan melakukan pre-fetching hasil kueri data dan menyimpannya sebagai cache file, bukan cache database dengan format JSON. JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan yang memiliki efisiensi parsing lebih tinggi dari XML dan karena kekurangan XML, JSON telah banyak digunakan dalam aplikasi web, terutama dalam layanan berbasis ajax [20]. Dengan menyimpannya sebagai file JSON, maka diharapkan file yang di-cache dapat disimpan hingga tingkat browser sehingga akses data bisa sangat cepat karena tidak perlu melalui jaringan dan pada saat yang sama lalu lintas permintaan dan respons berkurang.

2. METODE PENELITIAN

Penelitian ini mengusulkan metode pre-fetching dan caching yang selanjutnya disebut PrefetchCache. Metode ini bertujuan untuk meningkatkan kinerja aplikasi web melalui proses pre-fetching dan proses cache yang terdiri dari: konversi *request*, pengambilan data awal, pembuatan file dan akses data. Ajax digunakan dalam metode PrefetchCache dengan tujuan untuk menjalankan proses penyajian data pada latar belakang tanpa mengganggu interaksi pengguna di browser. Dengan metode ini, aplikasi secara diam-diam menyiapkan data terlebih dahulu dan disimpan sebagai file JSON yang siap menunggu permintaan. Gambar 2 menjelaskan desain proses metode PrefetchCache.

Desain PrefetchCache dapat digambarkan dalam algoritma sebagai berikut:

a. Pengguna menginginkan data spesifik dalam bentuk permintaan berparameter.
b. Sebuah skrip mengonversi permintaan ke dalam bentuk panggilan terhadap nama file JSON.
c. Ajax mengirimkan permintaan melalui browser ke server. Browser memeriksa apakah ada cache dari file JSON itu untuk ditampilkan, jika tidak, permintaan dipindahkan ke proksi terdekat untuk memeriksa apakah ada cache dari file JSON itu, jika tidak, permintaan dipindahkan ke server untuk memeriksa apakah ada cache untuk file JSON itu, jika tidak, server mengganti nama file JSON menjadi permintaan berparameter.
d. Server memproses permintaan dan menyimpan hasilnya sebagai file JSON dan menyimpannya sebagai cache.
e. Server mengirim data JSON ke browser melalui Ajax.
f. Ajax menampilkan data di halaman web.

2.1 Konversi Permintaan

Konversi diperlukan saat pengguna mengakses data tertentu. Biasanya, permintaan data dilakukan dengan menentukan nilai-nilai parameter permintaan, dan konversi diperlukan untuk memanggil nama file yang berisi parameter yang ditentukan. Misalnya permintaan string kueri dalam Listing 1 diubah menjadi nama file seperti employ_pns_active.json. Ekstensi file json digunakan untuk file berformat JSON.

```
POST /path/employ HTTP/1.0
kategori=pns&status=aktif
```

List. 1 Permintaan data ke server menggunakan URL dan query string

Untuk memfasilitasi penamaan file yang secara eksplisit mengandung nilai parameter,

nama file tersegmentasi ke dalam sejumlah parameter. Pada contoh di atas, ada dua parameter sehingga ada dua segmentasi setelah nama entitas, segmen pertama adalah pns yang merupakan nilai dari kategori parameter dan segmen kedua adalah aktif dan itu adalah nilai status parameter . Segmentasi ini harus didefinisikan dan didokumentasikan dengan benar untuk menghindari kesalahan dalam mengubah permintaan menjadi data yang diinginkan.

Dalam aplikasi web berbasis Ajax, bagian konten dari halaman web dapat diperbarui tanpa harus me-refresh seluruh halaman, dan penyajian data dapat dilakukan dengan pemuatan data berdasarkan permintaan yang diminta ke server seperti ditunjukkan pada Listing 2.

```
1       ajax.request ({
                url: ‘employ.php’,
                params: {kat:’pns’, status:’aktif’},
                success: function (response, opts) {}
5       });
```

List. 2 Permintaan data ke server menggunakan URL dan parameter

URL digunakan untuk mengakses halaman web. Untuk menampilkan data tertentu, parameter ditambahkan ke permintaan dengan kunci: format nilai. Dengan PrefetchCache, permintaan yang semula berbentuk URL dan parameternya dikonversi ke nama file JSON, seperti yang ditunjukkan pada Listing 3.

```
1       ajax.request({
                url: ‘employ_pns_active.json’,
                success: function(response, opts){}
4       });
```

List. 3 The parameters for the request are converted to the JSON file name section

2.2 Akses Cache File

Penelitian ini menggunakan aturan *rewrite* pada *htaccess* untuk mengarahkan permintaan jika file akses yang dimaksud tidak ditemukan (karena belum diambil sebelumnya). Konfigurasi .htaccess mentransfer file yang tidak ada ke permintaan ke proses pre-fetch.

```
1       RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule ^page (. *) $
4               prefetch.php?file= page_$1 [L]
```

List. 4 Konfigurasi .htaccess untuk mengarahkan permintaan yang tidak ditemukan

Server melakukan pre-fetching untuk menyediakan file yang dimaksud karena file tersebut tersedia untuk akses selanjutnya. Hal ini dilakukan dengan mengubah kembali nama file menjadi kueri parameter dan kemudian menjalankan kueri itu.

Konfigurasi dalam Listing 4 berarti bahwa jika file dengan halaman nama _... tidak ditemukan, maka akan diarahkan ke file prefetch.php dengan halaman nama _... sebagai parameter. Prefetch.php adalah file yang akan memproses pengambilan data awal dan pembuatan file.

```
<FilesMatch "\.(html?|php|css|js|json)$">
SetOutputFilter DEFLATE
</FilesMatch>
```

List. 5 Konfigurasi .htaccess untuk mengkompres file.

Listing 5 menunjukkan penggunaan DEFLATE dalam konfigurasi .htaccess. DEFLATE adalah filter kompresi untuk mempercepat akses file [11], termasuk file JSON yang digunakan dalam PrefetchCache.

2.3 Pre-fetching Data

Dalam menjalankan proses kueri yang rumit yang melibatkan beberapa tabel untuk menghasilkan data pivot, server database menghabiskan waktu. Selain itu, ini membuat server bekerja ekstra dan proses panjang ini menyebabkan gangguan pada interaksi pengguna. Pra-pengambilan data adalah solusi tempat data yang dihasilkan dari proses kueri disimpan sebagai cache. Ketika ada permintaan yang sama datang, server merespons dengan memberikan data yang diambil dari cache.

Secara umum, cache data disimpan dalam cache database atau disimpan dalam memori server. Alih-alih menyimpannya sebagai cache data, metode PrefetchCache menyimpan data itu dalam bentuk file. Selain itu, file dibuat berdasarkan frekuensi penggunaan data dalam aplikasi dan berdasarkan permintaan pengguna.

Untuk data yang tidak banyak berubah dan sering ditampilkan dalam aplikasi seperti dalam laporan dan sekarang infografik, maka file JSON diambil terlebih dahulu. Adapun data yang diperlukan, pre-fetch khusus pengguna dilakukan ketika ada permintaan dari pengguna.

Listing 6 menjelaskan contoh menghasilkan file untuk dataset. Baris 1 adalah permintaan untuk dieksekusi dan baris 11 adalah file yang datanya akan disimpan.

```
str_query = "
SELECT *
FROM artikel
WHERE category=sains
        AND tahun='2018"';

stmt = db->query( str_query );
data = stmt->fetchAll();

page = 'page_sains_2018.json';
fwrite(page);
```

List. 6 Results of queries fetched into a file

Listing 6 menjelaskan bagaimana json_encode mengubah hasil kueri menjadi data format JSON. Data JSON disimpan sebagai file ekstensi JSON dengan memberi nama berdasarkan urutan dan nilai parameternya.

2.4 Akses Data

Biasanya, query string digunakan untuk menampilkan data tertentu sebagaimana ditunjukkan Listing 7.

```
1    http://webapp/employ?cat=pns&status=active
```

List. 7 Query string digunakan dalam URL untuk mengakses data tertentu

Dalam metode Ajax, parameter digunakan untuk mengakses data seperti yang ditunjukkan pada Listing 8.

```
1    Ajax.request({
         url: 'employ',
         params: {cat:'pns'; status:'active'
4    });
```

List. 8 Parameter digunakan pada Ajax untuk meminta data tertentu

Dengan PrefetchCache, permintaan terhadap data tidak lagi dalam bentuk URL parameter, tetapi digantikan oleh nama file tersegmentasi JSON, yang ditunjukkan pada Listing 9.

```
1    Ajax.request({
         url: 'employee_pns_active.json',
     });
```

List. 9 URL berparameter dikonversi menjadi sebuah nama file

Seperti diilustrasikan pada Listing 9, proses akan lebih cepat jika data yang diperlukan sudah dalam bentuk file dan ada di cache terdekat.

2.5 Pengujian

Untuk melakukan pengujian, aplikasi web ditempatkan pada server web yang dapat diakses melalui internet. Dua komputer klien digunakan untuk mengakses web. Komputer pertama sebagai akses pertama, dan komputer kedua sebagai akses berikutnya. Tujuan menggunakan komputer pertama adalah untuk mengetahui akses pertama kali, di mana data yang diminta baru disediakan oleh server dan belum tersedia di cache, sedangkan komputer kedua digunakan untuk akses selanjutnya dengan asumsi bahwa data yang diminta telah disimpan dalam cache di proksi Penyedia Layanan Internet (ISP), sehingga diharapkan akses oleh komputer kedua lebih cepat dari komputer pertama seperti yang diprediksi oleh metode PrefetchCache. Untuk mengukur latensi permintaan data, browser FireFox dipasang di komputer-komputer itu, browser ini memiliki alat yang menyediakan informasi tentang waktu akses untuk setiap permintaan data. Dua skenario dilakukan dalam mengukur latensi, yaitu: mengukur latensi berdasarkan ukuran data dan jumlah data.

3. HASIL DAN PEMBAHASAN

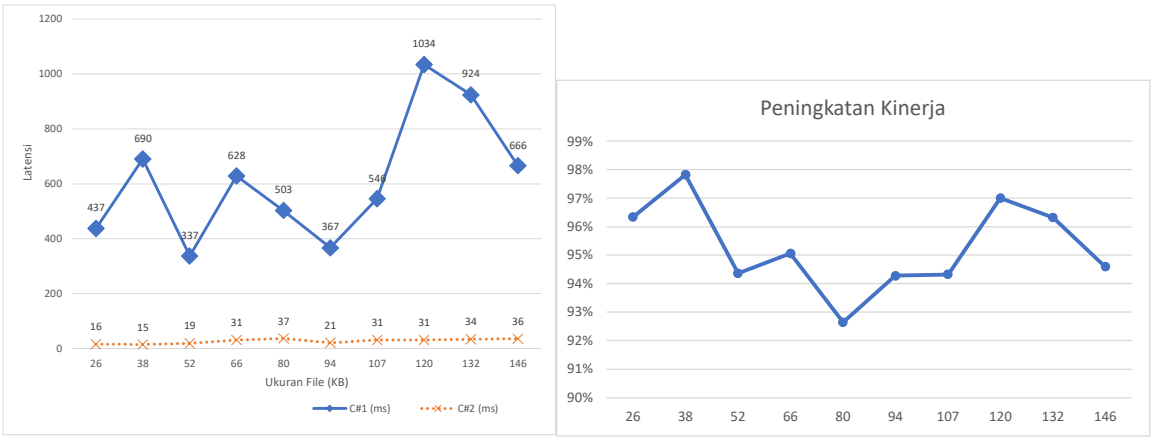
Dalam percobaan pertama, sejumlah data diminta ke server. Beberapa parameter ditentukan untuk melihat perbedaan antara mengakses data yang tersedia sebagai file cache dan mengakses data yang belum tersedia dalam cache.

Tabel 1. Akses Data dengan Variasi Ukuran File (KB)

No.	Size (KB)	C#1 (md)	C#2 (md)	C#3 (md)
1	25,86	437	16	0
2	38,08	690	15	0
3	51,89	337	19	0
4	66,11	628	31	0
5	79,44	503	37	0
6	93,51	367	21	0
7	106,55	546	31	0
8	119,54	1034	31	0
9	132,09	924	34	0
10	146,37	666	36	0

* md = milidetik

Tabel 1 menunjukkan hasil pengukuran latensi penyajian data pada aplikasi web dengan memberikan variasi dalam ukuran data yang diakses. C#1 adalah hasil akses pertama dari komputer #1 (dalam milidetik) yang mengakses server ketika file cache belum tersedia, C#2 adalah akses dari komputer #2 ketika file cache tersedia pada server proxy ISP dan C#3 adalah akses berikutnya dari kedua komputer ketika file cache tersedia di browser.



Gambar 3. Grafik hasil pengukuran latensi (kiri) dan peningkatan kinerja (kanan)

Grafik pada gambar 3 menyajikan latensi C#1 dan C#2 yang menunjukkan bahwa latensi atau waktu akses pada C#2 sangat rendah, berkisar antara 16 sampai dengan 36 milidetik dan memberikan peningkatan kinerja yang signifikan, peningkatan kinerja sebagaimana ditunjukkan pada gambar 3 sebalah kanan mencapai rata-rata 95%. Akses pertama dari C#1, secara umum, memakan waktu lebih lama karena file yang diakses belum tersedia dan server memerlukan waktu lebih lama untuk mengeksekusi kueri pada database, sedangkan pada akses kedua yang ditunjukkan pada kolom C#2, latensi dalam penyajian data mengalami penurunan mencapai 15 hingga 37 milidetik. Adapun pada kolom C#3 adalah akses ketika cache sudah tersedia di browser, sehingga penyajian data tidak mengalami latensi, atau waktu akses sama dengan 0 detik. Dengan berkurangnya latensi hingga 95% dalam menyajikan data, aplikasi web memberi pengguna perasaan seperti menggunakan aplikasi desktop. Pengalaman ini akan membuat kenyamanan pengguna dalam menggunakan aplikasi web.

Tabel 2 menunjukkan bahwa dengan peningkatan jumlah record data maka akses membutuhkan waktu respons yang lebih lama, tetapi dalam akses file yang di-cache, ada

Tabel 2. Akses Data dengan Variasi Jumlah Record Data

No.	Record	C#1 (md)	C#2 (md)	C#3 (md)
1	500	187	15	0
2	1000	706	31	0
3	1500	317	31	0
4	2000	717	46	0
5	2500	749	47	0
6	3000	1158	47	0
7	3500	1393	47	0
8	4000	752	78	0
9	4500	1179	78	0
10	5000	1274	78	0

* md = milidetik

peningkatan yang signifikan dalam waktu respons. Untuk hitungan record antara 4000 dan 5000, latensi hanya 78 milidetik, atau tidak sampai sepersepuluh detik. Jika ini diterapkan, maka aplikasi dapat menampilkan semua data yang diminta tanpa khawatir masalah latensi dalam pemuatan data. Berkat file cache yang tersedia.

Selain ukuran file dan jumlah record, eksekusi query SQL adalah salah satu faktor yang mempengaruhi latensi aplikasi web. SQL kompleks yang melibatkan beberapa sub-kueri dan join dapat menyebabkan waktu eksekusi yang lambat. PrefetchCache mencegah eksekusi berulang dan memberikan peningkatan kinerja seperti yang ditunjukkan pada tabel 1 dan 2.

Berdasarkan hasil pengujian pada tabel 1 dan 2, metode PrefetchCache menunjukkan peningkatan yang signifikan dalam kinerja, hal ini karena mengakses file cache yang ada di proxy atau di browser tidak memerlukan akses ke server yang biasanya menyebabkan keterlambatan. Apalagi jika cache sudah ada di browser, tidak ada lagi latensi. Di sisi pengguna, aplikasi terasa lebih responsif, ketika pengguna mengakses data tertentu, aplikasi segera menampilkan data yang diinginkan, seperti diungkapkan oleh [8], kinerja meningkat dengan memindahkan file lebih dekat ke pengguna melalui mekanisme cache.

4. KESIMPULAN DAN SARAN

Metode PrefetchCache meningkatkan kinerja aplikasi web. PrefetchCache mengubah setiap permintaan yang diparameterisasi ke web menjadi permintaan ke file cache yang statis melalui mekanisme respons permintaan Ajax. Cara ini secara signifikan mengurangi pemrosesan yang menghabiskan waktu dan sumber daya server karena setiap data tidak lagi secara langsung mengakses ke server melainkan diperiksa terlebih dahulu keberadaannya dalam cache. Strategi ini meningkatkan respons aplikasi web dan sangat mudah diimplementasikan karena hanya mengubah kueri parameterisasi menjadi nama file yang disegmentasi.

Penelitian lebih lanjut diperlukan untuk mengelola usia cache dari setiap file yang diambil sebelumnya sedemikian rupa sehingga file-file cache selalu dapat menampilkan data/informasi terbaru.

DAFTAR PUSTAKA

[1] E.-I. Apăvăloaie, "The Impact of the Internet on the Business Environment," *Procedia Econ. Financ.*, vol. 15, no. 14, pp. 951–958, 2014.

[2] E. C. Ehman *et al.*, "A Literature Review: Website Design and User Engagement Renee," vol. 46, no. 5, pp. 1247–1262, 2017.

[3] J. Benitez, Y. Chen, T. S. H. Teo, and A. Ajamieh, "Evolution of the impact of e-

business technology on operational competence and firm profitability: A panel data investigation," *Inf. Manag.*, vol. 55, no. 1, pp. 120–130, 2018.

[4] P. Soontornpipit, C. Viwatwongkasem, C. Taratep, W. Teerawat, and P. Vanitchatchavan, "Development of the Electronic Surveillance Monitoring System on Web Applications," *Procedia Comput. Sci.*, vol. 86, no. March, pp. 244–247, 2016.

[5] M. F. Gholami, F. Daneshgar, G. Low, and G. Beydoun, "Cloud migration process—A survey, evaluation framework, and open challenges," *J. Syst. Softw.*, vol. 120, pp. 31–69, 2016.

[6] M. R. Hasan, M. I. Ibrahimy, S. M. A. Motakabber, M. M. Ferdaus, M. N. H. Khan, and M. G. Mostafa, "Development of a Web-based financial application System," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 53, p. 012080, 2013.

[7] Gomez Inc, "Why Web Performance Matters : Is Your Site Driving Customers Away?," 2011. [Online]. Available: http://www.mcrinc.com/Documents/Newsletters/201110_why_web_performance_matters.pdf. [Accessed: 01-Dec-2017].

[8] A. D'Ambrogio and G. Iazeolla, "Steps Towards The Automatic Production Of Performance Models Of Web Applications," *J. Comput. Networks*, vol. 41, no. 1, pp. 29–39, 2003.

[9] S. K. Shivakumar, "3 – Optimizing Performance of Enterprise Web Application," in *Architecting High Performing, Scalable and Available Enterprise Web Applications*, New York: Morgan Kaufmann, 2015, pp. 101–141.

[10] Ishaniaggarwal, "What is Lazy Loading?" [Online]. Available: https://www.geeksforgeeks.org/what-is-lazy-loading/. [Accessed: 03-Aug-2019].

[11] Y. Jin, "Research and application of Ajax technology in Web Development," *Proc. - 2014 IEEE Work. Electron. Comput. Appl. IWECA 2014*, pp. 256–260, 2014.

[12] Y. Jie, L. Zhongwei, and L. Fang, "The impact of Ajax on network performance," *J. China Univ. Posts Telecommun.*, vol. 14, pp. 32–34, 2007.

[13] D. Sachan and D. Rao, "Performance Improvement of Web Caching Page Replacement Algorithms," vol. 5, no. 3, pp. 3112–3115, 2014.

[14] M. Gupta and A. Garg, "A Comparative Analysis of Content Delivery Network and Other Techniques for Web Content Delivery," *Int. J. Serv. Sci. Manag. Eng. Technol.*, vol. 6, no. December, pp. 43–58, 2015.

[15] A. P. Pons, "Web-application centric object prefetching," *J. Syst. Softw.*, vol. 67, no. 3, pp. 193–200, 2003.

[16] W. Ali, S. M. Shamsuddin, and A. S. Ismail, "Intelligent Naïve Bayes-based approaches for Web proxy caching," *Knowledge-Based Syst.*, vol. 31, pp. 162–175, 2012.

[17] E. Monteiro, C. Costa, J. L. Oliveira, D. Campos, and L. B. Silva, "Caching and prefetching images in a web-based DICOM viewer," *Proc. - IEEE Symp. Comput. Med. Syst.*, vol. 2016-Augus, pp. 241–246, 2016.

[18] M. Y. Katsaros D., "Cache Management for Web-Powered Databases," *Web-Powered Databases*, pp. 201–242, 2002.

[19] W. G. Teng, C. Y. Chang, and M. S. Chen, "Integrating web caching and web prefetching in client-side proxies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 5, pp. 444–455, 2005.

[20] K. Afsari, C. M. Eastman, and D. Castro-lacouture, "Automation in Construction JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange," *Autom. Constr.*, vol. 77, pp. 24–51, 2017.