社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

レンダリング結果のキャッシュによるWebパフォーマンス向上手法

中野　雄介[†,††]　上山　憲昭[†,††]　塩本　公平[††]　長谷川　剛[†††]　村田　正幸[†]

宮原　秀夫[†]

†† 日本電信電話株式会社 NTT ネットワーク基盤技術研究所　〒 180–8585 東京都武蔵野市緑町 3–9–11
††† 大阪大学サイバーメディアセンター　〒 560–0043 大阪府豊中市待兼山町 1–32
† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田 2–1
E-mail: †{nakano.yuusuke,kamiyama.noriaki,murata,miyahara}@ist.osaka-u.ac.jp,
††shiomoto.kohei@lab.ntt.co.jp, †††hasegawa@cmc.osaka-u.ac.jp

あらまし　近年，Web パフォーマンスの重要性が注目されている．Web パフォーマンスとは，Web ページの表示にかかる時間であり，ユーザは Web パフォーマンスが低い Web ページから離れる傾向にある．そこで，本研究では Web パフォーマンス低下の原因を特定するため，PlanetLab を用い，世界 4 地点のホスト上で Web ブラウザを動作させることで，実際に Web ページの表示の際にダウンロードされるオブジェクトのダウンロード時間を測定した．この結果，Web ブラウザ内での待ち時間が Web パフォーマンス低下の原因であることを明らかした．本稿では，この様な Web ブラウザ内での待ち時間を削減するためのレンダリング結果のキャッシュによる Web パフォーマンスの向上手法を提案する．提案手法は，これまで端末内のブラウザで行われてきたレンダリングの処理を肩代わりする機能をネットワーク内に配置し，本機能によって生成されるレンダリング結果をキャッシュしておくことで，キャッシュヒット時に，レンダリングにかかる時間を削減する．提案手法の有効性の評価の結果，RTT が長い地域や Web ページにおいて，提案手法は有効であることがわかった．また，この様な地域や Web ページにおいては，動的なオブジェクトの割合が 8 割程度までの Web ページで効果が期待できることがわかった．
キーワード　Web パフォーマンス，Web ブラウザ，レンダリングオフロード

Web Performance Acceleration by Caching Rendering Results

Yuusuke NAKANO[†,††], Noriaki KAMIYAMA[†,††], Kohei SHIOMOTO[††], Go HASEGAWA[†††],

Masayuki MURATA[†], and Hideo MIYAHARA[†]

†† NTT Network Technology Laboratories, NTT Corporation　Midori–cho 3–9–11, Musashino–shi, Tokyo, 180–8585 Japan
††† Cybermedia Center, Osaka University　1–32, Machikaneyama, Toyonaka–shi, Osaka, 560-0043 Japan
† Department of Information Science, Osaka University　1–5, Yamadaoka, Suita–shi, Osaka, 565-0871 Japan
E-mail: †{nakano.yuusuke,kamiyama.noriaki,murata,miyahara}@ist.osaka-u.ac.jp,
††shiomoto.kohei@lab.ntt.co.jp, †††hasegawa@cmc.osaka-u.ac.jp

Abstract　Web performance is getting important in recent years. Web performance means the time from clicking a link on a web page to finishing showing the web page of the link and low web performance web pages are tend to lose customers. In our research, we measured download time for downloading files in popular web pages by running web browsers on 4 hosts all over the world using PlanetLab and found the longest portion of the download time. As a result of the measurement, we found the longest portion of the download is Blocked time in browsers. In this paper, we propose a method for accelerating web performance by reducing such blocked time with cache of rendering results. The proposed method uses the in-network rendering function which renders web pages instead of web browsers. The in-network rendering function also stores the rendering result in a cache and reuses it for the other web browsers for reducing the blocked time. To evaluate the proposed method, we calculated the web performance of web pages whose rendering results are cached by analyzing the measured download time. As a result, we found that the proposed method accelerate web performance of long RTT web pages whose dynamic objects ratio within 80%.
Key words　Web performance, Web browser, Rendering offload

1. はじめに

近年，Web パフォーマンスの重要性が注目を集めている．Web パフォーマンスとは，Web ページの表示にかかる時間である．ユーザは Web パフォーマンスが低い Web ページから離れる傾向にあり，Web パフォーマンスの低下はサービス提供者の収入の低下に直結する．例えば 2000ms の遅延は Bing におけるユーザあたりの収入を 4.3%下げることがわかっている．

Web パフォーマンスの低下の原因は 3 つに分類される．1 つは Web サーバの性能に起因するものである．Web サーバによるリクエストの処理時間は Web パフォーマンスに直結する．また，Web サーバの計算リソースの量がリクエストの処理にかかる量を下回れば，Web サーバ側のキューの溢れにより，リクエストの破棄による再送が発生し，パフォーマンスは大きく低下する．2 つ目はネットワークの性能に起因するものである．Web ブラウザから Web サーバまでの RTT が大きい場合，スループットが大きい場合と比較して，Web パフォーマンスは大きく低下する傾向にある [1]．また，経路上で輻輳が発生した場合も，パケットの破棄，再送が発生することで，Web パフォーマンスは低下する．3 つ目は Web ブラウザや Web ページの作りに起因するものである．Web ブラウザは，Web ページを構成するスクリプトや画像などのオブジェクトをダウンロードし，それらを画面上に表示できる形に整形，ビットマップ化を行っており，このような処理は一般的にレンダリングと呼ばれる．また，Web ブラウザは，オブジェクトを並列にダウンロードすることで，Web パフォーマンスの向上を図っている．しかし，Web ページの作りによっては，効率的な並列ダウンロードができない場合があり，Web パフォーマンスに大きく影響する [2]．

本稿では，現在公開されている Web ページを構成するオブジェクトのダウンロード時間を測定することで，上記 3 つの Web パフォーマンス低下の原因のうち，最も支配的な原因が Web ブラウザ内での待ち時間であることを明らかにした．なお，文献 [2] においても同様の知見が報告されている．

そこで，本稿では，これまで端末内のブラウザで行われてきたレンダリングの処理を肩代わりする機能をネットワーク内に配置し，本機能によって生成されるレンダリング結果をキャッシュしておくことで，キャッシュヒット時にブラウザ内での待ち時間を削減し，レンダリングにかかる時間を削減する手法を提案する．

2. Web パフォーマンス低下の主な原因の調査

先に述べたように，Web パフォーマンス低下の原因は Web サーバ，ネットワーク，クライアントそれぞれに含まれる．これらのうち，最も支配的な原因を特定するため，以下のような調査を実施した．

（1） 実際の Web ページの表示の際に送受信されリクエストレスポンスから，各オブジェクトのダウンロードにかかる時間の測定

（2） 収集された各オブジェクトのダウンロード時間から，Web サーバ，ネットワーク，クライアントそれぞれにかかる時間を抽出し，最も時間のかかっている処理を特定

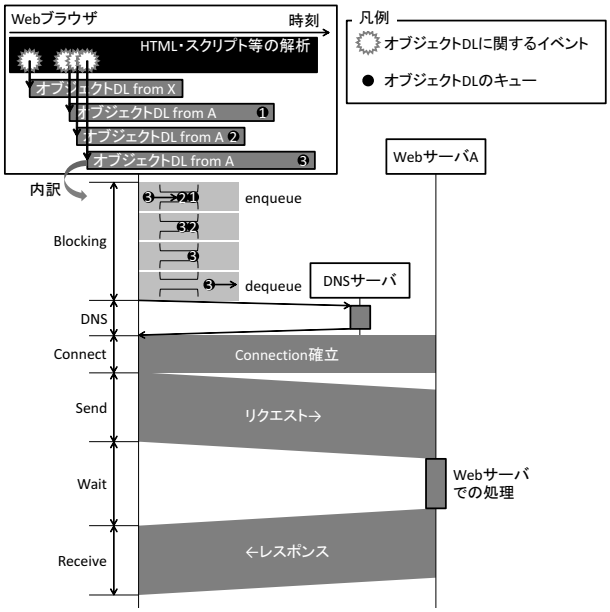

図 1　オブジェクトのダウンロードにかかる時間の内訳

2.1 Web ページを構成するオブジェクトのダウンロード時間の測定

2.1.1 測定結果の形式

Web ページを構成するオブジェクトのダウンロードにかかる時間は，Firefox や Chrome といった近年のブラウザによって測定することができる．また，測定結果は HAR(HTTP Archive) [3] と呼ばれる，JSON ファイルとして出力できる．HAR には Web ページを構成する各オブジェクトがダウンロードの開始時刻と，ダウンロードにかかる時間の内訳が含まれる．

2.1.2 オブジェクトのダウンロード開始時刻

図 1 は Web ブラウザの動作と，ダウンロード開始時刻，ダウンロード時間の内訳の関係を示したものである．ブラウザは Web ページを構成する HTML や各種スクリプトを解析すると同時に，解析結果にしたがって，Web ページの表示に必要なスクリプトや画像などのオブジェクトをダウンロードするためのイベントを発行する．このイベントを契機としてそれぞれのオブジェクトのダウンロードが開始され，ダウンロード開始時刻として HAR に記録される．図 1 の例では，サーバ X に対するオブジェクトのダウンロードのイベントが発生し，その後，立て続けにサーバ A に対する 3 つのダウンロードのイベントが発生する．なお，オブジェクト間には依存関係があり，その依存関係に従った順番でオブエジェクトはダウンロードが開始される．

2.1.3 オブジェクトのダウンロード時間の内訳

単一の Web サーバから複数のダウンロードを行う際は，一定の並列数に制限される．このような最大コネクション数は，Web サーバの負荷を制限するためにブラウザに固定で設定されている値である．この例では，並列数は 1 とされており (Chrome では 6 に設定されている)，1 つのオブジェクトがダウンロードされている間，後のダウンロードはエンキューされ，デキュー

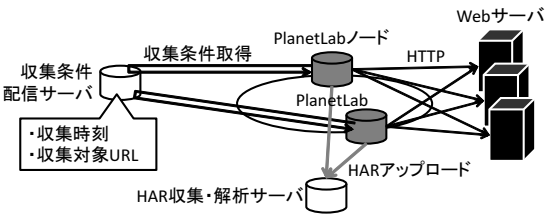

図 2　PlanetLab を用いた HAR の収集環境

されるまで待つ．これは HAR 内では Blocked として記録される．なお，Blocked の原因には様々なものがあり，文献 [4] で紹介されている．その後デキューされ，ダウンロードが開始される．まず，Web サーバの IP アドレスを取得するため，DNS サーバを用いて名前解決する．この時間は DNS として記録される．なお，DNS キャッシュされている場合は，この時間は 0 となる．Web サーバの IP アドレスが判明すると，ブラウザはサーバとコネクションを確立し，HTTP リクエストを送信し，HTTP レスポンス受信まで待ち，HTTP レスポンスを受信する．これらの時間はそれぞれ Connect，Send，Wait，Receive として記録される．

Blocked はクライアントに関する時間である．また，DNS，Connet，Send，Receive はネットワークに関する時間である．一方，Wait はサーバでの処理時間と，ネットワークでの RTT 両方が含まれる．それぞれの内訳のうち，多くの時間を要する内訳に関わる箇所が，Web パフォーマンス低下の主な原因であると考えられる．

2.1.4　HAR の収集環境

以上のように，Web パフォーマンス低下の主な要因を特定するため，ブラウザで実際の Web ページを表示した際の HAR を収集した．図 2 に HAR 収集環境を示す．Web ページを構成するオブジェクトのダウンロード時間は，Web ブラウザが置かれるネットワーク環境によって，大きく左右されると考えられる．そこで，インターネットに接続された世界中のホストを自由に利用できる実験環境である，PlanetLab [5] 上に HAR 収集環境を構築した．まず，PlanetLab に登録されているホストから，4 地点のホストに Web ブラウザと HAR 収集プログラムをインストールした．今回使用したホストのある 4 地点は，米国のカリフォルニア州，フランス，オーストラリア，アルゼンチンとした．HAR 収集プログラムは，収集条件配信サーバから，収集時刻や収集対象 URL などの収集条件を取得し，その条件にしたがって Web ブラウザを操作し，HAR を生成する．その後，HAR 収集プログラムは生成された HAR を HAR 収集・解析サーバにアップロードする．

一方，表示する Web ページによっても，ダウンロード時間は異なるため，多様な Web ページを収集対象とする必要がある．そこで，Web サイトの人気ランキングを提供している Alexa [6] が，Web サイトのカテゴリ (ニュースやショッピング等) ごとに分けて保持している人気ランキングを用い，各カテゴリから上位同数ずつ URL を抽出し，マージすることで 959 個の URL リストを生成した．このリストに従って，HAR ファイルを収集した．

2.2　最も時間のかかる内訳の特定

以上のようにして，各地点から収集対象の Web ページを表示した際の HAR を収集し，オブジェクトのダウンロード時間のうち，最も時間のかかる内訳を特定した．表 1 に 4 地点におけるオブジェクトダウンロード時間の内訳の平均 (ms) を示す．

表 1 を参照すると，Blocked と Wait の時間が突出して長く，特に Blocked が長いことがわかる．Blocked はブラウザ内でのオブジェクトのダウンロードが開始されるまでの待ち時間であり，先にも述べたが，Web サーバあたりの最大コネクション数の上限に達することで，このような待ち時間が発生する．

加えて，オブジェクト間の依存関係による待ち時間もある．文献 [4] に示されている通り，Web ページを構成するオブジェクト間には様々な依存関係があり，あるオブジェクトのダウンロードを開始するためには，その前にダウンロードが完了していなければならない別のオブジェクトが存在することが多く，多くの時間が割かれていると考えられる．

以上のように，Web パフォーマンス低下の主な要因は，ブラウザ内での待ち時間であることがわかった．このような待ち時間を削減することで，Web パフォーマンスの大幅な向上が図れると考えられる．

3.　提 案 手 法

Web サーバあたりのオブジェクト数や，オブジェクト間の依存関係が多いと，待ち時間も多くなると考えられる．このような待ち時間を削減するためには，ダウンロードするオブジェクト数の削減が有効である．そこで本稿では，レンダリング結果をキャッシュすることで，ダウンロードするオブジェクト数を削減し，ブラウザ内の待ち時間を削減することで Web パフォーマンスを向上させる手法を提案する．

提案手法はネットワーク内のエッジ付近に配置されたレンダリング・キャッシュ機能と，ユーザの端末にインストールされた専用ブラウザから構成される．なお，専用ブラウザは既存の Web ブラウザのエクステンションとして実装することを想定している．ユーザはブラウザを介して，レンダリング・キャッシュ機能に対して閲覧したい Web ページの URL を送信する．URL を受信したレンダリング・キャッシュ機能は，既存の Web ブラウザと同様にして該当する URL の Web ページをレンダリングする．その後，レンダリング結果をキャッシュすると同時に，ブラウザに返送する．ブラウザは返送されたレンダリン

表 1　地点ごとのオブジェクトダウンロード時間に含まれる内訳の平均 (ms)

内訳＼地点	California	Argentina	France	Australia
Blocked	255.36	552.92	418.45	377.77
DNS	0.68	0.30	0.43	0.85
Connect	13.86	25.01	81.32	21.18
Send	0.02	0.01	0.01	0.02
Wait	195.10	392.28	235.04	292.07
Receive	53.10	78.61	46.43	72.43

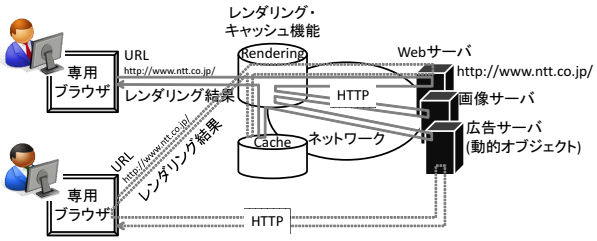

図 3　提案手法の概要

グ結果を画面に表示する．このように，レンダリング・キャッシュ機能が初めてレンダリングする Web ページについては，既存の Web ページと同様に Web ページをレンダリングするため，これまで発生していた待ち時間は依然存在する．一方，同じ URL の Web ページを別のユーザが表示する際は，既にレンダリング結果がキャッシュされているため，レンダリング・キャッシュ機能は URL を受信後すぐにレンダリング結果をブラウザに返送し，ブラウザは待ち時間なくレンダリング結果を画面に表示することができる．

しかし実際には，多くの Web ページは動的なオブジェクトを含むため，あるユーザのためにレンダリングされたキャッシュを，他のユーザがそのまま表示することはできず，先のように待ち時間なくレンダリング結果を画面に表示できるのは，静的なオブジェクトのみで構成される一部の Web ページのみである．例えば，表示するたびに内容の変わる広告や，ユーザごとに内容の変わる Twitter や Facebook などのタイムラインなどが，動的なオブジェクトであるといえ，多くの Web ページがこの様な動的なオブジェクトを含む．

この様な動的なオブジェクトの存在を考慮した提案手法の概要を図 3 に示す．レンダリング・キャッシュ機能は，先に述べたように，初めてレンダリングする Web ページについては既存の Web ブラウザと同様にレンダリングし，レンダリング結果をブラウザに返送する．一方，2 回目以降にレンダリングする際，レンダリング・キャッシュ機能の動作は次のように分岐する．

2 回目　新たにレンダリングし，その結果とキャッシュされているレンダリング結果との差分をとり，差分のない静的部分のみをキャッシュし，新たなレンダリング結果をブラウザに返送する．この時キャッシュするレンダリング結果は，Web ページの静的な部分に関する DOM ツリー・レンダーツリー・Javascript のブラウザプログラム内部の表現といったレンダリング時に生成される全てのデータをシリアライズしたもの，スクリプトや画像のような外部リソース，そして，HTML 内の動的な部分と，レンダーツリーの各ノードとを関連付けるための XSL である．

3 回目以降　レンダリング対象の URL の HTML を Web サーバから取得し，キャッシュされている静的なオブジェクトのレンダリング結果とともにブラウザに返送する．

なお，より正確に静的なオブジェクトを判定するために，2 回以上のレンダリング結果の差分をとり，共通部分をキャッシュしてもよい．

レンダリング結果を取得したブラウザは，DOM ツリー，レンダーツリー，Javascript などをメモリ上に展開すると同時に，レンダリング結果とともに取得した HTML 内の動的な部分を XSL を用いて抽出し，その部分のみをレンダリングする．その際，必要となるスクリプトや画像などのオブジェクトを Web サーバから取得する．このようにして，動的部分のオブジェクトのみをダウンロード，レンダリングすることで，ダウンロードするオブジェクト数を削減し，ブラウザ内の待ち時間を削減することができる．これにより，Web ページのレンダリングにかかる時間を削減し，Web パフォーマンスの向上を図る．

4. 評　　価

提案手法を実装する前に，有効性について評価した．

4.1 評 価 手 法

有効性の評価のために，レンダリング結果をキャッシュした場合のレンダリング時間を算出し，キャッシュを利用しない場合のレンダリング時間と比較する．以下の手順でレンダリング時間を算出した．

（1） 実際に公開されている Web ページの動的オブジェクトを特定する．

（2） 動的オブジェクトのみをレンダリングするためにかかる時間を，オブジェクトのダウンロード開始時刻と，ダウンロード時間とを用いて机上計算する．

このようにして，提案手法においてブラウザが行う動的オブジェクトのみのレンダリングとにかかる時間を算出することで，キャッシュされた静的な部分のみを再利用する場合のレンダリング時間を算出する．なお，ブラウザでのレンダリング結果の受信と，受信されたレンダリング結果の処理にかかる時間は 0 とした．

以降では，上記のそれぞれの手法について詳述する．

4.1.1 実際の Web ページに含まれる動的オブジェクトの特定手法

動的オブジェクトの特定には，Web ページを構成するオブジェクトのサイズと URL とを用いる．HAR には各オブジェクトのサイズに関する情報が含まれるため，複数回同一の Web ページに対する HAR を収集し，同一の URL のオブジェクトであっても，サイズが異なっていれば動的オブジェクトであると判断する．また，それぞれの HAR において URL の異なるオブジェクトについても動的オブジェクトであると考えられる．このようにして，同一の Web ページの HAR 間で差分を取ることで，動的オブジェクトと静的オブジェクトとを分類する．

4.1.2 測定結果を用いた動的オブジェクトのみのレンダリング時間の算出手法

Web ページは複数のオブジェクトで構成されており，それらのオブジェクト間には依存関係がある．この様な依存関係を考慮して動的オブジェクトのみのレンダリング時間を算出する必要がある．まず，ダウンロード開始時刻でオブジェクトをクラスタリングする．これは，同一クラスタのオブジェクトはダウンロードが同時に行われているため，依存関係はなく，クラスタ間での依存関係を考慮するためである．その後，静的オブ

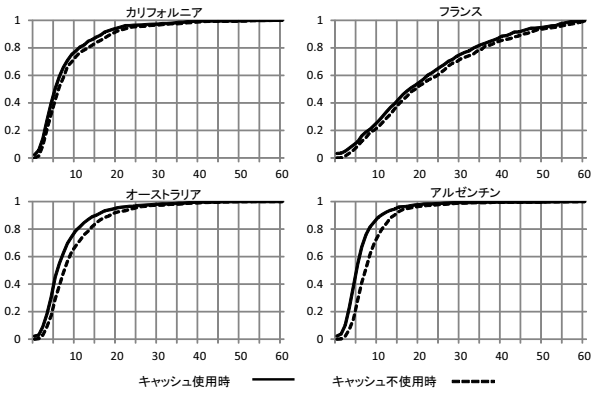

図 4　地点毎の onload 時間の累積分布のキャッシュ有無の比較 (縦軸：各 onload 時間の累積確率 横軸：onload 時間 (秒))

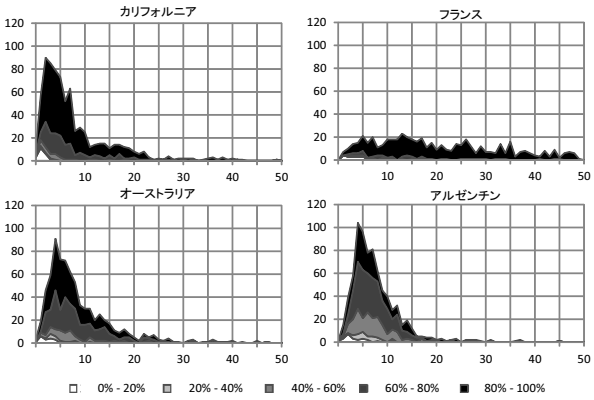

図 5　地点毎の onload 時間のヒストグラム (縦軸：各 onload 時間の Web ページの数，横軸：キャッシュしない場合の onload 時間 (秒)，各エリア：キャッシュした場合の onload 時間/キャッシュしない場合の onload 時間 * 100(%) を 20%毎に色分け)

ジェクトのダウンロード時間を 0 とした場合の，各クラスタのダウンロード時間の平均を算出する．これを各クラスタのダウンロード時間とする．動的オブジェクトのみをダウンロードするため，各クラスタのダウンロード時間が短縮される．この短縮された時間分，以降のクラスタのダウンロード開始時刻を早める．これは，クラスタ間には依存関係があるため，あるクラスタのダウンロード時間の増減は，それ以降のクラスタのダウンロード開始時刻に影響すると考えられるためである．このようにして，クラスタのダウンロード時間と，ダウンロード開始時刻とを更新することで，レンダリングに必要なオブジェクト (onload 時間以内のオブジェクト) のダウンロード完了時間を算出し，動的オブジェクトのみのレンダリング時間とした．

4.2 評価結果

以上のような手法を用いて，提案手法の有効性を評価した．評価には 2.1.4 章で収集された HAR を用い，各地点の同一 Web ページの HAR 間で差分を取ることで動的オブジェクトと静的オブジェクトとを分類した．

図 4 に地点毎の onload 時間の累積分布を，レンダリング結果のキャッシュ使用時と不使用時について示す．オーストラリア，アルゼンチンにおいて他の地点と比較して，onload 時間が

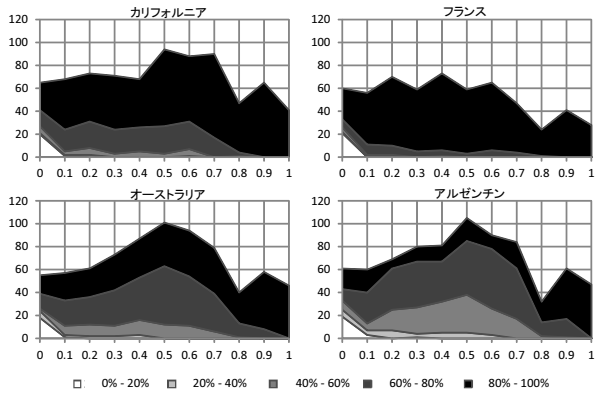

図 6　地点毎の動的オブジェクトの割合のヒストグラム (縦軸：各動的オブジェクトの割合の Web ページの数，横軸：動的オブジェクトの割合，各エリア：キャッシュした場合の onload 時間/キャッシュしない場合の onload 時間 * 100(%) を 20%毎に色分け)

削減されている．これは，オーストラリアとアルゼンチンとに関しては，図中のキャッシュ使用時の線 (実線) がキャッシュ不使用時の線 (点線) よりも大きく左に位置していることからわかる．ユーザが閲覧を諦める時間は 5 秒と言われており，例えばアルゼンチンにおいて，この 5 秒以内に onload が完了する Web ページの割合は，28%から 45%に増加している．

図 5 に地点毎の onload 時間の 1 秒間隔のヒストグラムを示す．なお，(キャッシュ使用時の onload 時間)/(キャッシュ不使用時の onload 時間) * 100(%) の値を 20%ごとに色分けしてヒストグラムを分割しており，キャッシュを使用しない場合の各 onload 時間の何割の onload 時間になっているのかを表している．図 5 を参照すると，オーストラリア，アルゼンチンにおいて，キャッシュを使用しない場合の多くの Web ページの onload 時間が，キャッシュを使用することで元の onload 時間の 80%以下まで短縮されていることがわかる．一方，オーストラリア，アルゼンチン以外の地域においては，キャッシュによる onload 時間の短縮は限られた Web ページに限定されている．

図 6 に地点毎の動的オブジェクトの割合の 0.1 間隔のヒストグラムを示す．なお，先と同様キャッシュを使用する場合の onload 時間が，元の onload 時間の何割になるかを色分けして示す．図 6 を参照すると，オーストラリア，アルゼンチンにおいて，動的オブジェクトが 8 割以下の Web ページのほとんどが元の onload 時間の 80%以下になっていることがわかる．一方，その他の地域においては，動的オブジェクトが 1 割以下の Web ページで onload 時間が短縮する Web ページが増加しているが，これは地域によらず，グラフ中に白く示されている，全体が静的である Web ページが一定数存在するためであると考えられる．

4.3 考　　察

評価の結果，提案手法はオーストラリア，アルゼンチンにおいて onload 時間を短縮することがわかった．これは，オーストラリア，アルゼンチンにおける Web サーバとの RTT が他の地域と比較して長い傾向にあることが原因として考えられる．これは，表 1 のオーストラリア，アルゼンチンの Wait が

他の地域と比較して長いことからわかる．このような地域では，個々のオブジェクトのダウンロード時間が長くなる傾向にあるため，ダウンロードが必要なオブジェクト数の削減は，onload 時間の削減に大きく影響すると考えられる．一方，Web サーバとの RTT が十分短い地域においては，提案手法による効果は少ないと言える．

以上のことから Web サーバとの RTT が長い Web ページを表示する際には，提案手法は有効であると言えるため，提案手法の実装時には，RTT の測定結果に応じたキャッシュ要否を動的に判断する仕組みを検討する．

また，アルゼンチン，オーストラリアでは動的オブジェクトの割合が 8 割以下の Web ページにおいて提案手法は有効であったが，その他の地域では 1 割以下の Web ページである程度の onload 時間の短縮が見られた．このことから，動的オブジェクトの割合によって，キャッシュ要否を動的に判断するような仕組みつについても今後検討する．

一方，図 5 から，キャッシュしない場合の onload 時間が短い Web ページのほうがキャッシュによる onload 時間の削減が大きくなる傾向が見られる．これは，onload 時間が短い Web ページを構成するオブジェクトに占める，静的なオブジェクトの割合が高いためであると考えられる．

5. 関 連 研 究

サーバやクラウド等でレンダリングすることで，レンダリング処理をオフロードし，モバイル環境での Web パフォーマンスの向上について研究されてきた．Kim らはシンクライアントの技術を用い，サーバ上の Web ブラウザを PDA 上で表示する pHTINC を提案している [7]．また，Shen らはプロキシにおいて，アニメーション等の動的な部分とそれ以外の静的な部分を分けてレンダリングし，静的な部分を分割された画像として，動的な部分とともにモバイル端末に送信することで，プロキシにおいてレンダリングした結果であっても，高速に動的な Web ページを表示することのできる Web ブラウザを提案している [8]．一方，Web サーバ内で DOM などのレンダリング結果を生成し，Web ブラウザに送信し，その後 Ajax のイベントをブラウザに送信する Web アプリケーションのフレームワークも提案されている [9]．以上のように，レンダリング処理のオフロードには様々な手法があり，サーバとクライアントとのレンダリング処理の役割分担が様々であった．Wang らはこの様なレンダリング処理の効率的な役割分担について議論している [10]．このように，レンダリング処理のオフロードに関しては，様々な研究がされており，近年では実際に多くのユーザに利用されている [11]．

以上のように，レンダリング処理のオフロードは様々な研究が行われてきた．しかし，レンダリング結果をキャッシュして，再利用するような提案はされていない．

6. ま と め

本稿では，これまで端末内のブラウザで行われてきたレンダリングの処理を肩代わりする機能をネットワーク内に配置し，本機能によって生成されるレンダリング結果をキャッシュしておくことで，キャッシュヒット時に，レンダリングにかかる時間を削減する Web パフォーマンス向上手法を提案した．

提案手法の有効性を評価するため，全世界で Web ページをブラウザで表示する際の時間を測定し，測定結果を用いてレンダリング結果がキャッシュされている場合の Web ページの表示にかかる時間を机上計算した．この結果，RTT が長い地域や Web ページにおいて，提案手法は有効であることがわかった．また，この様な地域や Web ページにおいては，動的なオブジェクトの割合が 8 割程度までの Web ページで効果が期待できる一方，そうでない地域や Web ページにおいては，動的なオブジェクトの割合が 1 割以下の Web ページである程度の効果が期待できることがわかった．

今後は，評価結果をもとに，RTT や動的なオブジェクトの割合などを考慮したキャッシュ要否の判断手法について検討する．また，今回は Web ページ間の差分を取ることで，静的なオブジェクトの判断を行ったが，実際には静的なオブジェクトではないことも考えられる．ユーザ間で共通の静的なオブジェクトの判断手法についても，HTTP ヘッダのキャッシュコントロールに関する情報を参照するなど，正確に判断するための手法を検討する．最後に，提案手法のプロトタイプを実装することで，Web パフォーマンスを実測し，提案手法を評価する．

文　献

[1] I. Grigorik, High Performance Browser Networking What every web developer should know about networking and web performance, O'Reilly Media, 2013.

[2] S. Souders, High Performance Web Sites Essential Knowledge for Front-End Engineers, O'Reilly Media, 2007.

[3] "Http archive (har) format". https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/HAR/Overview.html

[4] X.S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "Demystifying page load performance with wprof," Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), pp.473–485, USENIX, Lombard, IL, 2013.

[5] "Planetlab". https://www.planet-lab.org/

[6] "Alexa". http://www.alexa.com/

[7] J. Kim, R.A. Baratto, and J. Nieh, "pthinc: A thin-client architecture for mobile wireless web," Proceedings of the 15th International Conference on World Wide Web, pp.143–152, WWW '06, ACM, New York, NY, USA, 2006.

[8] H. Shen, Z. Pan, H. Sun, Y. Lu, and S. Li, "A proxy-based mobile web browser," Proceedings of the International Conference on Multimedia, pp.763–766, MM '10, ACM, New York, NY, USA, 2010.

[9] B. McDaniel and G. Back, "The cloudbrowser web application framework," Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity, pp.141–156, SPLASH '12, ACM, New York, NY, USA, 2012.

[10] X.S. Wang, H. Shen, and D. Wetherall, "Accelerating the mobile web with selective offloading," Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, pp.45–50, MCC '13, ACM, New York, NY, USA, 2013.

[11] "What is amazon silk?". http://docs.aws.amazon.com/silk/latest/developerguide/introduction.html