複雑化するWebトラフィック

Webは様々なサービスのフロントエンドとして広く利用されています。現在のWebは静的なWebページから、JavaScriptを用いた動的でインタラクティブなWebアプリケーションへと変化しており、Webトラフィックにもその影響が表れています。HTTPリクエストレベルでWebトラフィックの振る舞いを見ながら、JavaScript処理に起因するボトルネックやその複雑さについて解説し、終わりに今後のWebの展望についてまとめます。

3.1 Webを取り巻く状況

Webは、最初はシンプルなHTMLファイルで書かれた単純なWebページから始まりましたが、現在では、様々なサービスのフロントエンドとしてWebアプリケーションが利用されるようになってきました。そして、これまでデスクトップ上の専用アプリケーションで利用されてきたサービスも、次々にWebアプリケーションとして提供されるようになっています。ユーザは、ニュースやブログを見たり読んだりするだけでなく、検索やメール、ソーシャルネットワークサービスや地図など、インタラクティブなUI（ユーザインタフェース）を介してWebアプリケーションを利用するというように、Webアプリケーションの利用方法や利用範囲が増々広がってきています。

Webアプリケーションのサービスが多様化している一方で、それらを利用するユーザ側の環境も多様化してきています。この記事を読まれている方の中にも、会社ではノートパソコンやデスクトップパソコンを利用しながら、通勤時間や家ではスマートフォンやタブレットを利用される方もいらっしゃると思います。利用端末の変化に伴い、利用するネットワーク環境も、有線だけでなく3G、LTEなどのモバイル接続やWiFiの利用など、ユーザが利用するネットワーク環境も多様化してきています。

Webアプリケーションの利用範囲、ユーザの利用環境が多様化してきている中、近年Webトラフィック量が再び増加しています。増加の理由として、これまでHTTP上で扱われていなかったコンテンツが、HTTPを介して扱われるようになってきたことが挙げられます。例えば、これまで動画はRTSPなど別のプロトコルで提供されていたものが、FLASHで提供されるようになり、WebサーバからHTTPプロトコルで提供できるようになってきました。また、大容量コンテンツのダウンロードサービスがWebアプリケーションで提供されるようになってきたことも、Webトラフィック量が増加してきていることの要因の一つと報告されています。このように様々なWebアプリケーションが利用され、全体としてWebトラフィック量が増加している中で、Webアプリケーションに対するユーザの要求が高まっています。

Webアプリケーションの性能を比べる指標はいくつかありますが、その中でも応答時間（ここでは表示が完了するまでの時間とします）は重要な指標の一つです。いくつかの論文やレポートにおいて、このWebアプリケーションの応答時間の短さが、非常に重要であると指摘されています。例えば、マイクロソフトのBingやGoogleの検索において、それらの検索結果の表示速度が数百ミリ秒遅延するだけで、歳入やその後の訪問数の減少につながるという報告があります。また、Webの表示が完了するまでの時間について、3秒ルールといったものが言われていますが、あるオンラインショッピングの統計では、実際には利用者は2秒以内の表示を期待していると報告されています。

高機能なUIや高速な応答時間など、ユーザの高い要求に応えるため、また多様なサービスを提供するために、Webアプリケーションは複雑になっています。その結果、最近のWebトラフィックも、当初のHTMLをHTTP GETしていたシンプルな頃と比べると非常に複雑になっています。では、実際に最近のWebアプリケーションは、その裏側でどのようなWebトラフィックが流れているのでしょうか。

3.2 最近のWebトラフィックの特徴

最近のWebトラフィックがどうなっているのかを調べるために、まず、モダンなWebアプリケーションの例として、Yahoo.comのトップページを閲覧したときのWebトラフィックを見てみます。図-1に、2013年11月5日のYahoo.comのトップページを表示しました。このトップページにアクセスしたとき、どのようなWebトラフィックが流れて

いたのか、簡単に数値で示してみます。

HTTPリクエスト数: 83
転送量: 978.7Kbyte
接続先のドメイン数: 13
TCPコネクション数: 39

この結果は、あくまでもある日のYahoo.comのトップページをPCのブラウザ(Firefox)から見たときの例なので、ユーザの利用環境やコンテンツによってこれらの数値は変わりますが、図-1と並べて見ることで、普段はあまり意識することのないWebトラフィックのボリュームを感じて頂けるかと思います。

では、このYahoo.comのトップページのWebトラフィックの数値は、一般的なWebアプリケーションと比較して多いのでしょうか、少ないのでしょうか。そこで、比較のためにHTTP Archive[*1]が提供している値と比べてみました。HTTP Archiveは、Alexa[*2]やフォーチュンに挙げられている主要なWebアプリケーションから2週間ごとにWebのコンテンツを取得し、自動的に分析してレポートしています。HTTP Archiveの平均的なWebアプリケーションでは、1ページ当り90以上のHTTPリクエストと、1,400Kbyteを超える転送量、15以上の異なるドメインに接続していました(2013年の年間平均)。このことから、平均的なWebアプリケーションの場合、Yahoo.comのトップページの例で見た以上のWebトラフィックが、1ページごとに流れていることになります。

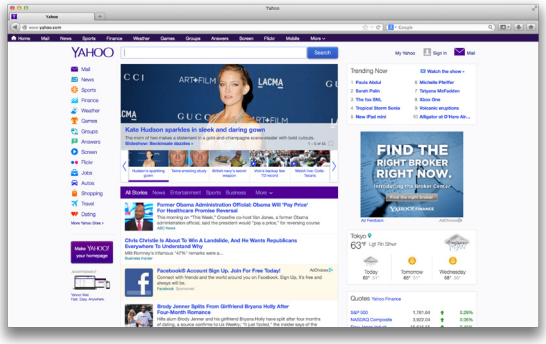

図-1 Yahoo.comのトップページ

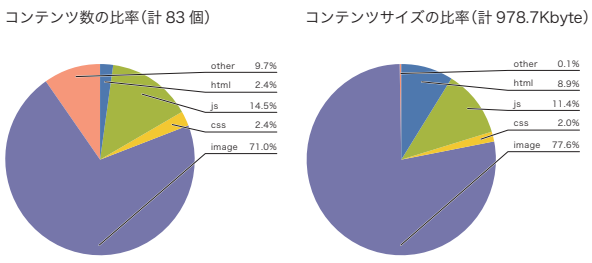

図-2 コンテンツタイプの比率

最近のWebトラフィックの量が把握できたので、次に、このWebトラフィックにはどのようなコンテンツが含まれているのか見ていきます。図-1のトップページに含まれていたコンテンツを、コンテンツタイプごとに分類し、コンテンツタイプの比率を図-2に示しました。左の円グラフはコンテンツ数、右の円グラフはコンテンツサイズで比較しています。コンテンツタイプの分類は、html、JavaScript(js)、Cascading Style Sheets(css)、JPEG・PNG・GIFなどを含む画像(image)と、それ以外(other)としました。図-2から、コンテンツ数とコンテンツサイズの両方で、画像が70%以上と一番大きな割合を占めており、2番目にJavaScriptが10%以上を占めていることが分かります。

では、このコンテンツタイプの比率も、一般的なWebアプリケーションと比べるとどうでしょうか。先程のHTTP

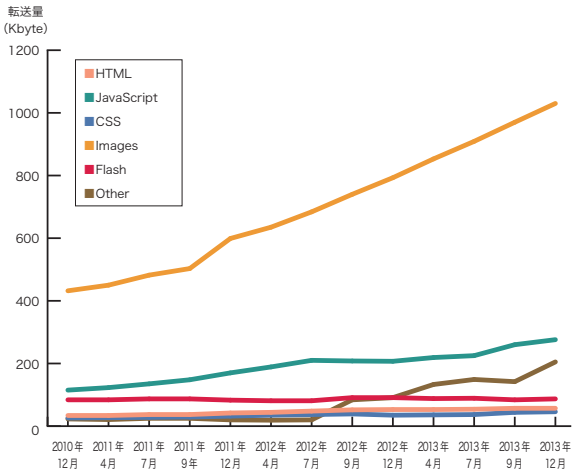

図-3 コンテンツタイプの転送量の推移

*1 "HTTP Archive"(http://httparchive.org/index.php)。
*2 "Alexa"(http://www.alexa.com)。

Archiveのデータを用いて、2010年12月から3年間、平均的なWebアプリケーションのコンテンツタイプの比率（転送量）を図-3に示しました。この3年間で画像とJavaScriptの転送量が２倍以上に増えてきていることが分かります。そして、図-2の結果と同様に、図-3でも画像の占める割合が一番大きく、2番目にJavaScriptが占めていることが分かります。

図-1を見てみると、サムネイル画像やアイコンなど、様々な箇所で画像が多用されていることが分かります。では、2番目に多いJavaScriptは、どのようなところで用いられているのでしょうか。図-1の例は、Yahoo.comのトップページということもあり、HTMLファイルを見てみると、YUI（Yahoo! User Interface Library）[*3]のJavaScriptライブラリが多く読み込まれています。これらのJavaScriptライブラリを用いることで、トップ中央のメイントピックにおける、マウスの位置に連動した画像表示の切り替えや、画像の自動切り替えなどの高機能なUIが実装されています。また、UIのような見た目には直接関係しませんが、広告のクリック数やユーザ動線のデータ取得などにもJavaScriptが利用されています。

このように近年WebアプリケーションにおいてJavaScriptの利用の比重は増していますが、JavaScriptが注目されるようになったのは2005年にGoogleマップのサービスが開始され、そのサービスと共にAjaxという言葉が広まった頃からです。

3.3 Ajaxによる応答速度の高速化とUIの高機能化

Ajax（Asynchronous JavaScript + XML）が登場する前のWebアプリケーションでは、ユーザの入力が確定するまで、Webサーバ側にブラウザからのリクエストは送信されず、レスポンスに必要なアプリケーションサーバやDBにおける処理は、Webサーバがブラウザからのリクエストを受信した後に開始されていました。そのため、ブラウザは、サーバ側での処理が完了し、Webサーバからレスポンスを受信するまで描画を開始できず、その描画もWebページ全体の再描画（画面遷移）であったため、Webアプリケーションの種類によっては、Webページ全体のデータ取得や描画処理に時間がかかりました。

Ajaxの登場により、ユーザによる操作が途中であっても、ブラウザとWebサーバ間で通信が行えるようになり、Webアプリケーションはマウスの移動やキーボードの入力など、ユーザの入力確定に先行して、操作途中の入力内容に基づいて、アプリケーションサーバやDBと連携したバックグラウンド処理をサーバ側で行うことができるようになりました。また、ブラウザ側ではサーバからのレスポンスを基に、Webページの部分的な再描画が行えるようになり、Webページのデータ取得にかかる時間や、描画処理の負荷が軽減可能になりました。

Ajaxを利用して、Webアプリケーションでリアルタイムの応答速度や、インタラクティブな操作感を実現するためには、ブラウザとWebサーバ間の非同期通信の仕組みの上に、Webアプリケーションごとの効果的なバックグラウンド処理が重要になってきます。そこで、Ajaxが用いられているサービスでは、実際にどのようなリクエストとレスポンスの送受信が行われているのか、検索のサジェスト機能と地図サービスについて見て行きます。

サジェスト機能は、途中まで入力された検索語に対して、その語に続く検索頻度の高い検索語を補完表示してくれます。図-4に、Yahoo検索で「IIJ」を検索したときの、サジェスト機能の結果を示しました。「I、I、J」と入力語が増えるたびに、それぞれ異なる補完結果が表示されていることが分

図-4 検索フィールド入力とサジェスト機能例

＊3　"YUI"（http://yuilibrary.com）。

かります。サジェスト機能はその性質上、高速に補完結果の表示が行われることが期待されています。どのようにして高速に補完結果を表示しているのか、その仕組みを知るために、サジェスト機能が利用されているときの、HTTPリクエストとレスポンスを見ていきます。図-5に示したように、検索語の入力フォームに「I」が入力されると、ブラウザは、図の上部に示したようなURLに「p=I」というクエリを含んだHTTPリクエストを、Webサーバに対して1つ送信します。このリクエストに対するレスポンスとして、Webサーバはサジェスト機能で表示する保管結果のデータを、JSONP形式のデータとして返します。ブラウザはこのデータを、事前にWebサーバから取得していたJavaScriptで処理することで、検索語に対する補完結果部分だけブラウザで再描画します。図-4の一文字目の例では、図-5のように小文字の「i」に続き、「イオン、ipad、一休、..」のようなデータがブラウザに返ってきました(2013年11月4日の結果)。このように、検索語のサジェスト機能では、必要なデータだけをやり取りするための最小限のHTTPリクエストとレスポンスと、受信したデータを用いた部分的なブラウザの再描画により、高速にサジェスト結果の表示を可能にしています。

Ajaxを利用した地図サービスでは、マウスの位置に追従して、地図の画像がグリグリと動きます。先程見てきたサジェスト機能と同じように、地図の場合もブラウザは表示の一部分を書き換えるだけで、ページ全体を表示し直すことはしません。そのために、地図サービスではマウスの動きを検知して、ブラウザは地図の足りない部分をWebサーバにリクエストします。ブラウザのリクエストに対して、Webサーバは足りない地図データをクライアントに送信し、クライアントは受け取ったデータを補正して、必要な部分を表示します。しかし、先程のサジェスト機能と違い、地図サービスではサーバ側からクライアント側に、複数の地図データが送られるため、サジェスト機能に用いられる小さなデータと比べると、サーバ側からクライアント側に送られるデータ量が大きくなります。そこで、データのダウンロードによりユーザが待っている時間を短くするために、あらかじめ次に表示されそうな場所を予測して、先行して地図データを取得するプリフェッチが重要な要素技術として用いられています。図-6は、Yahoo地図でフォームから神保町を検索したときの表示結果です。結果が表示された後、特にマウス操作などは行っていませんが、リクエストされた地図データを見ていくと、ブラウザ上に表示されているデータを取得し終わった後に、続けてブラウザ上に表示されていない部分のデータの取得が続いています。図-6の例では、表示されている地図の南に続く部分や、表示範囲を広げた場合など、図には表示しきれませんでしたが、他にもいくつかのデータが先行取得されていました。このように、地図の足りないところだけを部分的に取得して表示したり、ユーザが地図を見ている間に、ユーザの次の行動を予測して先行してデータを取得したりしておくことで、地図サービスの快適な操作感を可能にしています。

3.4 Webアプリケーションのボトルネックの複雑化

Ajaxが利用され始めた当初は、Ajaxを利用するためにブラウザ間の実装の違いに応じたJavaScriptのコードをWeb

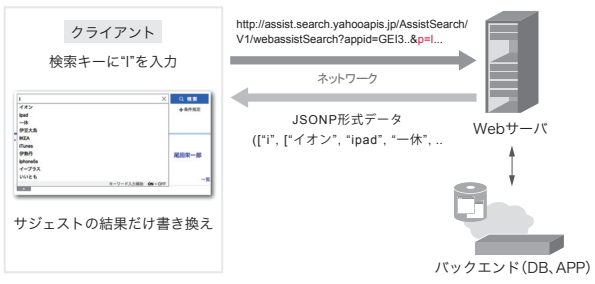

図-5 サジェスト機能のリクエスト・レスポンス

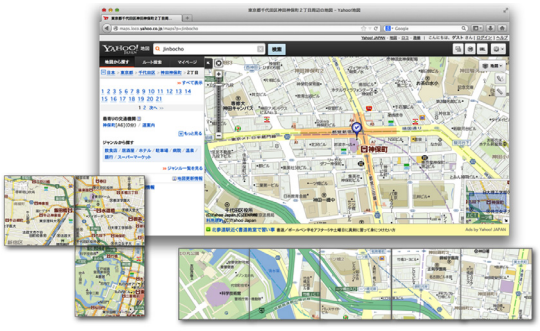

図-6 地図サービスにおけるプリフェッチの例

アプリケーション側で用意する必要がありましたが、最近では、アプリケーションフレームワークとしてjQuery[*4]、prototype.js[*5]やGoogle Web Toolkitが提供されており、Webアプリケーションにおいて比較的容易にAjaxを組み込むことができるようになってきました。Webアプリケーションで JavaScriptライブラリを利用することで、高速かつ快適なUIをユーザに提供できる反面、JavaScriptファイル間の依存関係や、描画処理における実行順序の依存関係により、Webアプリケーションのボトルネックを見つけるのが難しくなってきています。

ブラウザはその描画プロセスにおいて、HTMLとCSSからそれぞれオブジェクトモデル(DOMとCSSOM)という、ブラウザが直接解釈可能な中間データを生成します。JavaScriptを含まないWebページであれば、DOMとCSSOMからRender Treeが作られて、表示のレイアウトが決まり、実際に描画が行われます。ここで、HTMLとCSS以外に、JavaScriptが加わると、三者間の依存関係により、描画処理の流れが絡み合ってきます。何が起きるかというと、まず、JavaScriptはその実行のためにコンテンツのレイアウトを必要としているので、レイアウトを管理しているCSSからCSSOMができ上がるまで、JavaScriptはその実行を待ちます。更に、HTMLからDOMを構築するには、その前にJavaScriptの実行が完了している必要があるため、JavaScript実行が完了するまで、ブラウザはDOMの構築の実行を待つことになります。

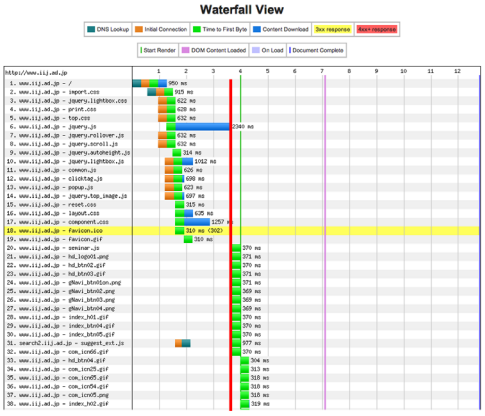

図-7 WEBPAGETESTによるウォーターフォール表示

では、このようなコンテンツ間の実行順序における依存関係が、実際のWebアプリケーションの実行時にどのように影響を及ぼしているのか、IIJのホームページを見たときのHTTPリクエストを例に見てみます。図-7にWEBPAGETEST[*6]で、IIJのホームページを指定したときのHTTPリクエストの様子をネットワークウォーターフォールで示しました。

WEBPAGETESTでは、URLと調査する地理的位置、ブラウザの種類、及びネットワーク帯域やRTTを指定して、Webアプリケーションのパフォーマンスを調べることができます。今回はコンテンツの依存関係による影響を分かりやすくするために、ネットワーク帯域にMobile 3G-Fast (1.6Mbps 150ms RTT)を指定してテストを実施しました。

ネットワークウォーターフォールは、HTTPリクエストが送信された順に、上から下にHTTPリクエストを並べて表示し、各HTTPリクエストにおけるDNS Lookupの有無やその時間、TCPコネクションの3-Way Handshakingの有無、そしてリクエストされたコンテンツのダウンロード時間などを合わせて表示できるため、各HTTPリクエストの振る舞いを調べるのに適しています。

さて、図-7で注目したいのは上から6番目のjquery.jsです。このコンテンツのダウンロードにかかった時間は、他のコンテンツに比べて圧倒的に長く約2秒です。jquery.jsは前述のアプリケーションフレームワークとして出てきたjQueryのコアファイルです。6番目のjquery.jsの取得に続いて、7番目以降にもjQuery関連のJavaScriptファイルの取得が続いています。それらのjQuery関連のJavaScriptファイルの中でも、jquery.jsが239Kbyteとファイルサイズが最も大きく、その結果ダウンロードに時間がかかっています。

更に、図-7で、このjquery.jsのダウンロード終了時刻に合わせて図上に垂直の赤線を引いてみると、20番目以降にリクエストされたコンテンツ群のダウンロード開始時刻に重なり

*4 "jQuery"(http://jquery.com)。
*5 "Prototype.js"(http://prototypejs.org)。
*6 "WEBPAGETEST"(http://www.webpagetest.org)。

ます。これを、先程のHTMLとCSS、JavaScriptの描画処理の順序に照らし合わせてみると、jquery.jsの取得が完了するまで、ブラウザは、jQueryを含むJavaScriptの実行を待ち、その間、HTMLのparseとDOMの構築及び、HTML中に埋め込まれている外部コンテンツの取得も停止しています。jquery.jsの取得完了、及びJavaScriptの実行完了の後、DOMの構築が進んだことにより、一気に20番目以降の画像などの外部コンテンツの取得が開始されたと考えられます。

ここでは、コンテンツ間の実行順序に依存関係がある場合を例として、その影響範囲が比較的分かりやすい例を見てきました。今回紹介したようなJavaScriptの実行により他のコンテンツの処理がブロックされるような例は、他の多くのWebアプリケーションでも散見され、Webアプリケーションの応答時間に大きな影響を与えていることが指摘されています。しかし、コンテンツの階層構造が多段にわたる場合や、複数コンテンツ間で入り組んだ依存関係を持つような場合、ネットワークウォーターフォールだけでは、ボトルネックとなるコンテンツや依存関係、そしてそのボトルネックの影響範囲を見分けるのは難しくなります。また、実行環境やネットワーク環境など、ユーザの利用環境が多様化してきていることから、複数の利用環境におけるボトルネックの影響度の違いも考慮する必要があるため、ボトルネックの判断は更に難しくなってきています。

3.5　今後のWebの集約化と簡素化の流れ

ここまでは、主にJavaScriptと関連したWebアプリケーションの挙動について見てきましたが、JavaScriptはWebアプリケーションの要素技術の一部に過ぎません。他にも、Webアプリケーションの機能性を向上させるために、新しい規格としてHTML5やCSS3が導入されたり、ターゲット広告のようなコンテンツのパーソナライズ化を行うためにCookieなどを用いたユーザ情報の取得が行われたり、外部WebAPIを利用したWebアプリケーション間の連携(マッシュアップ)が行われるなど様々な技術が利用されています。

また、Webアプリケーションの応答時間を高速化するために、CSSスプライトなど、一つにまとめられた複数の画像から特定の場所の画像だけを表示したり、JavaScriptの縮小化(ミニファイ化)として、JavaScriptファイルから不要箇所や空白改行を省くような、Webアプリケーションのコンテンツベースでの集約化技術も利用されています。このような既存の技術の組み合わせと応用によるWebアプリケーションの処理の集約化だけでなく、プロトコルベースでの集約化・高速化の流れとして、現在IETF[*7]においてHTTP2.0の策定作業が進んでいます。

このようにWebアプリケーションを取り巻く状況は、一部で集約化の流れがあり、それによりWebアプリケーションはある一面においては簡素化されることが期待されますが、その一方で集約化により、処理の流れが複雑化することも予想されます。

ISPとしてWebトラフィックの振る舞いやその特徴を理解することは、効率的な運用を図っていく上で非常に重要です。そのために、今後も引き続きWebアプリケーションを取り巻く技術や、Webアプリケーションの動きを理解していくことが重要になると考えています。また、ネットワーク側だけでなく、Webアプリケーションの構造やブラウザ内での処理など、多面的に可視化していくことが必要になると考えており、そのための技術開発を今後も行っていきたいと考えています。

執筆者:

二宮 恵 (にのみや めぐみ)
株式会社IIJ イノベーションインスティテュート 技術研究所 研究員。Webトラフィックの研究に従事。

＊7　"IETF"(http://www.ietf.org)。