УДК 004.455.1:004.738.12

Н. С. Мациевский

ПРОБЛЕМЫ КЛИЕНТСКОЙ ПРОИЗВОДИТЕЛЬНОСТИ ИНФОРМАЦИОННЫХ РЕСУРСОВ СЕТИ

Рассмотрены доступность информационных ресурсов Сети и проблемы, связанные с клиентской производительностью ресурсов. Предложен комплексный подход для устранения проблем доступности веб-ресурсов, связанных с клиентской производительностью.

Availability of information resources and client-side performance issues are reviewed. There is a generic approach to resolve website availability issues related to client side.

Ключевые слова: клиентская оптимизация, веб-ресурсы, проблемы производительности, информационные ресурсы.

Key words: website availability, client-side optimization, informational resources, web resources, web performance.

Введение

Информационный ресурс — организованная совокупность информационных объектов [1]. В Сети эта совокупность может быть реализована как в виде веб-сайта, отдельной веб-страницы, полноценного веб-портала и других формах. Одной из форм информационного ресурса является электронная библиотека — «упорядоченная коллекция разнородных электронных документов (в том числе книг), снабженных средствами навигации и поиска» [2, с. 12].

Информационные ресурсы в Сети — обычно набор веб-страниц. Каждая веб-страница представляет собой набор объекта-контейнера (ОК) и ряда внешних объектов (ВО). ОК обычно представлен в виде (X)HTML-файла, который содержит ссылки на ВО, являющиеся картинками, аудио- или видеофайлами, а также внешними CSS- и JavaScript-файлами. Большая часть нетекстовых ВО обычно изначально сжата, поэтому выигрыш от сжатия может быть получен только для самого (X)HTML-файа и некоторых CSS- и JavaScript-файлов. Для более 60 % веб-страниц ОК занимает менее 50 % общего объема всей страницы. В среднем ОК занимает 44 % от общего размера веб-страниц [7].

С 2003 по 2008 г. веб-страница более чем утроилась в размере, при этом число объектов почти удвоилось. Из-за этого для пользователей с низкой пропускной способностью канала (56 Кб и ISDN) постепенно время открытия сайта увеличивается. Для пользователей высокоскоростных каналов, напротив, среднее время открытия уменьшилось — по индексу интернет-производительности Keynote Business 40 (KB40) уменьшилось с 2,8 в 2006 г. до 2,33 с в 2008-м [7].

Несмотря на широкое распространение CSS, в 2007 г. было установлено, что из всех веб-страниц 62,6 % по-прежнему используют таблицы для разметки страницы, а 32,8 % — тег *font* для стилизации элементов. Средняя глубина вложенности таблиц уменьшилась при этом наполовину, по сравнению с 2006 г. — с 2,95 до 1,47. Сложные вложенные таблицы могут вызвать существенные задержки в браузерах при отображении страницы, потому что они должны быть проанализированы целиком перед своим отображением. Среднее число элементов на странице выросло более чем вдвое с 2006 по 2007 г. — с 281 до 592,6 [7].

Число запросов к страницам в пересчете на пользователя более чем удвоилось — с 7074 в 2000 г. до 14 670 в 2007-м. При этом средний размер страницы (учитывая то, что запросы к страницам отделялись от запросов к прокси-серверам по задержке ответа) удвоился с 2000 по 2007 г. [8].

По проведенному в 2007 г. исследованию [5] на 84,8 % веб-страниц был обнаружен элемент *script*. Средний размер внешних файлов скриптов при этом составил 8845 байтов в несжатом виде и 6302 байтов в сжатом. Общий размер скриптов на странице — соответственно 68 812 и 49 738 байтов. На странице в среднем использовалось 7 внешних файлов скриптов, 6 из которых были уникальными [5].

В 2007 г. 82,4 % страниц содержали тег *link*, а 54,5 % — тег *style* (при этом среднее число внутренних тегов *style* — 2,27). Средний размер внешнего файла стилей — 6575 байтов в несжатом виде и 4457 байтов в сжатом. Общий размер всех файлов стилей на среднестатистической странице при этом был 15 175 и 10 347 байтов соответственно [7].

Картинки использовались на 91,6 % всех веб-страниц в исследовании 2007 г. [7], GIF-формат при этом встречался на 84,6 % веб-страниц (77,9 % в 2006 г.), JPEG — на 64,5 % (в 2006 г. — 55,8 %), а PNG — на 32,2 %. При этом заметен существенный рост по сравнению с 2006 г. — 7,2 %. BMP, однако, обнаружено только на 0,8 % веб-страниц в 2006 г. Стоит заметить, что изображения в BMP-формате довольно велики по размеру, и данный формат не подразумевает их сжатие [7].

Средняя домашняя страница первых 100 блогов составляет 934 Кб. Две трети от этого занимают картинки (61,3 %), 17,2 % — приходится на скрипты, 15,3 % — на HTML, а 5,9 % — на CSS. На средней домашней странице этих блогов было 63 изображения (включая как HTML-, так и CSS-картинки) и 9 внешних скриптов [6].

Если в 2000 г. практически весь трафик обусловлен классическими HTML-форматами, текстовыми и графическими, то в уже 2007-м почти весь объем передаваемых данных — это видеофайлы, бинарные обновления и загрузки программного обеспечения. Благодаря этому сдвигу к большому объему видеоинформации и программного обеспечения средний размер ответа увеличился с 12 294 до 68 275 байтов — примерно в 5,5 раза. Эффективность (клиентского) кэширования в Интернете упала, поскольку число ответов с динамически создаваемыми данными увеличилось с 21,2 до 37,1 %. Эта статистика демонстрирует, как суть веб-сайтов изменилась от предоставления статической информации до обеспечения динамического взаимодействия с пользователем [8].

1. Проблемы низкой производительности

Исследование Акамай (Akamai) [5] показало, что наиболее приемлемым временем открытия страниц является 2 с. При этом повышенная доступность ресурсов позволяла удерживать пользователей ресурсов Сети дольше и обеспечивать их большей информацией.

Это же исследование установило, что 23 % пользователей электронных магазинов разочарованы качеством сервиса в связи с медленной работой веб-сайта магазина. Более быстрые веб-сайты больше удовлетворяют ожиданиям пользователей, имеют низкий показатель отказов и лучшее качество.

При большом времени ожидания количество пользователей, дожидающихся открытия веб-страниц, значительно сокращается [4, с. 12], приводя к снижению доступности ресурсов Сети и увеличивая порог необходимых технических требований для пользователей таких ресурсов. При этом существуют методы снижения порога доступности и приведения ресурса Сети в максимально быстрое состояние [3, с. 15].

Пользовательское раздражение сильно возрастает, если скорость загрузки страницы превышает 8—10 с безо всякого уведомления пользователя о процессе загрузки. Пользователи с широкополосным доступом еще менее терпимы к задержкам при загрузке веб-страниц по сравнению с пользователями с более узким каналом. В опросе, проведенном ЮпитерРесёч (JupiterResearch), было установлено, что 33 % пользователей скоростного соединения не хотят ждать более 4 с при загрузке страницы, при этом 43 % — не ждут более 6 с [9].

В исследовании, проведенном в 2004 г., Ф. На (F. Nah) установила, что терпимое время ожидания (ТВО) для неработающих ссылок (без обратной связи) — 5—8 с. С добавлением уведомления пользователя о процессе загрузки (обратной связи), например индикатора загрузки, ТВО увеличилось до 38 с. ТВО для повторных попыток зайти на неработающие ссылки составляло максимум 2—3 с (без обратной связи). Ф. На заключила, что ТВО для веб-пользователей достигает максимум около 2 с. Если учесть стремление пользователя посетить сайт повторно, то Дэнис Галета (D. Galletta) и другие авторы выявили, что кривая сглаживается при 4 с и более и уходит в нуль в районе 8 с и более [9].

2. Решение проблем производительности

Наиболее простой способ увеличения скорости загрузки страницы — уменьшение размера загружаемых объектов. В ряде ситуаций можно без потерь содержания изменить состав HTML-документа, файлов CSS и JavaScript и тем самым уменьшить суммарный размер загружаемых пользователями страниц.

На высоконагруженных страницах, какими являются, например, главные страницы поисковых систем Google, Yahoo, Яндекс, возникают ситуации, когда каждый лишний байт страницы критичен для быстродействия.

Минимизацией именуется процесс уменьшения объема кода за счет следующих операций:

— удаление избыточных пробелов, табуляций и переносов строк;
— удаление комментариев;
— удаление дублирующегося кода.

Минимизация применима к коду HTML, CSS и JS и в зависимости от размера и содержимого кода позволяет достичь результатов, близких к gzip-сжатию, уменьшать файлы до 30 % от исходного размера, а иногда и более. При использовании же еще и gzip-сжатия предварительная минимизация позволяет увеличить итоговую степень сжатия в среднем на 3—5 % [4, с. 18].

Помимо непосредственной загрузки каждого внешнего объекта веб-страницы браузеру необходимо совершить ряд дополнительных действий:

— определить IP-адрес сервера по его доменному имени;
— установить новое соединение с этим сервером;
— отработать возможные редиректы;
— отправить запрос на сервер;
— дождаться ответа от сервера.

Таким образом, из-за отсутствия этих временных издержек один внешний объект всегда загружается быстрее, чем несколько объектов того же суммарного размера, загружающихся последовательно, а поскольку многие браузеры загружают внешние файлы JS строго последовательно, количество этих объектов способно существенно повлиять на скорость загрузки веб-страницы.

Браузеры и прокси-серверы обычно стремятся сохранить максимум информации в своих хранилищах, для того чтобы ускорить повторную загрузку ранее загруженных объектов. Важно помнить, что при этом возможна потеря актуальности представляемых данных, поэтому политика кэширования должна быть организована с учетом всех ситуаций.

Кэширование — это один из наиболее мощных механизмов для уменьшения объема передаваемых по сети данных, притом внедряется этот механизм очень просто [4, с. 32].

Наибольший эффект от уменьшения количества запросов к серверу ощутят пользователи с низкой пропускной способностью канала и большим временем отклика от сервера — обычно это пользователи мобильных устройств и коммутируемых соединений.

Даже когда веб-страница и все внешние объекты загружены на компьютер пользователя, браузеру по-прежнему требуется время для того, чтобы разобрать страницу, интерпретировать код HTML и CSS, выполнить код JavaScript. Принимая во внимание особенности работы браузеров на этом этапе, можно достичь существенно более высокой скорости загрузки страницы.

Разбирая полученный HTML-код, браузер строит дерево документа, содержащее все элементы страницы. Затем, отыскав все взаимосвязи между элементами этого дерева и CSS-селекторами, относящимися к данной странице, он применяет к документу стили.

Если на веб-странице присутствует большое количество элементов или объем CSS-кода достаточно велик, страница может прорисовываться с ощутимой задержкой. Когда объем кода уменьшить уже невозмож-

но, более высокой скорости загрузки можно достичь за счет эффективной верстки. Основные рекомендации к верстке следующие [4, с. 34].

1. Наиболее важное содержимое страницы должно находиться в самом начале HTML-документа. Так пользователи смогут начать взаимодействовать с этим содержимым раньше.

2. Актуальные размеры изображений и ячеек таблиц, содержащих большое количество данных, должны быть явно заданы при помощи HTML-аттрибутов или CSS-свойств. Это позволит избавиться от лишних перерисовок веб-страницы. Например, когда браузер загрузит изображение и определит его размер, ему не потребуется обновлять макет веб-страницы, для изображения уже будет зарезервировано необходимое пространство. Кроме того, точно заданные размеры изображения избавят браузер от избыточной операции масштабирования изображения на лету.

3. Следует отказаться от использования CSS-expressions для браузеров Internet Explorer. Они отрицательно влияют на производительность браузера и в большинстве ситуаций могут быть заменены более производительным JS-кодом, а иногда и вовсе альтернативной версткой. В ситуациях же, когда для требуемой функциональности сайта использования expressions не избежать, следует применять одноразовые expressions.

4. Нужно использовать быстродействующие селекторы идентификаторов и классов. Поскольку большинство браузеров анализируют селекторы справа налево, с виду простой селектор #header. menu li a будет применяться дольше, чем аналогичный ему селектор #header. menu-item. В первом случае браузеру требуется найти все ссылки на странице, проверить, находятся ли они в контексте элемента списка, элемента с классом menu и, наконец, элемента с идентификатором header. Второй вариант более предпочтителен, поскольку поиск элементов по классу и идентификатору выполнится существенно быстрее.

5. Универсальные, дочерние, соседние селекторы, селекторы атрибутов, псевдоклассов и псевдоэлементов должны применяться только в тех ситуациях, когда это действительно необходимо. Все эти разновидности CSS-селекторов существенно более ресурсоемки, чем селекторы идентификаторов или классов.

От структуры HTML-документа во многом зависит скорость загрузки страницы пользователем. Даже при одинаковом суммарном размере страниц и равном количестве внешних объектов две различные страницы могут загружаться за совершенно разное время. Причина в том, что отображение элементов частично загруженной страницы в большинстве браузеров осуществляется только после выполнения следующих шагов [4, с. 35]:

1) получения HTML-документа;

2) получения всех внешних объектов CSS, вызываемых в HTML-документе;

3) получения всех внешних объектов JavaScript, вызываемых в HTML-документе внутри тега <head>;

4) получения всех внешних объектов JavaScript в HTML-документе внутри тега <body>, расположенных в потоке выше выводящегося элемента.

Список литературы

1. *Антопольский А. Б.* Сегодня абсолютно чистых электронных библиотек, которые никак не нарушают законодательство, просто нет // Интернет-портал интеллектуальной молодежи. URL: http://ipim.ru/persons/14.html.
2. *Земсков А. И., Шрайберг Я. Л.* Электронные библиотеки. М., 2003.
3. *Мациевский Н. С.* Разгони свой сайт. М., 2009.
4. *Мациевский Н. С., Степанищев Е. В., Кондратенко Г. И.* Реактивные веб-сайты. М., 2010.
5. *Akamai* reveals 2 seconds as the new threshold of acceptability for eCommerce web page response times. Business Wire. URL: http://www.businesswire. com/news/google/20090914005141/en.
6. *Average* top 100 weblog performance survey. Website Optimization. URL: http://www.websiteoptimization.com/speed/tweak/average-top-100-weblog.
7. *Average* web page size triples since 2003. Website Optimization. URL: http://www.websiteoptimization. com/speed/tweak/average-web-page.
8. *Evolution* of the web from 2000 to 2007. Website Optimization. URL: http://www.websiteoptimization.com/speed/tweak/evolution-web.
9. *The psychology* of web performance. Website Optimization. URL: http:// www. websiteoptimization. com/speed/tweak/psychology-web-performance.

Об авторе

Николай Сергеевич Мациевский — асп., Московский государственный университет культуры и искусств, e-mail: speed@webo.name.

Author

Nikolay Matsievsky — PhD student, Moscow State University of Culture and Arts, e-mail: speed@webo.name.