Вирішення, запропоновані авторами у статті, можна застосовувати у створенні реальних проектів систем електронної контент-комерції та у подальших дослідженнях у галузі створення систем суспільного доступу до інформаційних ресурсів.

1. Hamilton J. A Conversation with Pat Selinger / James Hamilton // ACM Queue, Vol. 3, No. 3 – April 2005. 2. Howard Ph. IBM touts information as as service [Електронний ресурс] / Philip Howard, Bloor Research.- [Режим доступу]: http://www.regdeveloper.co.uk/2005/10/07/ibm_information_mana gement .– 2005. 3. Seltzer M. Beyond Relational Databases: There is more to data access than SQL./Margo Seltzer// ACM Queue, Vol. 3, No. 3 – April 2005. – P. 128–132. 4. The Lowell Database Research Self-Assessment Meeting [Електронний ресурс]/ Lowell Massachusetts. – 4–6 May 2003.- [Режим доступу]: http://research.microsoft. com/~gray/lowell. – June, 2003. 5. Берко А.Ю. Системи електронної контент-комерції / А.Ю. Берко, В.А. Висоцька, В.В. Пасічник. – Львів: Видавництво Нац. ун-ту "Львівська політехніка", 2009. – 612 с. 6. Єрмошенко М.М. Інформація в системі виробничих відносин / М.М. Єрмошенко // Актуальні проблеми економіки №10 (76), 2007. – С. 59–65. 7. Висоцька В.А. Методи та засоби опрацювання інформаційних ресурсів систем електронної контент-комерції / В.А. Висоцька, Л.В. Чирун // Вісник Нац. ун-ту "Львівська політехніка". Інформаційні системи та мережі. – 2009. – С. 39–47. 8. Рейнольдс М. Электронная коммерция. Основы программирования / Мэтью Рейнольдс. – М.: Лори, 2001. – 538 с.

УДК 004.78

І.В. Брунець
Національний університет "Львівська політехніка",
кафедра інформаційних системи та мереж

МЕТОДИ ОПТИМІЗАЦІЇ КЛІЄНТСЬКОЇ СТОРОНИ КОЛАБОРАТИВНИХ СЕРЕДОВИЩ

Наведено результати дослідження різних аспектів веб-розробки для виявлення найкращих способів збільшення продуктивності. Запропоновано способи вирішення проблем колаборативних середовищ, що стосуються клієнтської сторони.

Ключові слова: клієнтська оптимізація, колаборація, веб-конференція, вебінар.

The results of various aspects of web development to identify the best ways to increase productivity are in this article. Shown the solutions to the problems of collaborative environments related to client side.

Keywords: ClientSide optimization, collaboration, web-conference, webinar.

Вступ

З кожним роком кількість і обсяг сайтів стрімко зростає. Збільшується пропускна спроможність каналів, користувачі переходять з комутованого доступу на безлімітний. Сайти стають більшими за розміром, за наповненням і складнішими у взаємодії. Колись в літературі про веб-програмування можна було прочитати, що розмір сайта не повинен перевищувати 100 Kb, а нині це вже далеко не показник. Розміри файлів, що завантажуються, при цьому збільшуються, а час очікування користувачів не зменшується.

За останні п'ять років середній розмір веб-сторінок зріс утричі (за даними дослідження Akamai[1]), а за останній рік – у півтора раза (за даними webo.in[2]). Кожна сторінка використовує в середньому по 60 об'єктів, що вкрай негативно позначається на загальному часі завантаження. Тільки близько 5–10 % від загального часу завантаження припадає на серверну частину. Все інше становить саме клієнтська архітектура.

У швидкодії сайта дорога кожна мілісекунда. Недарма високонавантажені проекти типу Google, Amazon, Flickr, Netflix, Яндекс, вКонтакте і Одноклассники так серйозно ставляться до питання швидкості завантаження сайтів. За кожним втраченим моментом часу криється певна сума грошей. Для мультимедійних колаборативних середовищ питання оптимізації є ще актуальнішим.

Основний час під час завантаження сторінки йде саме на клієнтську частину. Серверні витрати зазвичай вкрай малі і становлять від 50 до 500 мс. Середньому користувачеві насправді абсолютно все одно, скільки сторінка буде створюватися на сервері, якщо він її побачить через півсекунди. У цьому разі фокус зміщується саме на клієнтську, а не серверну оптимізацію.

Характер проблем варіюється від сайта до сайта. Іноді він полягає в особливості інтернет-під'єднання основної маси користувачів ресурсу (наприклад, якщо широко використовуються модеми або GPRS). Іноді – в складності самого сайта і невиправданому використанні ресурсів мережі. Іноді – в безграмотному використанні клієнтських технологій та великої кількості різнорідних рішень. Але всі ці проблеми можна вирішити.

Аналіз останніх досліджень

Потреба в оптимізації завжди існувала, а в контексті інтернет-сайтів, що розвиваються все активніше, це просто необхідність.

Великий доробок у дослідженні клієнтської оптимізації Стіва Саудерса, організатора групи із швидкодії та функціональності (usability) в Yahoo. Запропоновано низку методів, що пришвидшують роботу сайта в рази [2, 5]:

- Зменшення кількості http запитів до сервера. Рекомендується об'єднувати css і js файли, що завантажуються на одній сторінці. 80 % часу завантаження витрачається на фронтенд. Найбільше часу йде на завантаження компонентів сторінки: картинок, таблиць стилів, скриптів, флеш. Зменшення кількості цих компонентів зменшує кількість запитів до сервера. Це ключове правило створення швидких сторінок. CSS-спрайти є найкращим методом скорочення кількості запитів на сервер. Об'єднуючи всі картинки веб-сторінки в одну велику картинку і використовуючи CSS-властивості фонового зображення і позиціонування, можна показати потрібну ділянку картинки.
- Використання CDN. На швидкість завантаження сторінки істотно впливає і те, наскільки далеко користувач перебуває від сервера. Розміщення контенту між декількома серверами, що рознесені географічно, дає змогу скоротити відстань від сервера до користувача. CDN (Content Delivery Network) – це безліч веб-серверів, рознесених географічно для досягнення максимальної швидкості віддачі контенту до клієнта. Сервер, який безпосередньо віддаватиме контент користувачеві, вибирають на підставі деяких показників. Наприклад, вибирають сервер з найменшою кількістю проміжних хопів до нього або з найменшим часом відгуку.
- Використання HTTP-заголовка Expires. Дизайн сторінок стає все складнішим, у контексті збільшення використання скриптів, CSS, картинок і флеша. Встановлюючи заданий заголовок, заданими сторінками можна керувати. Заголовок Expires найчастіше використовується з картинками, проте його слід застосовувати з усіма компонентами сторінки, враховуючи скрипти, флеш, CSS.
- Компресія HTML-файлів. Рекомендується застосувати для них техніку minify, також розмір файлів може бути істотно (до 80 %) зменшено через архівування (gzip). Час, необхідний для пересилання через мережу HTTP-запиту і відповіді на нього, може бути зменшено рішеннями розробників фронтенда. Крім швидкості Інтернету, є інші фактори, що впливають на час завантаження сторінки. Стиснення файлів допомагає зменшити час завантаження HTTP-відповіді, зменшуючи її обсяг.
- Розміщення CSS на початку сторінки. Розміщення CSS у кінці сторінки не дає змоги почати поступовий рендеринг багатьом браузерам, серед яких Internet Explorer. Браузер не починає генерувати сторінку, щоб не перемальовувати елементи, у яких під час завантаження зміниться стиль. Firefox починає відразу генерувати сторінку, в процесі завантаження,

можливо, генеруючи деякі елементи, але це є причиною появи нестилізованого контенту (FOUC – Flash Of Unstyled Content).

- Розміщення скриптів у кінці сторінки. Скрипти (зовнішні JS-файли) створюють схожу проблему, але вирішується вона з точністю до навпаки: скрипти слід переносити в самий низ сторінки, якомога ближче до кінця. Тоді ми дозволяємо браузеру генерувати сторінку поступово і одночасно розпаралелювати завантаження.
- Уникати CSS-виразів (expressions). Проблема з цими виразами в тому, що вони обчислюються набагато частіше, ніж потрібно. Вони обчислюються не тільки під час генерування сторінки та зміни розмірів вікна, але також при скролінгу і навіть коли користувач просто водить мишкою над сторінкою. Це нескладно відстежити, досить додати лічильник у вираз. Звичайний рух мишкою над сторінкою запросто може викликати обчислення виразу більше ніж 10000 разів. Використання цих виразів вже не є актуальним, оскільки підримку Internet Explorer припинили навіть YouTube і Google у березні поточного року [4].
- Винесення JavaScript і CSS в окремі файли. Використання зовнішніх файлів на практиці забезпечує не тільки зручність у написанні коду, але й виграш у швидкодії. Це пояснюється тим, що браузери кешують файли скриптів і CSS. Код JavaScript і CSS, що вбудовується в HTML, завантажується кожен раз, коли завантажується сам HTML-документ. Це зменшує кількість необхідних HTTP-запитів, але збільшує обсяг HTML. З іншого боку, якщо скрипти і таблиці стилів містяться в окремих файлах, що вже закешовані браузером, розмір HTML зменшується, не збільшуючи при цьому кількість HTTP-запитів. Багато сайтів тільки наполовину задовольняють ці вимоги. Для таких випадків загалом найкращим рішенням буде створення зовнішніх файлів для скриптів і таблиць стилів. Єдиний виняток, коли використання inline-коду дає більшу перевагу, – це використання його на домашніх сторінках, таких як головна сторінка Yahoo! (Http://www.yahoo.com/) і My Yahoo! (Http:// my.yahoo.com /). Для сторінок, які завантажуються лише кілька (зазвичай один) раз за весь сеанс, вигідніше вбудовувати скрипти і таблиці стилів прямо в HTML-документ, щоб виграти у швидкості завантаження.
- Зменшення кількості DNS-запитів. Система DNS встановлює відповідність імен хостів з їх IP-адресами, точно так само, як телефонний довідник дає змогу дізнатися номер людини за його іменем. Коли ввести www.yahoo.com в адресному рядку браузера, перетворювач DNS (DNS-resolver), до якого звернувся браузер, повертає IP-адресу сайта. Зазвичай потрібно 20-120 мілісекунд, щоб виконати DNS-запит і отримати відповідь (в українських реаліях цей час більший). Браузер змушений чекати завершення DNS-запиту, оскільки до цього моменту він ще не може нічого завантажувати. Коли клієнтський кеш очищується (як системний, так і у браузера), кількість DNS-запитів зростає до кількості унікальних імен хостів на сторінці. А це вміщує власне адресу самої сторінки, картинок, скриптів, CSS, об'єктів Flash тощо. Зменшення кількості унікальних імен хостів зменшує кількість DNS-запитів. Зменшення кількості унікальних імен хостів потенційно зменшує кількість паралельних завантажень компонентів сторінки. Зменшення кількості DNS-запитів зменшує час завантаження сторінки, але зменшення кількості паралельних завантажень може збільшити цей час. Тому оптимально розподілити файли компонентів між 2–4 (але не більше) унікальними хостами. Це є компромісом між зменшенням кількості DNS-запитів і збереженням непоганої паралельності під час завантаження компонентів сторінки.
- Мінімізація скрипта – це видалення з коду всіх неістотних символів з метою зменшення обсягу файла скрипта та прискорення його завантаження. У мінімізованому коді видаляються всі коментарі, пробіли, переноси рядків, символи табуляції. У випадку з Javascript це зменшує час завантаження сторінки, оскільки розмір файла зменшується. Дві найпопулярніші утиліти для мінімізації javascript – JSMin і YUI Compressor.

CSS-продуктивність часто ігнорується під час розроблення клієнтських додатків для браузера. Дуже часто незнання ключових моментів може призвести до появи безлічі “вузьких” місць під час роботи веб-додатка, які не залежать безпосередньо від сервера або каналу. Вони всі розміщені на стороні браузера. Для уточнення істотних факторів і відносного ранжирування відомих правил з написання ефективного CSS-коду було взято за основу таку формулу [6]:

$$\begin{aligned}\text{Час\_відтворення} = \text{Розмір\_DOM} \times \text{Кількість\_CSS\_селекторів} \times \\ \text{Складність\_стильових\_правил} \times \text{Час\_відтворення\_одного\_правила} + \\ \text{Час\_створення\_документа}\end{aligned} \quad (1)$$

Одразу при погляді на формулу (1) стає очевидним, що нам потрібно брати усереднену складність правил по всій таблиці стилів, тобто підставляти у формулу суму складнощів всіх CSS-селекторів, розділену на їх кількість.

У формулі фігурує кількість елементів, на які впливає цей селектор (це, зокрема, пояснює, чому універсальний селектор, *, такий ресурсомісткий). Для уточнення цього моменту було виконано тест з однаковим DOM-деревом і різними CSS-правилами (одне застосовувалося до всього дерева, а інше – лише до десятої його частини).

Ще не варто забувати про наявність у браузерів власної таблиці стилів, яка застосовується до кожної сторінки, що виводиться на екран. Розмір цієї таблиці можна визначити досить просто: потрібно лише відкрити дві сторінки з різною (і досить великою) кількістю CSS-правил, але однаковим DOM-деревом і перевірити, наскільки сповільнилося завантаження. Знаючи відношення розміру двох таблиць стилів, можна обчислити невідомий розмір таблиці стилів самого браузера (для основних браузерів це приблизно 30–50 правил, для IE – близько 200).

І ще один момент, який виник у ході розслідування: відіграє роль розмір повного DOM-Дерева, не тільки кількість тегів, але і кількість текстових вузлів, хоча це ніяк і не впливає на основні висновки.

У результаті різних тестів модель вдалося уточнити і показати, що час створення документа залежить від кількості вузлів в дереві (що цілком очевидно). Це можна перевірити двома наборами тестів: на збільшення кількості CSS-селекторів без збільшення DOM-дерева і на збільшення DOM-дерева без збільшення кількості CSS-селекторів. Добре видно, що час створення документа не є постійним і дещо збільшується при збільшенні розміру документа.

Також була перевірена гіпотеза, наскільки складові селектори відпрацьовують повільніше від своїх елементарних побратимів (мається на увазі різниця між. class1,. class1. class2 і. class1. class2. class3). Різниця була зафіксована, але виявилася неістотною (кожна ланка додає приблизно 10–20 % до загальної складності селектора).

Отже, після всіх уточнень формула набула вигляду (2):

$$T = (\sum DOM1 \times K + DOM2 \times \ln) \times t + DOM2 \times L, \quad (2)$$

де T – час відображення документа на екрані; *DOM1* – кількість елементів, на які може вплинути це CSS-правило (розбір CSS-правил у браузерах йде справа наліво); *DOM2* – розмір всього DOM-дерева; K – складність кожного окремого CSS-правила в таблиці стилів, от 1 до 1,5; *ln* – кількість вбудованих CSS-правил у браузері, близько 40–200; t – характерний час обробки одного правила для одного вузла дерева, приблизно 0,0001...0,0005 мс; L – характерні затримкм на створення одного елемента DOM-дерева, приблизно 0,0005...0,005 мс.

Ця модель дала змогу апроксимувати час відображення сторінки з точністю 10 % (у окремих випадках 20 %, мабуть, є ще багато неврахованих чинників, наприклад, особливості виділення пам'яті). Тестування проводилося на документах від 5000 DOM-вузлів і від 0 CSS-правил.

Аналіз цієї моделі дає змогу зробити величезну кількість вельми цікавих **висновків**:

- Розмір DOM-дерева відіграє основну роль, найголовнішу. Тому порада: зменшуй DOM усіма можливими способами. Зменшення його (як добре видно з підсумкової формули) на 20 % приведе до пропорційного прискорення відображення сторінки.

- Варто також врахувати, що у формулі фігурує не тільки загальний розмір дерева, але і кількість елементів, які обробляються при застосуванні CSS-селектора. Саме з цієї причини неефективно використовувати універсальний (*) селектор та теги: вони охоплюють істотну кількість елементів.

Альтернативою використання міток можна назвати два виходи:

1. Використовувати унікальні теги для унікальних елементів на сторінці (наприклад, для заокруглених куточків використовувати рідкісні теги – ins, del, q, u, b, i).

2. Використовувати унікальні класи для кожного набору стильових правил.

Якщо перший підхід може бути застосовний для невеликих сайтів (наприклад, для зменшення розміру HTML-коду), то у разі середніх і великих проектів однозначно варто використовувати другий підхід (що рекомендує і Віталій Харіс у своїх правилах для ефективного CSS і фреймворка Monkey Joe).

- Використання складних правил (селектор з декількома ланками) може бути виправдане (це не спричиняє значних витрат), однак якщо використовувати скрізь унікальні класи, то спадкування зазвичай відбувається саме по собі.
- Як глобальне скидання стильових правил (підхід “ластик”) можна рекомендувати скидати правила тільки у тих елементів, які відображаються (наприклад, якщо на сторінці 90 % DOM-дерева – це div, для яких не потрібні ніяких правила за замовчуванням, то перехід від глобального “ластика” до локального або взагалі його усунення за рахунок індивідуальних правил здатне дещо збільшити продуктивність).
- Оптимізувати кількість CSS-правил варто, якщо їх більше ніж 100–200 (бо інакше через правила самого браузера всі ваші зусилля будуть марними).

Також варто зазначити, що за результатами тестування Віталія Харісова невикористовувані CSS-правила додають деяку затримку у відображенні сторінки (до 10 % від часу відтворення), тому їх теж варто уникати.

Для середньої HTML-сторінки час її відображення (розмір DOM-дерева – 1000 елементів, CSS-правил – близько 500, кожне з них в середньому застосовується до 40 % елементів) становитиме близько 100 мс. Простою оптимізацією можна зменшити цей показник удвічі (наприклад, звузивши область впливу самих селекторів, якщо DOM-дерево зменшити не вдається).

Формулювання цілей статті

У попередніх працях виявлено низку проблем, що унеможливлюють ефективне використання колаборативних середовищ. Серед таких потрібно відзначити [4]:

- слабко розвинений мобільний інтерфейс. Багато колаборативних мереж, зокрема в бізнес-сфері, не враховують можливості доступу через смартфони, КПК, блекбері тощо. Крім того, чимало рішень потребують оплати за користування їхньою платформою для взаємодії. Не передбачено можливості оплати через мобільний телефон;
- незручний інтерфейс взаємодії – у мережі з високим ступенем інтерактивності та використанням відеозв’язку ця проблема набуває великого значення. Зокрема, використання вікон, що спливають, які у 93 % користувачів заблоковані. Отже, відбуватиметься втрата учасників, а накладання вікон буде незручним при будь-яких розмірах екрана;
- проблема масштабованості – участь великої кількості учасників нині є проблемною. Сучасним рішенням бракує вбудованої масштабованості, через що доводиться позапланово збільшувати ресурси з розширенням організації або навіть у разі тимчасових пікових навантажень. Така відсутність масштабованості є дуже дорогою, ризикованою. Крім того, вона може призводити до зменшення продуктивності, якщо через збільшення інтенсивності обміну повідомленнями неефективна робота рішення спричинить перебої в їхній передачі. Наприклад, більшість онлайн-семінарів дозволяють участь не більше ніж 20 учасників, у Skype – 9. Якщо потрібно збільшити кількість учасників, недостатньо змінити деякі значення полів у базі даних. Теоретично можна модифікувати БД так, щоб помістити будь-яке значення, проте немає ніякої гарантії стабільної роботи системи;

- недостатній рівень безпеки і захищеності даних – відсутні фільтри на дані, що завантажуються, недостатній захист корпоративних серверів, відсутність антивірусного захисту. Системи безпеки повинні передбачати всі можливості захисту, що надаються окремими спеціалізованими компонентами безпеки – антивірусні програми, міжмережеві екрани і пристрої запобігання вторгненням (IPS). Можливості вузькоспеціалізованих рішень, таких як міжмережеві екрани і антивірусне програмне забезпечення, недостатні для протидії сучасним атакам змішаного типу. Окремі компоненти системи захисту не можуть глибоко аналізувати пакети даних, асемблювати контент і не можуть забезпечити контролю контенту. А в сучасних умовах ведення бізнесу дуже важливо вирішувати такі завдання, не порушуючи цілісності інформаційної системи і не перериваючи роботи в мережі. В результаті, тільки комплексні рішення можуть забезпечити динамічний захист корпоративної мережі від великої кількості різноманітних загроз і вторгнень. Програмно-апаратні комплекси забезпечують високий захист мережі та контенту та гарантують роботу корпоративної мережі в режимі реального часу. Необхідно, щоб ці системи являли собою сплановані, масштабовані і легкокеровані рішення, що пропонують найкращі можливості, серед яких фільтрація вмісту, міжмережеві екрани, захист від вторгнень, побудова і підтримка VPN та формування трафіку;
- нестабільність роботи – гальмування, зависання додатка на стороні клієнта, що зумовлюється багатьма факторами. З боку клієнта – це швидкість Інтернету, що є достатньою для проведення аудіо- і відеотрансляції. З боку сервера – повинен бути ефективний механізм кешування даних, що дає змогу значно прискорити доступ до даних і скоротити витрати трафіку.

Очевидно, що актуальним є розроблення такого інструменту, що не тільки задовольняє вимоги повнофункціонального вебінару, але й буде зручним, ефективним і доступним. Виправлення частини з цих недоліків можливе за **оптимізації клієнтської частини** і використання сучасних технологій відображення контенту.

Говорячи про швидкість завантаження, не можна не відзначити її ролі в оцінці технологічної якості будь-якого інтернет-проекту. Варто звернути увагу і на такі моменти (які можна доволі швидко перевірити за допомогою безкоштовних інструментів):

- крос-браузерність;
- відповідність стандартам;
- семантика HTML-коду;
- доступність сайта для користувачів;
- швидкість роботи на стороні сервера;
- швидкість роботи на стороні браузера.

Метою статті є оптимізація колаборативних середовищ, для чого необхідно розв’язати такі задачі:

- досягнення мінімально можливого часу завантаження сторінки;
- досягнення мінімально можливого часу завантаження групи сторінок, що переглядаються в довільному порядку;
- забезпечення мінімально можливого часу з моменту запиту сторінки до моменту появи у користувача можливості переглядати сторінку і взаємодіяти з нею.

Це далеко не повний перелік можливих цілей. Іноді й зовсім потрібно досягати компромісу і вибирати між кількома взаємовиключними варіантами оптимізації. У таких ситуаціях краще мати максимум можливої інформації про ваші веб-сайти і їхніх відвідувачів.

Виклад основного матеріалу

Виявлено багато проблем колаборативних середовищ, зокрема:

- слабко розвинений мобільний інтерфейс. Багато колаборативних мереж, зокрема в бізнес-сфері, не враховують можливості доступу через смартфони, КПК тощо. Крім того, чимало рішень потребують оплати за користування їхньою платформою для взаємодії. Не передбачена можливість оплати через мобільний телефон;

- незручний інтерфейс взаємодії – у мережі з високим ступенем інтерактивності і використанням відеозв'язку ця проблема набуває великого значення. Зокрема, використання вікон, що випливають, які у 93 % користувачів заблоковані. Отже, відбуватиметься втрата учасників, а накладання вікон буде незручним за будь-яких розмірів екрана;
- нестабільність роботи – гальмування, зависання додатка на стороні клієнта, що зумовлюється багатьма факторами. З боку клієнта – це швидкість Інтернету, що є достатньою для аудіо- і відеотрансляції. З боку сервера повинен бути ефективний механізм кешування даних, що дає змогу значно прискорити доступ до даних і скоротити витрати трафіку.

Можна вирішити оптимізацією клієнтської частини (фронтенду) колаборативних середовищ.

Пропрацювавши п'ять років back-end програмістом, я досліджував різні аспекти веб-розробки для виявлення найкращих способів збільшення продуктивності. Для дослідження колаборативного середовища об'єктом дослідження вибрано відому платформу проведення вебінарів від Google. Засобами дослідження були сервіси webo.in [2]. В результаті дослідження виявлено, яка із частин потребує першочергової оптимізації і які заходи будуть найефективнішими.

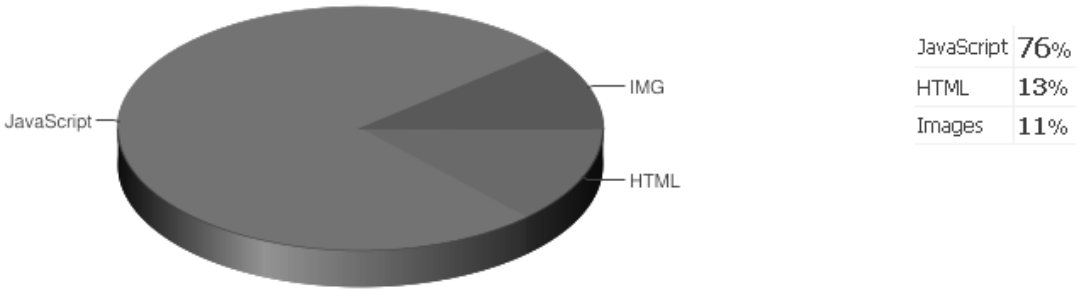

Рис. 1. Розмір різних типів файлів у Google OpenMeetings

На рис. 1 показано, яка із частин є найбільш ресурсомістка. Ця діаграма дає змогу зробити висновок, від якої оптимізації ми отримуємо найбільший ефект. Оскільки флеш завантажується через функцію JavaScript (для коректного відображення "небезпечного контенту"), то частина JavaScript містить в собі і JS і Flash.

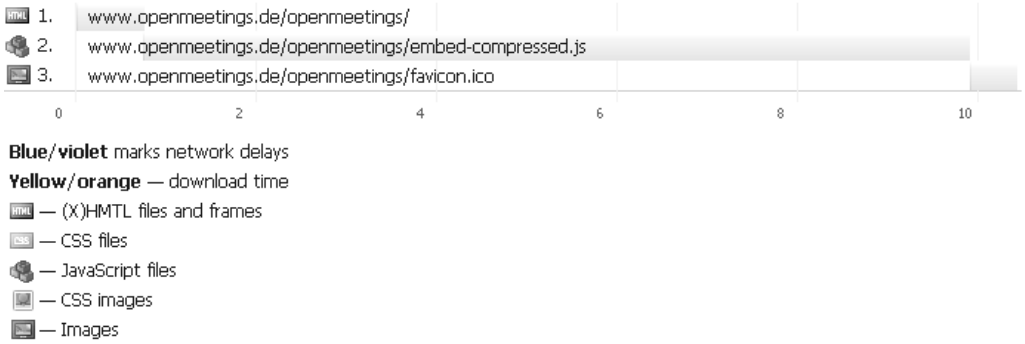

Рис. 2. Часова діаграма завантаження елементів сторінки Google OpenMeetings

На рис. 2 перший ряд, позначений "html", – це початковий запит HTML-документа. У такому разі на його отримання було витрачено лише 5 % всього часу завантаження сторінки. Це стосується не тільки веб-конференцій. Наприклад, всі, крім одного, сайти, що увійшли в десятку найпопулярніших сайтів у Штатах, витрачають менше ніж 20 % на завантаження HTML. Решта 80 % з гаком відсотків часу завантаження витрачаються на завантаження того, що зазначено в самому HTML – власне, фронтенда. Ось чому основним для створення швидких сайтів є покращення продуктивності фронтенда.

У контексті вебінарів оптимізація клієнтської частини відіграє доволі велику роль. Зокрема, доцільні:

- Компресія HTML-файлів. Рекомендується застосувати для них техніку minify, також розмір файлів може бути істотно (до 80 %) зменшено через архівування (gzip).

- Кешування для статичних файлів. Рекомендується віддавати всі ресурси (картинки, CSS-, JS-та мультимедіа-файли) з кешувальними заголовками (Cache-Control, зменшуючи тим самим кількість запитів з клієнта при наступних відвідуваннях чи використанні цих файлів для перегляду інших сторінок сайта).
- Компресія JS-файлів. Рекомендується скористатися інструментом для стиснення JS-файлів від Dean Edwards або YUI Compressor (докладніше про мінімізацію JS-коду). Після цього можна віддавати JS-файли у вигляді архівів із сервера (загальний виграш до 70 %).
- Оптимізація зображень. Використання CSS-спрайтів.
- Зменшення кількості http запитів до сервера. Рекомендується об'єднувати css і js файли, що завантажуються на одній сторінці.
- Використання сучасних мов гіпертекстової розмітки (HTML 5).

У HTML 5 для того, щоб під'єднати до сторінки відео- або аудіодоріжку, достатньо скористатись відповідними тегами.

Такі сайти, як Youtube, Viddler, Revver, Myspace і десятки інших, дають змогу будь-кому опублікувати своє відео або аудіо. Багато сайтів удаються до Flash, щоб забезпечити потрібну функціональність, оскільки HTML 4.1 не вистачає засобів для вбудовування і управління мультимедіа. І, хоча такі можливості надають різноманітні плагіни (Quicktime, Windows Media тощо), нині Flash – це єдиний поширений плагін, який надає крос-браузерне рішення з відповідним API для розробників.

Як доведено великою кількістю медіа-програвачів на основі Flash, автори зацікавлені в наданні інтерфейсу з їхнім власним дизайном, який, як правило, дає змогу користувачам вмикати, ставити на паузу, зупиняти, перемотувати і управляти гучністю. Стоїть завдання надати таку функціональність за допомогою додавання можливості вбудовувати відео- і аудіозасобами браузера, а також надати DOM API для доступу скриптів.

Елементи video і audio легко уможливлюють це. Більшість API – загальні між цими елементами, з відмінностями лише відносно visual і не-visual медіа. Всі сучасні браузери (окрім Internet Explorer) вже реалізували підтримку цих елементів. Найпростіший спосіб вбудувати відео – це використовувати тег video і дозволити браузеру відобразити інтерфейс за замовчуванням. Булевий атрибут controls визначає, чи включати за замовчуванням цей користувацький інтерфейс.

Необов'язковий атрибут poster може використовуватися для вказання зображення, яке буде відображатися до того, як відео почне програватися. Хоча є формати відео, що підтримують власний попередній перегляд (такі як Mpeg-4), цей спосіб – рішення, що дає змогу бути незалежним від відеоформату.

Також просто під'єднати і аудіо – використовуючи елемент audio. Хоча з очевидних причин у тегу audio немає атрибутів height, width і poster, між video і audio більшість атрибутів спільні.

У HTML 5 входить елемент source для вказання альтернативних відео- і аудіофайлів, щоб браузер міг вибрати той, який підходить до підтримуваного медіа-типу або кодеків. Атрибут media визначає вибір медіа-запиту, що оснований на обмеженнях пристроїв, а атрибут type – можливості медіа-типів і кодеків. Коли використовується атрибут source, слід опускати src в елементах video (audio), інакше source буде проігнорований.

Використання можливостей HTML 5 дасть змогу відмовитись від Flash технологій не тільки в контексті відеотрансляції. Малювання зображень у режимі реального часу, збереження даних у браузері, drag&drop – це все є в HTML 5.

Компанія Apple, що раніше не підтримувала Flash, вже заявила про підтримку нового стандарту. Використання веб-конференцій на продуктах компанії Apple, зокрема на iPhone, сприятиме розширенню аудиторії потенційних користувачів.

Висновки

Грамотне впровадження засобів колаборації забезпечує відчутні фінансові й адміністративні переваги великим розподіленим організаціям і організаціям, що використовують віддалену робочу силу.

У цій публікації проаналізовано найбільш ресурсомісткі місця в клієнтських частинах колаборативних середовищ. Тільки близько 5–10 % від загального часу завантаження припадає на серверну частину. Все інше становить саме клієнтська архітектура. Запропоновано заходи для оптимізації клієнтської частини (фронтенда) колаборативних середовищ.

Використання HTML 5 як альтернативи Flash не тільки позначиться на швидкодії, але й розширить спектр потенційних користувачів. Вирішення проблем гальмування, зависання, незручності сприятиме підвищенню продуктивності кожного працівника в середовищі, раціоналізації бізнес-процесів, і як наслідок, підвищенню загальної продуктивності компанії.

1. Web Application Accelerator // Akamai: The Leader in Web Application Acceleration and Performance Management, Streaming Media Services and Content Delivery. – [Електронний ресурс]. – http://www.akamai.com/html/solutions/web_application_accelerator.html. 2. Online website load time test. Website speedup. Website load time optimization// WEB Optimizator. – [Електронний ресурс]. – http://www.webo.name. 3. Брунець І. Особливості мультимедійних колаборативних середовищ з напівжорсткою організацією // Комп'ютерні науки та інженерія: матеріали 3-ї Міжнародної конференції молодих науковців CSE-2009. – Львів: Видавництво Нац. ун-ту "Львівська політехніка", 2009. – С. 42–44. 4. Брунець І. Основні показники вибору колаборативного мультимедійного середовища // Комп'ютерні науки та інженерія: матеріали IV Міжнародної науково-технічної конференції CSIT-2009. – Львів: Видавництво Нац. ун-ту "Львівська політехніка", 2009. – С. 263–266. 5. Best Practices for Speeding Up Your Web Site// Yahoo! Developer Network Home [Електронний ресурс] – http://developer.yahoo.com/performance/rules.html. 6. Мациевский Н. Реактивные веб-сайты. Клиентская оптимизация в алгоритмах и примерах / Мациевский Н., Степанищев Є., Кондратенко Г. – М.: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2010. – 336 с. 7. HTML5: (Wikipedia, the free encyclopedia) [Електронний ресурс] – http://en.wikipedia.org/wiki/HTML5.

УДК 004.89

Є.В. Буров, А.І. Городецька
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

ІНТЕЛЕКТУАЛЬНИЙ ТУРИСТИЧНИЙ СЕРВІС З ОПРАЦЮВАННЯМ КОНТЕКСТУ СИТУАЦІЇ

Запропоновано підхід до побудови контекстно-залежних сервісів інтелектуальної мережі, оснований на використанні концептуальних моделей, які описують ознаки певних ситуацій. Як приклад реалізації цього підходу розроблено макет сервісу електронного туризму.

Ключові слова: модель, контекст, інтелектуальний сервіс.

Paper proposes an approach for creating intellectual context-aware and situation-aware services in intellectual networks. This approach is based on conceptual models, describing possible situations and processed by modeling environment. An example of intellectual service providing information services for tourists is developed.

Keywords: model, context, intellectual service.

Постановка проблеми у загальному вигляді

Сьогодні туризм стає однією з найдинамічніших галузей світової економіки, що демонструють високі темпи зростання. Світова організація туризму (WTO – World tourist organization) прогнозує, що до 2020 року річна кількість туристичних поїздок збільшиться на 200 % – до 1.6 млрд. [1].