УДК 004.415

О.А. КРАВЧУК
Хмельницький національний університет

АСПЕКТИ ПЕРЕХОДУ НА HTTP2

В статті розглядається HTTP / 2 – протокол, заснований на протоколі SPDY від Google. Це найновіший стандарт, який вже підтримується Chrome, Firefox, Opera, Safari та іншими браузерами. Основні завдання HTTP / 2 – дозволити клієнтам серверів обирати протокол, реалізувати сумісність з HTTP / 1.1, знизити затримку завантаження сторінок, а також підтримати існуючі способи використання HTTP.

Ключові слова: протокол HTTP /2, новітні технології, шардінг, мультиплексування веб-сервера, відкритий код, розробка сайтів.

O.A. KRAVCHUK
Khmelnytsky National University

ASPECTS OF THE TRANSITION ON THE HTTP 2

Annotation In the article we review HTTP / 2 - protocol based on SPDY protocol from Google. This is the most recent standard is already supported Chrome, Firefox, Opera, Safari and other browsers. The main tasks of HTTP / 2 - enable client / server protocol to select, implement compatibility with HTTP / 1.1, lower latency pages as well as support existing uses of HTTP. When planning the transition to the HTTP / 2? No single answer to this question is not and can not be. Let us, however, a hint: watch regularly logs traffic to your service. When you see that most of the visitors are using supports HTTP / 2 browsers - can proceed. To date, support for HTTP / 2 is implemented in Chrome (including the mobile versions for Android), Firefox, Opera, Edge, Safari. When planning the transition should take into account the specifics of your project. If you have many people who come to you with a mobile, it means that you preferably go to the HTTP / 2 as soon as possible. Smartphones and tablets advantages of the new protocol will be especially apparent. However, one needs to consider many nuances: for example, in many parts of the world are still many users browser Opera Mini, but it is HTTP / 2 is not yet supported. If you are planning to launch a new Web service - think about the prospect of switching to HTTP / 2. Of course, you still have to use HTTP / 1.1 for some time, but now you can take steps to optimize that will facilitate your life in the future.

Keywords: protocol HTTP / 2, the latest technology, fast sharding, multiplexing, Web server, open source, web development

Постановка проблеми. У минулому році в світі мережевих технологій відбулася дуже важлива подія: була затверджена і стандартизована нова версія протоколу HTTP – HTTP / 2. Вона вже підтримується на популярних веб-серверах: Apache і Nginx, йде робота по впровадженню HTTP / 2 в IIS та реалізована підтримка більшості сучасних браузерів.

В останній час використання HTTP/2 істотно розширилося. За статистичними даними на середину 2015 року відсоток сайтів і веб-сервісів, які перейшли на HTTP / 2, був невеликий – всього 0,4%. А в січні 2016 року відбулося значне зростання: з 0,4 % до 6,5%. Є всі підстави вважати, що найближчим часом ці темпи будуть збільшуватися. Задуматися про практичні аспекти переходу на HTTP / 2 варто вже сьогодні.

Аналіз досліджень. Протокол HTTP 1.1 став незамінним для використання в Інтернеті. Величезні інвестиції були вкладені в протоколи та інфраструктуру, які тепер отримують з цього прибуток. Дійшло до того, що сьогодні часто простіше запустити що-небудь поверх HTTP, ніж створювати щось нове замість нього.

Коли HTTP був створений і випущений в світ, він, ймовірно, сприймався скоріше як простий та прямолінійний протокол, але час показав, що це не так. HTTP 1.0 в RFC 1945 – це 60 сторінок специфікації, випущеної в 1996 році. RFC 2616, який описував HTTP 1.1, був випущений лише трьома роками пізніше в 1999 році та значно розрісся до 176 сторінок. Крім того, коли в IETF працювали над оновленням специфікації, вона була розбита на шість документів з ще більшою кількістю сторінок в підсумку. Без сумнівів, HTTP 1.1 великий та включає безліч деталей, тонкощів, опціональних розділів.

Природа HTTP 1.1 полягає в наявності великої кількості дрібних деталей та опцій, доступних для подальшої зміни, що виростила систему програм, де немає жодної реалізації, яка б втілила все. Це привело до ситуації, коли можливості, що спочатку мало використовувалися, з'являлися лише в невеликому числі реалізацій, і ті хто їх реалізовував після спостерігали незначне їх використання.

Пізніше це викликало проблеми в сумісності, коли клієнти та сервери почали активніше використовувати подібні можливості. Конвеєрна обробка HTTP (HTTP pipelining) – це один з показових прикладів подібних можливостей.

HTTP 1.1 пройшов важкий шлях, щоб по-справжньому скористатися всією потужністю та продуктивністю, яку дає TCP. HTTP-клієнти і браузери повинні бути винахідливими, щоб знайти способи для зменшення часу завантаження сторінки.

Інші експерименти, які паралельно проводилися протягом багатьох років, також підтверджували, що TCP не так просто замінити і тому фахівці продовжують працювати над поліпшенням як TCP, так і протоколів, що працюють поверх нього.

TCP можна легко почати використовувати повноцінно, щоб уникнути пауз та періодів часу, які могли бути використані для відправки або прийому більшої кількості даних.

Коли дивишся на тенденції розвитку деяких найбільш популярних на сьогодні сайтів і порівнюєш скільки часу займає завантаження їх головної сторінки, вони стають очевидними. За останні роки кількість даних, які потрібно поступово передати, зросла до позначки 1,5Мб і вище, але, що найбільш важливо для нас в цьому контексті, так це число об'єктів, яке в середньому тепер близько до сотні. Сто об'єктів необхідно завантажити, щоб відобразити всю сторінку цілком.

HTTP 1.1 дуже чутливий до затримок частково через те, що в конвеєрної передачі HTTP й раніше вистачало проблем і вона відключена у переважної більшості користувачів.

У той час, як всі спостерігали значне збільшення пропускної смуги у користувачів за останні кілька років, ми не бачили подібного рівня зниження затримки. Канали з високою затримкою, як у багатьох сучасних мобільних технологіях, значно знижують відчуття швидкої веб-навігації, навіть якщо у вас є дійсно високошвидкісне підключення.

Конвеєрна передача HTTP – це спосіб відправки чергового запиту, вже чекаючи відповідь на попередній запит, що схоже на чергу до касира в супермаркеті. Ви не знаєте, що за люди перед вами: швидкі клієнти або докучливі персони, яким буде потрібно нескінчену кількість часу, щоб завершити обслуговування, блокування початку черги. Безумовно, ви можете ретельно вибирати чергу і в підсумку вибрати ту, яку вважаєте за правильну, а іноді можете створити свою власну чергу, але, врешті-решт, ви не зможете уникнути прийняття рішення і одного разу вибравши чергу, не зможете її змінити. Створення нової черги пов'язане з продуктивністю та втратою ресурсів, і не може маштабуватися за межі невеликого числа черг. Для вирішення цього завдання немає ідеального рішення.

Заключний нюанс, на нашу думку, який застосовується власниками сайтів для поліпшення завантаження в браузерах, часто називають «шардінгом». Це, в основному, означає розосередження вашого сервісу по максимально можливому числу різних хостів. На перший погляд це звучить божевільно, але на це є проста причина.

Спочатку HTTP дозволяв використовувати клієнту максимум два TCP з'єднання на кожен хост. Таким чином, щоб не порушувати специфікацію, просунуті сайти просто придумували нові імена хостів, що дало можливість отримати більше число з'єднань для вашого сайту та скоротити час завантаження сторінки. Згодом, це обмеження було прибрано зі специфікації. На сьогодні клієнти використовують 6-8 з'єднань на хост, але й досі мають обмеження, тому сайти використовують цю техніку збільшення числа з'єднань. По мірі збільшення числа об'єктів велика кількість з'єднань почала використовуватися просто для того, щоб переконатися, що HTTP справляється добре й робить сайт швидше.

Ще одна причина шардінга – це розміщення зображень і подібних ресурсів на окремих хостах, які не використовують cookie, оскільки вони на сьогоднішній день можуть бути значного розміру. Використовуючи хости зображень без cookie, ви можете збільшити продуктивність за рахунок значно менших HTTP-запитів.

Специфікація протоколу роз'яснює, як саме використовувати його для тексту і TLS. Однак Google (Chrome) і Mozilla (Firefox) оголосили про те, що будуть підтримувати HTTP / 2 тільки по TLS. Хоча Microsoft і не оголосила про це офіційно, але тестові версії Internet Explorer на Windows 10 показали, що компанія прийняла те ж саме рішення. Все це говорить про те, що новий протокол поступово стане застосовуватись тільки через TLS. Отже, HTTP / 2 буде підтримуватися тільки для URL-адрес типу HTTPS. Безумовно, це звучить як примус до використання HTTPS, проте в кінцевому рахунку це призведе тільки до вигоди, адже Інтернет стане безпечніше.

Цей стандарт має певні вимоги до TLS і буде використовувати саму безпечну реалізацію TLS. Специфікація вимагає TLS 1.2 і вище, забороняє стиснення та повторне погодження, а також має жорсткі вимоги до розміру ключів і набору шифрів.

Нижче представлені загальні рекомендації (можливості) TLS для HTTP / 2:

- застосування щодо безпечного використання TLS;
- підтримка Server Name Indication (SNI), що потрібно тільки для стандарту з TLS3 і вище;
- для HTTP / 2 через TLS2 відключення стиснення (воно не потрібно, оскільки протокол забезпечує можливості безпечного стиснення);
- для HTTP / 2 через TLS2 необхідно відключення повторного погодження;
- розмір ключа повинен бути як мінімум 2048 бітів для DHE і 224 біта для ECDHE;
- протокол не використовує набори шифрів з «чорного списку»;
- набір шифрів повинен бути тільки AEAD;
- мінімальний розмір ключів – 128 біт EC, 2048 біт RSA.

Формулювання цілей статті. Мета дослідження – огляд та аналіз протоколу HTTP / 2 – це нова версія стандартного протоколу передачі гіпертексту. Протокол HTTP (Hypertext Transfer Protocol) є основою взаємодії браузерів з веб-серверами, завдяки чому здійснюються завантаження і відображення веб-сторінок. Головною метою поновлення якого було підвищення швидкості завантаження сторінок.

Виклад основного матеріалу дослідження. Задуматися про практичні аспекти переходу на HTTP / 2 варто вже зараз. Цю тему ми хотіли б розглянути в сьогоднішній статті. Особливо нас буде цікавити проблема адаптації існуючих прийомів оптимізації продуктивності веб-сайтів під специфіку нового протоколу.

Перш ніж перейти безпосередньо до розгляду цього питання, звернемося до історії протоколу HTTP / 2 і коротко опишемо основні нововведення, що відрізняють його від HTTP / 1.1.

Від HTTP до HTTP / 2. Трохи історії. Перший опис протоколу HTTP (HyperText Transfer Protocol) було опубліковано в 1991 році. У 1999 році була розроблена і описана версія HTTP 1.1, яка використовується і до цього дня. У той далекий час (майже 20 років тому) веб-сайти були зовсім не такими, як зараз. За відносно невеликий період часу сайти стали «важити» набагато більше. Домашня сторінка середньостатистичного сучасного сайту містить приблизно 1,9 МБ даних: зображення, JS, CSS і багато іншого.

Через обмеження на кількість одночасних підключень в HTTP / 1.1 завантаження сторінок, що містять велику кількість «важкого» контенту, здійснюється повільно. Можна виділити два шляхи вирішення цієї проблеми. Перший полягає у використанні різних технік оптимізації продуктивності, а другий – в спробі модифікації самого протоколу HTTP з метою усунення можливих вузьких місць. Розглянемо такі спроби більш докладно.

Перший масштабний проект реформування HTTP був представлений в 2009 році інженерами

Google. Це протокол SPDY, метою якого в першу чергу було прискорення роботи веб-сайтів і додатків шляхом модифікації традиційних способів прийому і відправлення запитів.

SPDY вимагає підтримки як на стороні сервера, так і на стороні клієнта. Розробники Google створили спеціалізовані модулі для Apache (mod_spdy) і для Nginx (ngx_http_spdy_module). Підтримується він і практично у всіх популярних браузерах.

HTTP / 2, представлений шістьма роками пізніше, багато в чому грунтується на SPDY. Нова версія HTTP була створена робочою групою Hypertext Transfer Protocol working group. У травні 2015 року специфікація HTTP / 2 була опублікована як RFC 7540.

Протокол HTTP / 2 сумісний з HTTP / 1.1. Зміни, спрямовані на усунення вузьких місць і підвищення продуктивності, багато в чому продовжують лінію SPDY. Розглянемо коротко найбільш важливі з них.

Мультиплексування, можливо, це найголовніша перевага HTTP / 2. У HTTP / 1.1 для кожного запиту потрібно встановлювати окреме TCP-з'єднання. Мультиплексування ж дозволяє браузеру виконувати безліч запитів в рамках одного TCP-з'єднання: HTTP / 2. В сучасних браузерах кількість одночасних TCP-з'єднань обмежена. Тому сторінки з великою кількістю статичного контенту завантажуються не так швидко, як хотілося б. У HTTP / 2 завдяки мультиплексуванню статичні елементи завантажуються паралельно, і завдяки цьому істотно поліпшується продуктивність.

Ще одне нововведення HTTP / 2 – це пріоритезація. Кожному запиту можна призначити пріоритет. Існує два підходи до призначення пріоритетів: на основі ваги і на основі залежностей. У першому підході кожен потік отримує певну вагу. Потім на основі ваги сервер розподіляє навантаження між потоками. Такий підхід вже використовувався в протоколі SPDY.

Другий метод, який є основним в HTTP / 2, полягає в наступному: браузер просить сервер завантажувати певні елементи контенту в першу чергу. Наприклад, браузер може попросити сервер спочатку завантажити CSS-файли або JavaScript, а вже потім – HTML або зображення. У HTTP / 2 пріоритезація є не обов'язковим, а бажаним методом. Однак мультиплексування без неї працювати належним чином не буде. Якщо не налаштувати пріоритезацію може трапитись таке, що швидкість завантаження може бути навіть нижче, ніж HTTP / 1.1. Ресурси з більш низьким пріоритетом будуть займати смугу, що призведе до зниження продуктивності.

Наступним важливим нововведенням є стиснення HTTP-заголовків. Сучасна веб-сторінка складається з безлічі елементів: зображення, JS, CSS та інші. У запиті на завантаження кожного з цих елементів браузер передає HTTP-заголовок. Відправляючи запитні дані, сервер також додає до них заголовок. Все це пов'язане із зайвою витратою ресурсів. У HTTP / 2 заголовки передаються в стислому вигляді. Завдяки цьому зменшується кількість інформації, якою обмінюються між собою сервер і браузер. Замість алгоритмів gzip / deflate використовується HPACK. Це знижує уразливість до атак типу BREACH.

HTTP / 2 є на порядок безпечнішим для передачі даних ніж попередня версія протоколу. Одним з найважливіших вимог протоколу SPDY є обов'язкове шифрування (HTTPS) з'єднання між клієнтом і сервером. У HTTP / 2 воно обов'язкового характеру не має. Однак розробники браузерів прийняли рішення запровадити новий протокол тільки для TLS (HTTPS)-з'єднання. Тому тим, хто замислюється про перехід на HTTP / 2, потрібно спочатку перейти на HTTPS.

Це потрібно не тільки для HTTP / 2. У пошуку Google використання безпечного з'єднання є одним з критеріїв ранжирування. Браузери скоро будуть позначати сайти, які не підтримують https, як «небезпечні». Додамо також, що багато можливостей HTML5, наприклад, геолокація – без безпечного з'єднання будуть недоступні.

Важливою особливістю даного протоколу є просте налаштування на серверах. Коротко розглянемо базове підключення протоколу в найрозповсюдженіших серверах..

Як вже було сказано вище, більшість сучасних браузерів працюють з HTTP / 2 тільки через TLS, тому в конфігурації вашого веб-сервера повинні бути прописані відповідні налаштування. Підтримка HTTP / 2 реалізована тільки в новітніх версіях Nginx (1.9.5 і вище). Якщо у вас встановлена інша версія, вам буде потрібно оновити її.

Після цього в конфігураційному файлі /etc/nginx/nginx.conf і знайдіть в секції server наступний рядок:

```
listen 443 ssl;
```

і замініть її на:

```
listen 443 ssl http2;
```

Збережіть внесені зміни і перезавантажте Nginx:

```
$ sudo service nginx reload
```

Це все, що необхідно для налаштування HTTP / 2 в Nginx.

В Apache HTTP / 2 підтримується тільки в версіях 2.4.17 і вище. Якщо у вас встановлена більш рання версія, виконайте оновлення і підключіть модуль mod_http2. Після цього додайте в конфігураційний файл наступні рядки:

```
# For a https server
Protocols h2 http / 1.1
# For a http server
Protocols h2c http / 1.1
```

Після цього перезапустіть Apache. Ось і все – для базової установки цього цілком достатньо.

Розглянемо оптимізацію сайтів для використання нового протоколу. HTTP / 2 сумісний з HTTP / 1.1. тому ви в принципі можете не робити ніяких дій: роботі вашого сервісу нічого не загрожує.

Але в міру переходу популярних веб-серверів і веб-браузерів на HTTP / 2 ви побачите, що ваш сайт, який колись був оптимізований для збільшення швидкості завантаження сторінок і підвищення продуктивності, вже працює не так швидко, як раніше.

Багато способи оптимізації, які успішно використовувались в HTTP / 1.1, в HTTP / 2 працювати не будуть. Деякі з них буде потрібно модифікувати, а від деяких – відмовитися взагалі. Розглянемо це питання більш детально.

У HTTP / 1.1 було зручніше завантажити одне велике зображення, ніж робити безліч запитів і завантажувати багато маленьких. Це обумовлено тим, що запити ставляться в чергу один за одним. Найпоширеніший спосіб збільшення швидкості завантаження полягав в об'єднанні множинних невеликих зображень в спрайт-файл.

Спрайт повертався у відповідь на єдиний запит. Навіть якщо користувач заходив на сторінку, на якій знаходиться всього одне невелике зображення, потрібно було завантажити весь спрайт.

У HTTP / 2 з його мультиплексуванням таких проблем немає, проте використання спрайтів в певних ситуаціях може виявитися корисним. Об'єднання декількох зображень в спрайт (особливо якщо всі ці зображення знаходяться на одній сторінці) допомагає поліпшити стиснення і таким чином знизити загальний обсяг даних при завантаженні.

Ще один популярний спосіб вирішення проблеми множинних HTTP-запитів в HTTP / 1.1 – вбудовування зображень з використанням Data URI. Це істотно збільшує в розмірі таблицю стилів.

Якщо одночасно зі вбудовуванням зображень для оптимізації використовується ще й конкатенація JS і CSS, користувачеві швидше за все доведеться завантажити весь відповідний код, навіть якщо він не буде відвідувати сторінку з цими зображеннями. У HTTP / 2 така практика скоріше погіршить, а не поліпшить продуктивність.

Для оптимізації роботи сайтів часто використовується конкатенація невеликих CSS- і JS-файлів. Багато маленьких файлів об'єднуються в один великий. Таким чином вдається обійти ліміт на кількість HTTP-запитів.

Однак при використанні конкатенації може виникнути та сама проблема, що і зі спрайтами: зайшовши на якусь одну сторінку сайту, користувач завантажить всі використовувані на ньому CSS- і JS-файли (при цьому дуже ймовірно, що більшість з цих файлів йому ніколи не знадобляться). Звичайно, можна ретельно відбирати файли для кожної сторінки сайту, але це буде займати занадто багато часу.

Ще одна складність полягає в тому, що всі елементи об’єднаного файлу потрібно вичищати з кешу одночасно. Неможливо зробити так, щоб для одних елементів була виставлена одна дата закінчення терміну дії, а для інших (які до того ж і використовуються набагато частіше) – інша. Якщо змінити хоча б один рядок в CSS, термін зберігання в кеші закінчиться відразу у всіх елементів.

Чи варто користуватися конкатенацію в HTTP / 2? Якщо HTTP-запити не вимагають істотних витрат ресурсів, то без неї цілком можна обійтися. Завантаження безлічі невеликих файлів стилів ніякої проблеми не складе.

У HTTP / 1.1 є обмеження на кількість відкритих з'єднань. Щоб обійти це обмеження, доводиться завантажувати статичні ресурси з декількох піддоменів одного домену. Такий прийом називається доменним шардуванням; воно часто використовується, наприклад, для сторінок з великою кількістю зображень. Це допомагає збільшити швидкість завантаження, але разом з тим і створює додаткові проблеми. З переходом HTTP / 2 необхідність в доменному шардуванні відпадає. Ви можете запросити стільки ресурсів, скільки вам потрібно. Більш того, у випадку з HTTP / 2 шардування не поліпшить продуктивність, а призведе швидше до протилежного ефекту, так як створить додаткові TCP-з'єднання і буде заважати пріоритезації.

Висновки. Коли планувати перехід на HTTP / 2? Однозначної відповіді на це питання немає і бути не може. Ось одна підказка: регулярно переглядайте логи відвідуваності вашого сервісу. Коли ви побачите, що більша частина відвідувачів використовують підтримують HTTP / 2 браузери – можна переходити. На поточний момент підтримка HTTP / 2 реалізована в Chrome (в тому числі і в мобільній версії для Android), Firefox, Opera, Edge, Safari.

При плануванні переходу слід враховувати і особливості вашого проекту. Якщо у вас багато користувачів, які приходять до вас з мобільних пристроїв, то це означає, що вам бажано перейти на HTTP / 2 якомога швидше. На смартфонах та планшетах переваги нового протоколу будуть особливо очевидними. Однак і тут потрібно враховувати безліч нюансів: наприклад, у багатьох регіонах світу до цих пір багато користувачів браузера Opera Mini, а він HTTP / 2 поки що не підтримує.

Якщо ви плануєте запускати новий веб-сервіс, задумайтеся про перспективу переходу на HTTP / 2. Звичайно, вам ще доведеться використовувати HTTP / 1.1 протягом якогось часу, але вже зараз можете вжити заходів щодо оптимізації, які полегшать вам життя в майбутньому.

Література

1. Пелещишин А.М. Позиціонування сайтів у глобальному інформаційному середовищі / А.М. Пелещишин. – Львів : Вид-во Національного університету "Львівська політехніка", 2007. – 258 с.

2. Пасічник В.В. Глобальні інформаційні системи та технології (моделі ефективного аналізу, опрацювання та захисту даних) / Пасічник В.В., Жежнич П.І., Кравець Р.Б., Пелещишин А.М., Тарасов Д.М. – Львів : Вид-во Національного університету "Львівська політехніка", 2006. – 350 с.

3. Биктимиров М.Р. Избранные главы компьютерной безопасности / М.Р. Биктимиров, А.Ю. Щербаков. – Казань : Изд-во Казанского математического общества, 2004. – 372 с.

4. Биктимиров М.Р. Модели управления доступом в распределенных компьютерных системах : дис. … канд. техн. наук : 05.13.18 / Биктимиров М.Р. – Казань, 2008. – 137 с.

Рецензія/Peer review : 9.10.2016 р. Надрукована/Printed : 30.10.2016 р.
Рецензент: д.ф-м.н., проф.. Бедратюк Л.П.