УДК 004.42

В. С. Курко, І. С. Колесник, Т. М. Боровська

МЕТОД ПІДВИЩЕННЯ ШВИДКОСТІ ЗАВАНТАЖЕННЯ ВЕБ-СТОРІНОК

Вінницький національний технічний університет, Вінниця

Анотація. Розроблено програмний засіб для безпечного конфігурування кешування динамічного вмісту веб-сторінок, метою якого є спрощення роботи користувача в процесі конфігурування динамічного контенту, який може бути закешований, зменшення ризику кешування персональних даних, збільшення відсотку закешованих сторінок, які містять динамічний контент, зниження навантаження на основний сервер, та пришвидшення завантаження сторінок для кінцевого користувача.

Ключові слова: веб сайт, веб-сервіс, кешування, динамічне кешування, Azure, хмарні технології.

Аннотация. Разработано программное средство для безопасного конфигурирования кэширования динамического содержимого веб-страниц, целью которого является упрощение работы пользователя в процессе конфигурирования динамического контента, который может быть кэширован, уменьшение риска кэширования персональных данных, увеличение процента кэшированных страниц, содержащих динамический контент, снижение нагрузки на основной сервер и ускорение загрузки страниц для конечного пользователя.

Ключевые слова: веб-сайт, веб-сервис, кэширование, динамическое кэширование, Azure, облачные технологии.

Abstract. Developed software for secure configuration of caching of dynamic content of web pages, which aims to simplify the user's process of configuring dynamic content that can be cached, reduce the risk of caching personal data, increase the percentage of cached pages containing dynamic content, reduce the load on the main server, and speed up page loading for the end user.

Key words: website, web service, caching, dynamic caching, Azure, cloud technologies.

DOI: 10.31649/1681-7893-2021-41-1-13-19

Вступ

Зі збільшенням кількості користувачів глобальної мережі інтернет, проблема доставки об'ємного контенту в інтернеті стає все більш актуальною. В зв'язу з цим виникає необхідність пошуку методів оптимізації процесів обробки запитів та доставки вмісту веб сторінок. Особливо це актуально для контенту, який потрібно одночасно роздати великій кількості користувачів. Виникають нові і нові технології та підходи до вирішення цієї проблеми, такі як налаштування швидкої інфраструктури та використання швидкого хосту, балансування трафіку, розподілені мережі доставки контенту, використяння Gzip для зменшення об’єму файлів, мініфікація CSS та Javascrip файлів, оптимізація зображень, кешування заголовки Expores та ін. Проте, пошуки додаткових методів пришвидшення завантаження веб-сторінок продовжуються і одним з підходів для оптимізації процесу обробки та доставки контенту є пошук способу кешувати динамічно змінювані дані.

Актуальність

В сучасному інформаційному суспільстві проводяться численні дослідження, які стосуються покращення якості обслуговування користувачів інформаційних технологій. Дослідження, які стосуються безпосередньо веб-технологій показали, що користувачі відвідують більше сторінок веб-сайту, коли він завантажується швидше.

«Вебкеш» (або «кеш HTTP») — інформаційна технологія для тимчасового зберігання (кешування) вебдокументів і медіа контенту задля зменшення серверних затримок. Система вебкешу зберігає копії документів, які проходять через неї, внаслідок чого подальші запити можуть бути виконані з кешу за

певних умов. Система ж може при цьому посилатися або на програмно-апаратний комплекс, або на комп'ютерну програму [1].

Основною перевагою кешування є пришвидшення роботи веб-сторінок, це призводить до того, що більш швидкі веб-сторінки покращують якість користування, а це означає, що відвідувачі веб-сайту будуть вище оцінювати його роботу

Через високу цінність динамічного кешування як у перевагах продуктивності, так і вартості сервера, здається, що кожен веб-сайт, який існує, захоче скористатися цією практикою. Однак, оскільки документ HTML є основою всієї веб-сторінки, багато адміністраторів веб-сайтів вважають, що надто ризиковано кешувати його. Така ситуація виникла через те, що існуючі нині методи конфігурації кешу не дозволяють належним чином налаштовувати виняткове кешування динамічного контенту та перевірити і протестувати конфігурацію кешу [3].

Динамічний вміст веб-сторінок - це вміст, який не є завжди однаковим, він змінюється в залежності від користувача, для якого цей контент відображається, або з плином часу. Під час зміни контенту в залежності від користувача, контент змінюється залежно від факторів, характерних для певного користувача або групи користувачів, таких як час відвідування, місцезнаходження, пристрій, вік, стать, історія попереднього відвідування інших сторінок та інше. Такий контент також часто називають персоналізованим. Прикладом контенту, що змінюється з плином часу, можуть бути новини, пости в блозі, списки товарів в інтернет магазині, та інший контент який часто додається, оновлюється, або видаляється.

Динамічні веб-сторінки не зберігаються у вигляді статичних файлів HTML. Натомість сценарії на стороні сервера генерують HTML-файл у відповідь на події, такі як взаємодія користувачів або вхід користувача, і надсилають HTML-файл у веб-браузер. Оскільки динамічний вміст створюється на стороні сервера, він зазвичай подається з вихідних серверів, а не з кешу.

Протягом довгого періоду часу динамічний вміст вважався некешованим. З появою сучасних веб-технологій для веб-сайтів з’явилась можливість подавати динамічний вміст із кешу, значно скорочуючи затримку передачі інформації, зберігаючи при цьому взаємодію з користувачами.

Кешування динамічного вмісту веб-сторінок зводиться до зберігання повністю готової, відрендереної веб-сторінки. На цьому єтапі можна визначити, що деякий контент ми не можемо кешувати ні в якому разі, це такий контент який містить, зокрема, персональну інформацію користувача, стан його майнових рахунків, інформацію, щодо того, як він використовував веб-сайт. Прикладами таких сторінок можуть бути: особистий кабінет користувача, історія його попередніх замовлень в інтернет магазині.

Для кешування інших типів динамічного контенту потрібно використовувати декілька кардинально різних підходів. Так, для кешування контенту, що змінюється з часом, нам достатньо налаштувати час, протягом якого цей контент може бути актуальним і закешованим. Наприклад, на сайті новин, де новини публікуються раз в декілька годин, ми можемо спокійно кешувати новинну стрічку протягом 10-20 хвилин, і проводити валідацію кешу на його актуальність. Задля вирішення задачі кешування персоналізованого контенту для окремих груп користувачів може використовуватися метод кластеризації. Наприклад для кешування окремих новинних стрічок для жителів Львова та Харкова [4].

Динамічний вміст генерується сценаріями, які змінюють вміст сторінки. Запускаючи сценарії в кеші CDN, а не на віддаленому вихідному сервері, динамічний вміст можна генерувати та доставляти з кешу. Таким чином, динамічний вміст, по суті, "кешується" і не повинен обслуговуватися весь шлях від початку, зменшуючи час відповіді на запити клієнта та прискорюючи динамічні веб-сторінки [3].

Інший підхід до прискорення динамічних веб-сторінок полягає у стисненні динамічного вмісту, створеного вихідним сервером, і його доставки максимально швидко та ефективно. При динамічному стисненні вміст все ще надходить із вихідного сервера, а не з кешу, але сформовані HTML-файли значно зменшуютьсядля того, щоб вони могли швидше досягти клієнтського пристрою.

Часто велика кількість вмісту на динамічній веб-сторінці залишається послідовною для всіх користувачів і лише певні елементи на сторінці є динамічними. Це означає, що значна частина HTML-коду дублюється в кожній динамічній копії сторінки. Щоб усунути цей дефект, ряд компаній працювали разом над розробкою Edge Side Includes (ESI), мови розмітки, яка визначає, де динамічний вміст відображається на веб-сторінці. (ESI використовується на деяких CDN, але ще не прийнятий W3C, організацією, що керує веб-стандартами) [5].

Вміст із тегом ESI завантажується з вихідного сервера, а решту вмісту веб-сторінки можна кешувати. Якщо лише частина веб-сторінки генерується динамічно, а решта кешується, то тоді веб-сторінка завантажуватиметься набагато швидше, для прикладу, якби всю сторінку потрібно було створити заново для кожного користувача.

Витрати щодо забезпечення пропускної спроможності для розміщення веб-сайтів є найсуттєвішими.

Якщо 1000 відвідувачів відкриють одну й ту саму сторінку веб-сайту протягом 1 хвилини, документ потрібно буде відправити з вихідного сервера 1000 разів.

Завдяки високій тимчасовій активності на вихідних серверах менеджерам веб-сайтів необхідно планувати максимальний обсяг трафіку, який, на їхню думку, вони матимуть у певний час, в залежності від попиту користувачів. Ця ситуація виникає в залежності від різного виду трендів, які залежать від часу доби, погоди, політичних ситуацій та інших непередбачених факторів. В залежності від таких ситуацій необхідно купувати сервери для пікових періодів трафіку, навіть якщо ці піки досягаються лише в 1% випадків [2].

Коли документ кешується, кешуючий сервер є єдиним, хто робить запит до вихідного сервера: Якщо даний документ налаштовано кешувати протягом 1 хвилини, то сервер кешування зробить один запит до вихідного серверу за хвилину, незалежно від того, чи відвідує веб-сайт 10 відвідувачів чи 1000 відвідувачів за цю хвилину (рис. 1).

За рахунок цього сервери веб-сайту звільняються для критичних навантажень, збільшуючи кількість відвідувачів, яких можна обслуговувати одночасно. В той самий час це зменшує кількість серверів, які потрібно купувати у разі пікового трафіку. Це заощаджує витрати підприємства на розміщення та гарантує, що користувачі будуть звертатися саме до цього веб-сайту та динамічно отримувати інформацію щодо питань, які є актуальними на даний момент часу.

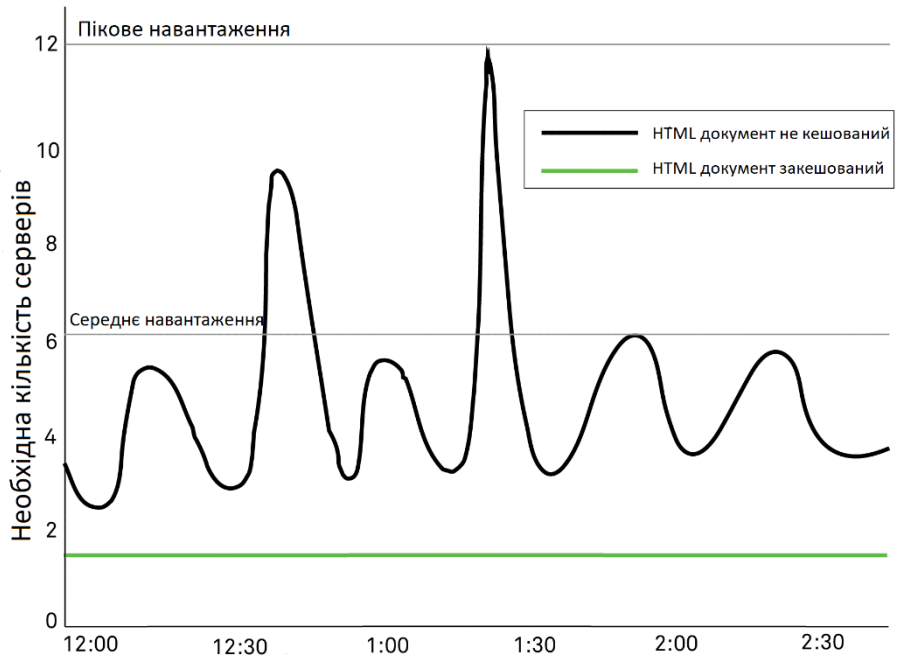

Рисунок 1 – Порівняння навантаження на вихідний сервер з та без використання кешування HTML сторінок

Час, впродовж якого ми можемо кешувати певні ресурси залежить від кількості трафіку, розміру ресурсів та розміру кеша. Крім того, слід брати до уваги головний недолік кешування – зменшення контролю над ресурсами. Якщо існує необхідність оновити дані миттєво, в залежності від потреб користувача, а саме: знижки, акції, ребренденгу, то можуть виникнути проблеми, в тому випадку, якщо для цьому ресурсу був заданий життєвий цикл.

Найтриваліший життєвий цикл для кеша - рік, або 31536000 секунд. Але задавати таке значення не рекомендується. Якщо в цьому є необхідність – то найкращим рішенням буде підвищити продуктивніть та пропускні можливості сервера а не кешу.

Результати вимірювань втрати часу на кеш показано в табл. 1. Для вимірювання до ресурсу надсилалося 500 запитів в хвилину.

Таблиця 1 – Порівняння еількості запитів до кешу, за різних налаштувань тривалості часу кешування.

Час кешування (хв)	Відсоток запитів до кешу	Запитів до оригіналу в годину
1	99.8%	60
5	99.96%	12
20	99.99%	3
60	99.97%	1
86400	99.9998%	<1

Управління кешуванням та валідація кешу

На сьогоднішній день одним з найпопулярніших підходів до управління кешуванням вмісту сторінок є заголовки запиту Http протоколу.

Він містить поле Cache-Control що використовується для завдання інструкцій по механізму кешування як в запитах, так і у відповідях. Застосовується для завдання політик кешування, серед яких

- Повна відсутність кешування

Параметри no-cache, no-store, must-revalidate

- Кешувати, але перевіряти актуальність

Параметри: no-cache

- Приватний (private) і загальний (public) кеш
- Термін дії (Expiration)

Найважливішою тут є директива "max-age = <seconds>" - максимальний час, протягом якого ресурс вважається "свіжим" та директиви Expires, вона прив'язана до моменту запиту.

- Перевірка актуальності - параметер "must-revalidate"
- Заголовок Pragma

Крім того, оскільки обсяг сховища кешу кінцевий, а ресурси на сервері можуть змінюватись то кеш потрібно чистити та оновлювати. Для цього доступні заголовки ETag та заголовок HTTP відповіді Vary.

Даний метод управління кешем є досить складним, для його грамотного використання потрібно мати неаби які технічні знання, і він не є достатньо гнучким для використання його в управлінні кешуванням динамічного контенту. Для цього необхідний інший підхід, який буде більш інтуїтивно зрозумілий та гнучкий.

Мета

Метою статті є представлення методу збільшення швидкості завантаження веб-сторінок, який реалізується за рахунок створення алгоритмів і створення архітектури програмного засобу управління кешуванням динамічного вмісту веб-сторінок. Даний програмний засіб повинен задовольняти наступні критерії:

- зручний графічний інтерфейс;
- інтуїтивно зрозумілий та структурований покроковий алгоритм налаштувань;
- можливість детального налаштування кешування для кожної сторінки, або групи сторінок;
- широкий вибір параметрів персоналізації та коректної валідації кешу;
- простий спосіб інтеграції програмного засобу з більшістю платформ та методологій побудови веб-сайтів.

Особливості побудови програмного засобу для конфігурування кешування динамічного вмісту веб-сторінок

Для реалізації поставленої мети було розроблено інтелектуальний модуль аналізу параметрів та налаштуваннь для оцінювання можливості кешування та правильного способу кешування веб сторінки, що містить динамічний контент. Загальна схема роботи такого програмного забезпечення зображена на рис. 2.

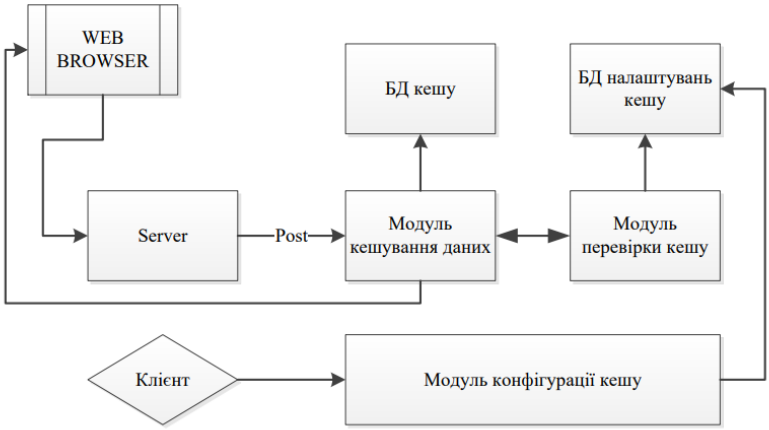

Рисунок 2 – Загальна схема програмного засобу для кешування динамічного вмісту веб-сторінок

Клієнт, або власник веб-сайту за допомогою «Модулю конфігурації кешу» та зручного інтерфейсу налаштовує як він хоче кешувати кожну окрему сторінку сайту і чи хоче взагалі. Далі ці дані потрапляють до «БД налаштувань кешу», де міститься інформація по налуштуванням для кожної сторінки.

Коли веб-браузер звертається до веб-серверу, той, в свою чергу, відправляє запит до «модулю кешування данних». Цей модуль відповідає за збереження та отримання сторінок з кешу. Крім того він звертається до «Модулю перевірки», що необхідний для того, щоб визначити, чи можна кешувати сторінку за поданим запитом.

Якщо можна, тоді модуль кешування дістає сторінку з кешу. Якщо не можна - сервер самостійно збирає сторінку. А якщо можна, але такої сторінки в кеші поки що немає, то сервер самостійно збирає сторінку, і перед відправленням її користувачу, передає її до «модулю кешування данних» і той кешує її.

На разі, практично всі великі програмні системи, є розподіленими. Розподіленою називається така система, в якій обробка інформації зосереджена не на одній обчислювальній машині, а розподілена між декількома комп'ютерами. Клауд сервіси, такі як Azure дозволяють вирішити питання, які дадуть нам можливість створити декілька екземплярів кожного модуля, якими необхідно користуватись, фізично розмістити їх в будь-якій точці, якомога ближче до користувача, налаштувати автоматичне масштабування та безпечну і стійку комунікацію між різними компонентами.

Microsoft Azure надає декілька різних способів розміщення та виконання коду або робочих процесів без використання віртуальних машин (ВМ), включаючи Azure функції. Функція Azure — це простий спосіб запускати невеликі фрагменти коду в хмарі, не турбуючись про інфраструктуру, необхідну для розміщення цього коду. Функцію можна написати на C#, Java, JavaScript, PowerShell, Python чи багатьох інших популярних мовах програмування. Azure автоматично масштабує функцію у відповідь на запит користувачів.

Якщо програма складається з компонентів, запущених на різних комп’ютерах, серверах і мобільних пристроях, зв’язок між цими компонентами може бути складним і ненадійним. Azure надає кілька технологій, які можна використовувати для надійнішого зв’язку, зокрема черги Storage queues, Event Hubs, Event Grid, and Service Bus.

Саме тому було прийняте рішення будувати даний додаток на базі хмарних сервісів Azure, а для комунікації між компонентами використовувати Azure Event Grid та Azure Service Bus.

Налаштування кешу

Для початку клієнт завантажує сайтмап свого сайту, після чого всі сторінки сайту будуть графічно відображені в деревовидній структурі. Далі клієнт проходиться по всіх сторінках, обирає які сторінки кешувати не можна, конфігурує параметри за якими можна кешувати сторіку та налаштовує валідацію кешу.

Окрім того, модуль конфігурації кешу буде генерувати бази даних та інші модулі в хмарі у вигляді Azure сервісів, з використанням всіх переваг та найсучасніших технологій побудови розподілених програмних систем, які пропонує Azure. Всі сервіси будуть створюватися в тій кількості, яку захоче клієнт, і до того ж розташовуватися фізично найближче до кінцевого користувача. Завдяки цьому буде зменшено час відповіді та підсилено пропускову здатність всього додатку.

Розробка алгоритму роботи модулю перевірки кешу

Модуль перевірки кешу (рис. 3) отримує запит у форматі Json. Далі перевіряє чи запитується саме html сторінка, а не статичний ресурс чи щось інше, чи можна кешувати сторінку принаймні за деяких обставин, чи немає в запиті параметрів, що забороняють кешування, чи є кеш валідним, і вкінці кінців повертає позитивний або негативний результат.

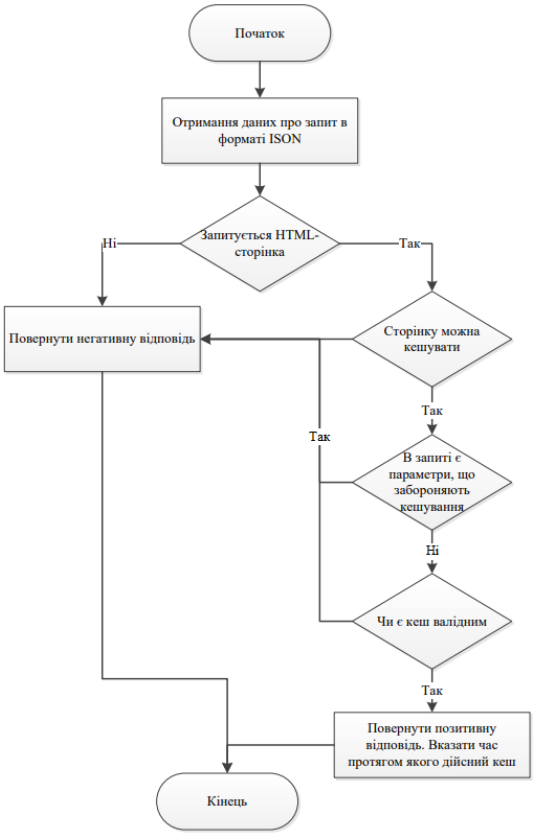

Рисунок 3 – Алгоритм роботи модулю перевірки кешу

Висновки

Розроблений підхід щодо прискорення відображення динамічного контенту відбувається за рахунок створення власної розподіленої системи для налаштування кешування динамічного вмісту веб сторінок, на основі хмарних технологій. Дана система дозволяє безпечно кешувати можливості використання HTML сторінки, що значно знижує навантаження на вихідний сервер, в наслідок чого зменшується вартість хостингу веб-сайтів. За рахунок використання хмарних технологій, розроблений додаток є більш гнучким, легко масштабованим та надійним, внаслідок чого витрати на обслуговування додатку є також значно зниженими. Данний програмний засіб рекомендовано використовувати лише для великих систем, з великою кількістю даних, ресурсів та сторінок. Якщо ж мова йде про невеликий сайт, то витрати на впровадження такого програмного рішення можуть бути більшими, ніж його користь.

Список літератури

[1] Adrian Popescu, David Erman, Dragos Ilie, Doru Constantinescu: Internet Content Distribution: Developments and Challenges, 2018.
[2] Т. М. Боровська, І. С. Колесник, В. А. Северілов: Оптимізація стратегій розвитку розподілених виробничих систем на базі агрегування виробничих функцій, Вісник Вінницького політехнічного інституту, 2010.Фіва
[3] Pallis G. and Vakali A., Insight and Perspectives for Content Delivery Networks, Communications of the ACM, Vol 49, No 1, January 2006

[4] Cui Y., Li B. and Nahrstedt K., oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks, IEEE Journal onSelected Areas in Communications, Vol 22, No 1, January 2004
[5] G. Peng, CDN: Content Distribution Network, Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY 2003.

References

[1] Adrian Popescu, David Erman, Dragos Ilie, Doru Constantinescu: Internet Content Distribution: Developments and Challenges, 2018
[2] T. M. Borovska, I. S. Kolesnyk, V. A. Severilov: Optymizatsiia stratehii rozvytku rozpodile-nykh vyrobnychykh system na bazi ahrehuvannia vyrobnychykh funktsii, Visnyk Vinnytskoho poli-tekhnichnoho instytutu, 2010.Fiva
[3] Pallis G. and Vakali A., Insight and Perspectives for Content Delivery Networks, Communications of the ACM, Vol 49, No 1, January 2006
[4] Cui Y., Li B. and Nahrstedt K., oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks, IEEE Journal onSelected Areas in Communications, Vol 22, No 1, January 2004
[5] G. Peng, CDN: Content Distribution Network, Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY 2003.

Відомості про авторів

Курко Владислав Сергійович – студент групи 2КІ-20м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, Хмельницьке шосе 95, email: **v.kurko2012@gmail.com**

Колесник Ірина Сергіївна - к.т.н, доцент кафедри обчислювальної техніки, Вінницький національний технічний університет, Вінниця, Хмельницьке шосе 95, email: **kolesnyk.iryna@vntu.edu.ua**

Боровська Таїса Миколаївна – д.т.н., професор кафедри комп'ютерних систем управління, Вінницький національний технічний університет, Вінниця, Хмельницьке шосе 95, email: **taisaborovska@vntu.edu.ua**