ARTICLE

DOI 10.36074/grail-of-science.27.05.2022.055

ВІДМОВОСТІЙКІСТЬ ТА АВТОМАСШТАБУВАННЯ ВЕБ-РЕСУРСУ

Тетяна Іванівна Коробейнікова
канд. техн. наук, доцент,
доцент кафедри безпеки інформаційних технологій
Національний університет «Львівська політехніка», Україна

Захарченко Сергій Михайлович
канд. техн. наук, доцент, доцент кафедри обчислювальної техніки
Вінницький національний технічний університет, Україна

Анотація. *Метою даної роботи є дослідження питання прискорення роботи серверних додатків під час високих навантажень та автоматизації процесів масштабування серверних систем, що дозволить знизити витрати на інфраструктуру та її обслуговування.*

Ключові слова: *перевантаження, зниження продуктивності ресурсів, відмовостійкість веб-ресурсів, масштабування/автомасштабування веб-ресурсів.*

Історія веб-ресурсів починається з 1990-х років, а зараз маємо кілька поколінь еволюції WWW. Є складні розподілені системи, які виконують складні обчислення та обробку інформації. Для керування роботою ПЗ та сервера в комплексному режимі використовуються додаткові системи та підсистеми. Таким чином, світ прискорюється, інформація має бути доступною в будь-який момент, і компанії, які не врахували такі вимоги ринку, втратили вплив у IT-світі [1-2].

Зросла кількість користувачів, питома вага Інтернету зростає з кожним днем, кількість пристроїв, підключених до Інтернету зростає щосекунди, люди не можуть уявити роботу з електронікою, яка не має доступу до мережі [3-4].

Зростає вектор технологічного розвитку та Інтернет і це змушує компанії зосереджуватися на швидкості та надійності продукції, залучати висококваліфікованих фахівців з перших етапів розробки продукту. Зараз неможливо уявити собі повноцінний сервіс, який використовує тільки HTML і CSS, оскільки вони не створюють динамічні програми. У наш час ключовим моментом є сервер, який складається з десятків і сотень інших підсистем/додатків/мікросервісів [2-4]. Навіть простий сервер містить принаймні одну програму, веб-сервер, який організовує доступ до цієї програми, базу даних, де зберігається вся інформація про користувача та систему. Як варіант, така система може включати сервер черги, сервер кешу та багато інших систем [5].

Успіх IT-проекту визначається кількістю користувачів, які він обслуговує, оскільки це безпосередньо впливає на прибуток проекту. Легко побачити величезну кількість користувачів, які використовують одну програму в один момент часу [6]. На практиці більшість проектів не готові до природного зростання. Коли система руйнується під перевантаженням – з'являється багато різних проблем перевантаження. Найпоширенішими проявами перевантаження є: зниження продуктивності ресурсу, збільшення кількості помилок, вихід з ладу обладнання, тимчасова недоступність деяких компонентів. Всі ці прояви впливають на високорівневий показник – час доступності ресурсу (або безвідмовний час). Саме тому користувачі відмовляються від такого ресурсу на користь конкурентів

Існує суперечність між постійним зростанням користувачів і відмовостійкістю в процесах розробки та підтримки веб-ресурсів, що робить *проблему перевантаження додатків* актуальною та має шляхи для розвитку в науковій сфері.

Концепція відмовостійкості в процесах розробки та обслуговування web-додатків. Перевантажена програма стає нестабільною, відновлення будь-якого компонента вимагає часу, а іноді навіть призводить до пошкодження/втрати даних. Перевантажена система зазвичай залишається недоступною, поки процес відновлення не завершить фахівець. Відмовостійкість – це відсутність простою ресурсів через непередбачувані збої програмного/апаратного забезпечення або проблеми мережевого рівня OSI. Відмовостійка система – це система, яка зберігає свою працездатність у разі нестандартних ситуацій; відновлення недоступних вузлів відбувається в ручному або автоматичному режимі. Невідмовостійка система – це система, яка не має можливості самостійно відновлювати пошкоджені компоненти через ненормальну ситуацію.

Відновлення після збоїв у відмовостійких системах можна описати як запуск або відкат. Коли система виявляє помилку, запуск виправляє поточний стан системи, щоб продовжити. Відкат відновлює систему до попередньої версії, наприклад, за допомогою контрольних точок. Відкат вимагає, щоб операції між контрольними точками та виявленим статусом помилки були ідемпотентними. Деякі системи використовують обидва підходи для різних помилок або для різних частин однієї помилки.

Основними характеристиками відмовостійкої системи є:

– Більш ніж одна точка відновлення (перехоплення контролю перемикання збоїв може виконуватися одним із двох серверів);

– Локалізація пошкодження в пошкодженому компоненті;

– Обмеження впливу на інші системи та поширення несправності;

– Доступність режимів відкату.

Відмовостійкість пов'язана з підтримкою безперервності бізнесу за допомогою високодоступних комп'ютерних систем і мереж. Відмовостійкі середовища відновлюють службу відразу після вимкнення служби, тоді як середовище високої доступності, як правило, доступне 99,999% часу.

У кластері високої доступності незалежні набори серверів об'єднані для забезпечення загальносистемного обміну критичними даними та ресурсами.

Кластери відстежують продуктивність один одного та забезпечують усунення несправностей, щоб гарантувати, що програми доступні. І навпаки, кластер відмов складається з кількох фізичних систем, які спільно використовують одну копію серверного програмного забезпечення. Команди програми, що виконуються однією системою, також виконуються в іншій системі.

Відмовостійкість залежить від резервування. Інформація надто захищена через реплікацію даних або синхронне дзеркальне відображення томів в інший центр обробки даних. Для фізичного резервування додаткове обладнання залишається в режимі очікування для швидкого введення в експлуатацію.

Резервне копіювання даних часто поєднується з резервуванням. Обидві стратегії розроблені для захисту від втрати даних, хоча резервне копіювання зазвичай зосереджено на відновленні з часом, включно з відновленням даних. Резервні системи розроблені для навантажень додатків, які мають дуже малий час простою. Резервне копіювання не може замінити надмірність даних і навпаки. Архітектура системи відмовлення повинна бути доповнена регулярним резервним копіюванням критичних даних, можливо, включаючи дзеркальне відображення (на вторинному або альтернативному сервері). Безпека має бути частиною планування для запобігання несанкціонованого доступу.

Масштабування та автомасштабування при розробці та супроводі веб-ресурсів. Оскільки однією з основних умов побудови відмовостійкої системи є наявність більше однієї точки відновлення для кожного компонента, розробка такого компонента є *актуальною*.

Масштабованість – це здатність системи впоратися з зростаючим навантаженням за рахунок збільшення системних ресурсів. В економічному контексті масштабована бізнес-модель передбачає, що компанія може збільшити продажі за рахунок збільшення ресурсів. У комп'ютерних системах масштабованість є характеристикою комп'ютерів, мереж, алгоритмів, мережевих протоколів, програм і додатків. Наприклад, пошукова система, яка підтримує збільшення кількості користувачів і кількості тем, які вона індексує.

Масштабність можна виміряти за допомогою таких вимірювань [7]:

–Адміністративна масштабованість: можливість збільшення кількості користувачів для доступу до системи;

–Функціональна масштабованість: можливість вдосконалювати систему шляхом додавання нових функцій без порушення поточної діяльності;

–Географічна масштабованість: можливість ефективної підтримки в той час як область розширюється від локальної до більшої;

–Масштабованість навантаження: здатність розподіленої системи розширюватися/звужуватися для обробки більших/менших навантажень, легко змінювати, додавати або видаляти компонент для задоволення змінних навантажень;

–Масштабування Generatoin: здатність системи масштабуватися за допомогою нових поколінь компонентів;

–Гетерогенна масштабованість: здатність приймати компоненти від різних постачальників.

Розрізняють два типи масштабування: горизонтальне та вертикальне [7]. Можна використовувати обидва типи, однак правильно розроблена система масштабується відразу двома способами, це залежить від кожного компонента завдання та потреб.

Горизонтальне масштабування означає додавання/вилучення вузлів до/з системи, наприклад, додавання нового комп'ютера чи будь-якого іншого пристрою до розподіленого програмного забезпечення. Інший приклад – масштабування з одного веб-сервера до трьох. Високопродуктивні обчислювальні програми (наприклад, математичний компонент обробки контенту [8-9], обробки кодової послідовності [10], графічні перетворення [11-12] та багато інших завдань, які потребують високої обчислювальної потужності) масштабуються горизонтально для підтримки завдань це вимагало б величезних повноважень. Дуже популярні нині соціальні мережі, які перевищують потужність суперкомп'ютерів і обробляються лише масштабованими системами. Використання цього типу масштабування вимагає програмного забезпечення для ефективного управління ресурсами та їх підтримки [13].

Вертикальне масштабування (вгору/вниз) означає додавання/вилучення ресурсів до/з окремого вузла (процесори, оперативна пам'ять або енергонезалежна пам'ять). Вертикальне масштабування використовується, коли горизонтальне масштабування вимагає більше зусиль або грошей. Зі збільшенням кількості елементів, що збільшується складність управління, деякі програми не масштабуються горизонтально.

Масштабування бази даних вимагає, щоб система баз даних могла виконувати додаткові завдання з урахуванням більших апаратних ресурсів, таких як додаткові сервери, процесори та пам'ять. Робочі навантаження продовжують зростати, і вимоги до баз даних слідують цій тенденції. Алгоритмічні нововведення включають блокування на рівні рядків і розбиття таблиць та індексів. Архітектурні інновації включають архітектури «нічого спільного» та «все спільного» для керування конфігурацією кількох серверів.

У контексті масштабування зберігання даних, масштабованість – це максимальний розмір кластера зберігання, який гарантує узгодженість даних, тобто існує лише одна дійсна версія збережених даних у всьому кластері, незалежно від кількості надлишкових копій даних. Кластери, які забезпечують "ліниве" резервування шляхом асинхронного оновлення копій, називаються "в кінцевому рахунку несуперечливими". Цей тип дизайну масштабування підходить, коли доступність і реагування оцінені вище, ніж узгодженість, що справедливо для багатьох файлообмінників або веб-кешів (якщо вам потрібна остання версія, зачекайте кілька секунд, поки вона поширюється). Такого дизайну слід уникати для всіх класичних додатків, орієнтованих на транзакції.

Більшість відкритих і комерційних кластерів сховища на основі стандартних мереж забезпечують лише можливу узгодженість (NoSQL, CouchDB та ін.). Операції запису скасовують інші копії. Операції читання зазвичай не перевіряють кожну зайву копію перед відповіддю, тому потенційно немає попередніх операцій запису. Високий трафік сигналу метаданих вимагає спеціалізованого обладнання та коротких відстаней для обробки з прийнятною продуктивністю (тобто як некластеризовані пристрої зберігання даних або БД).

У сфері високопродуктивних обчислень існують дві загальні концепції масштабованості. Перший – це сильна масштабованість, яка визначається як зміна часу вирішення в залежності від кількості процесорів для фіксованого загального розміру проблеми. Другий – слабке масштабування, яке визначається як зміна часу прийняття рішення залежно від кількості процесорів для фіксованого розміру завдання на процесор.

Автомасштабування – це метод хмарних обчислень, тому кількість обчислювальних ресурсів у фермі серверів (яка зазвичай вимірюється кількістю активних серверів) автоматично масштабується на основі загального навантаження ферми. Автомасштабування пов'язане з ідеєю балансування навантаження і спирається на неї.

Автомасштабування пропонує такі переваги: для компаній, які мають власну інфраструктуру веб-серверів, автомасштабування зазвичай означає, що деякі сервери можуть «засинати» при низькому навантаженні, економлячи енергію (а також воду, якщо вода використовується як охолоджувач). Для компаній із хмарною інфраструктурою автомасштабування забезпечує менші рахунки, оскільки більшість постачальників хмарних послуг стягують плату на основі загального використання, а не максимальної потужності. Навіть для компаній, які не можуть зменшити загальну обчислювальну потужність, яку вони виконують або платять у будь-який момент часу, автомасштабування може допомогти, дозволяючи компаніям працювати на менш трудомістких робочих навантаженнях на ПК (які вивільняються за допомогою автоматичного масштабування під час низьких навантажень). Рішення автоматичного масштабування (Amazon Web Services) також можуть забезпечити заміну проблемних екземплярів серверів і, таким чином, забезпечити певний захист від збоїв обладнання/програмного забезпечення/мережі.

Автоматичне масштабування відрізняється від фіксованого щоденного/тижневого/річного циклу використання сервера, оскільки воно реагує на фактичні моделі використання і, таким чином, зменшує потенційний недолік у наявності занадто малої або занадто великої кількості серверів для завантаження трафіку. Наприклад, якщо трафік зазвичай нижчий вночі, рішення для статичного масштабування може призвести до сну деяких серверів вночі, але це може призвести до простою вночі, коли люди більше використовують Інтернет (наприклад, через вірусні події, новини). З іншого боку, автомасштабування дозволяє краще справлятися з несподіваними піками навантаження.

Автомасштабування за замовчуванням використовує реактивний підхід до прийняття рішень для масштабування трафіку: масштабування відбувається лише в режимі реального часу, коли змінюються показники. У деяких випадках, особливо коли зміни відбуваються дуже швидко, цього реактивного підходу до масштабування недостатньо, тому альтернативою є автоматичне планування або автомасштабування з прогнозом.

Автоматичне масштабування за розкладом – зміни вносяться через мінімальний/максимальний розмір або бажану потужність групи автоматичного масштабування в точний час доби. Планове масштабування корисно, якщо відоме навантаження трафіку збільшується/зменшується в точний час доби, але зміна надто раптова при реактивному підході.

Підхід прогнозування автоматичного масштабування використовує прогнозну аналітику. Ідея полягає в тому, щоб об'єднати останні тенденції використання з тенденціями використання в минулому, а також інші типи даних, щоб передбачити майбутні потреби в масштабуванні.

Забезпечення відмовостійкості та автомасштабування веб-ресурсів. Забезпечення відмовостійкості веб-програми представлено в запропонованих властивостях кожної окремої частини веб-ресурса:

Механізм обміну даними між програмою та базою даних. Веб-ресурс описує основну бізнес-логіку продукту; це масштабування досить просте, але переконайтеся, що програма «stateless» (без збереження стану). Необхідно забезпечити деякі оптимізації з компонентами, де можна зберігати користувацькі файли, сесії, спільні системні файли.

Користувацькі файли, завантажені на сервер користувачами і належать їм. Якщо користувацькі файли збережені у файловій системі (за замовчуванням) – програма не є «stateless», тому потрібно перемістити ці файли за межі файлової системи.

Існує два підходи:

1. спільна файлова система (GlusterFS, CephFS) або
2. стороння сервісна система (Amazon S3).

У першому випадку перевагами є: простота налаштування, відсутність необхідності вносити зміни в код веб-програми; недоліки: затримка перед розсилкою файлу на всі підключені сервери та ймовірні проблеми при високих навантаженнях. У другому випадку перевагами є: автоматичне кешування всіх файлів і швидка доставка в будь-який регіон світу з низькою ціною за гігабайт пам'яті (порівняно з жорстким диском на сервері); недоліками є: необхідно доповнити програму змінами коду, що перенаправляє запис файлу на стороннє сховище, затримка запису файлу (порівняно з жорстким диском). При використанні Service Mesh за замовчуванням можна створити спільне сховище файлів, тому достатньо підготувати томи з даними користувача для вилучення. Недоліками є необхідність доповнювати код програми змінами, які перенаправляють запис файлу на стороннє сховище, затримка запису файлу (у порівнянні з жорстким диском). Під час використання Service Mesh за замовчуванням можна створити спільне сховище файлів, тому в цьому випадку вам потрібно лише підготувати томи з даними користувача для вилучення. Недоліками є необхідність доповнити код програми змінами, які перенаправлять запис файлу на стороннє сховище, затримка запису файлу (порівняно з жорстким диском). Під час використання Service Mesh за замовчуванням можна створити спільне сховище файлів, тому в цьому випадку вам потрібно лише підготувати томи даних користувача, щоб розмістити їх за межами.

Сесія – це тимчасовий набір інформації про користувача для ідентифікації його в різних частинах програми під час його сеансу. Сеанс відбувається у файловій системі Server_1, і якщо балансувальник навантаження перемикає його на інший Server_2, це призводить до «забуття» користувача (виходу), тому балансувальник навантаження перемикає користувача на інший екземпляр веб-програми. Єдиним рішенням є розміщення сеансів в головній базі даних або в базі даних «ключ-значення». Зазвичай вибирається другий варіант,

оскільки швидкість досить висока, а збереження-отримання даних сеансу зазвичай не вимагає складного розташування реляційних даних.

Спільні системні файли. Іноді програма може створювати певні файли, які сигналізують про різні стани, наприклад «заблоковані файли» (блокування ресурсів). У цьому випадку необхідно змінити логіку програми, щоб програма не використовувала такі конкретні файли, а використовувала зовнішні бази даних для налаштування станів програми.

Тепер програма готова до контейнеризації та подальшого масштабування. Вам потрібно лише вибрати метод балансування http-трафіку в додатку: 1. Балансування навантаження, 2. Балансування DNS, 3. Балансування трафіку мережевого рівня OSI. Використання балансувальника навантаження (nginx) вирішує багато проблем: простий алгоритм вибору сервера; фільтрація поганих запитів; опитувати сервери додатків, щоб визначити їх статус (готові вони прийняти запит чи ні) (рис. 1).

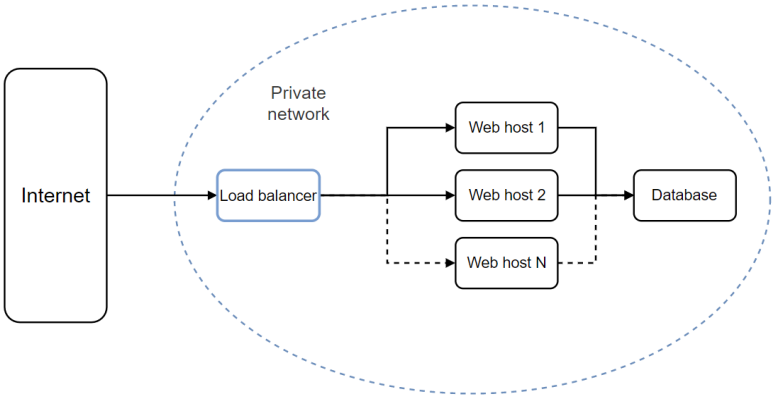

Рис. 1. Веб-ресурс використовує балансувальник навантаження

Недоліками використання балансувальника навантаження є: необхідність його встановлення та налаштування, а для такого програмного забезпечення зазвичай потрібен окремий сервер. Балансування DNS набагато простіше – для одного домену вказано кілька серверів, але можливість налаштування обмежена, тому цей тип балансування використовується для балансування кількох балансувальників навантаження, коли один з них більше не витримує високого навантаження.

Висновки. У цій статті досліджуються механізми забезпечення стійкості для типових веб-ресурсів, які використовують базу даних, і обговорюються проблеми, які можуть виникнути під час підготовки до масштабування для різних компонентів програми. Найефективнішим способом створення відмовостійкого веб-ресурсу є масштабування, тоді як автомасштабування може значно скоротити час ручного налаштування системи фахівцем і значно скоротити час відповіді на збільшення потоку користувачів.

Список використаних джерел:

[1] W. Iqbal, M. N. Dailey, D. Carrera. (2016) Unsupervised Learning of Dynamic Resource Provisioning Policies for Cloud-Hosted Multitier Web Applications. *IEEE Systems Journal* (10, №4), 1435-1446.

[2] E. G. Radhika, G. Sudha Sadasivam, J. Fenila Naomi. (2018) An Efficient Predictive technique to Autoscale the Resources for Web applications in Private cloud. *Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB).* 27-28 February 2018, Chennai, India.

[3] Q. Wang, H. Chen, S. Zhang, L. Hu,B. Palanisamy. (2019) Integrating Concurrency Control in n-Tier Application Scaling Management in the Cloud. *IEEE Transactions on Parallel and Distributed Systems* (30, №4), 855-869.

[4] H. Chen, Q. Wang, B. Palanisamy, P. Xiong. (2017) DCM: Dynamic Concurrency Management for Scaling n-Tier Applications in Cloud. *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 5-8 June 2017, Atlanta, Georgia, USA.

B. Kan. (2016) DoCloud: An elastic cloud platform for Web applications based on Docker. *18th International Conference on Advanced Communication Technology (ICACT).* 11-14 February 2016, Chuncheon, South Korea.

[5] Гороховський О. І., Трояновська Т. І., Азаров О. Д. (2016) *Інформаційна технологія доставки контенту у системах комп'ютеризованої підготовки спеціалістів*. Вінниця : ВНТУ.

[6] *Scalability.* (2022). Вилучено з : https://en.wikipedia.org/wiki/Scalability.

[7] О. І. Гороховський, Т. І. Трояновська. (2009). Інформаційна технологія побудови адаптивної системи дистанційного навчання. *Наукові праці ВНТУ* (2), 1-7. Вилучено з : http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/4403/126.pdf?sequence=3&isAllowed=y.

[8] Трояновська Т. І. (2013). Метод обробки даних дослідження індивідуальних характеристик суб'єкта СКП спеціалістів. *Вісник ВПІ* (4), 140–146.

[9] Volodymyr A. Luzhetsky, Liudmyla A. Savytska, Tetiana I. Troianovska, Zbigniew Omiotek, Aron Burlibay, Miergul Kozhambardiyeva, Gulzhan Kashaganova. (2017) Adaptive compression methods of data based on Fibonacci linear forms. *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments.* 7 August 2017, Wilga, Poland.

[10] Vyatkin, S.I., Romanyuk, A.N., Savytska, L.A., Troianovska, T.I., Dobrovolska, N.V. (2018) Real-Time Deformations of Function-Based Surfaces using Perturbation Functions. *Journal of Physics: Conference Series. International Conference Information Technologies in Business and Industry*. (1015), 1-6. 18-20 January 2018, Tomsk, Russia.

[11] Vyatkin, S., Romanyuk, A., Troianovska, T., Tsikhanovska, O., Nechiporuk, M., & Lysenko, I. (2019) Convolution surfaces using volume bounding. *9th International Conference on Advanced Computer Information Technologies, ACIT*. (8779894), 461-465. June 5-7, 2019, Ceske Budejovice, Czech Republic.

[12] С. М. Захарченко, Т. І. Трояновська, О. В. Бойко, В. С. Рибаченко (2016) Застосування односторінкових веб-орієнтованих інтерфейсів в соціально значущих проектах. *Вісник ХНУ* (3), 33-39.