Скольбуденко А. В., здобувач[2]
першого (бакалаврського) рівня
ОП «Комп'ютерні науки»
Житомирський державний університет
імені Івана Франка

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ОБРОБКИ ДАНИХ У ВЕБСИСТЕМАХ З ВІДКРИТИМИ БАЗАМИ ДАНИХ

Вступ. На сучасному етапі розвитку інформаційних технологій масштабовані вебсистеми відіграють фундаментальну роль у забезпеченні безперебійного функціонування цифрової економіки, інтелектуальної логістики та розподілених систем Інтернету речей. В умовах експоненційного зростання кількості клієнтських пристроїв обсяги інформації, які необхідно обробляти у режимі реального часу, невпинно збільшуються. Особливо гостро проблема стабільності та відмовостійкості постає для розподілених програмних рішень, архітектура яких передбачає глибоку інтеграцію з відкритими базами даних та сторонніми прикладними програмними інтерфейсами (API), такими як метеорологічні чи фінансові сервіси [1]. Специфіка взаємодії з публічними інтерфейсами полягає у наявності невідворотних апаратно-мережевих затримок (latency) та встановленні жорстких лімітів на частоту запитів з боку провайдерів даних. При виникненні пікових навантажень пряме синхронне переспрямування клієнтських запитів до зовнішніх джерел спричиняє критичне вичерпання серверних ресурсів, перевантаження центрального процесора внаслідок масового контекстного перемикання потоків та формування довготривалих черг. Для запобігання цієї проблематики сучасні системні архітектори активно втілюють механізми In-Memory кешування, де галузевим еталоном є Redis, у поєднанні з шаблонами інтелектуального спрямування. Однак, розгортання таких багатошарових рішень без попереднього обчислювального обґрунтування є небезпечним. Тому створення імітаційної моделі для нагляду та вдосконалення процесів обробки відомостей у вебсистемах є надзвичайно актуальним науково-прикладним дорученням.

Виклад основного матеріалу. Для всебічного розуміння процесів, котрі спричиняють погіршення ефективності вебсистем при критичних навантаженнях, потрібно детально дослідити фізичну сутність затримок у глобальних мережах. Загальний час відгуку системи консолідує в собі затримку поширення сигналу, апаратну затримку комутаційного обладнання, час перебування у чергах маршрутизаторів та, власне, затримку обробки інформації базою даних. Доведено, що при інтеграції з відкритими даними домінуючою складовою найчастіше стає мережева затримка (Propagation Delay). Враховуючи константу швидкості поширення світла в оптоволоконних лініях зв'язку, фізична передача пакету даних (Round Trip Time) на великі відстані займає значний проміжок часу, що формує фундаментальне обмеження, яке неможливо подолати екстенсивним шляхом [2].

Другою, не менш значущою проблемою є архітектура керування потоками виконання операційної системи (OS Thread Management). У класичних моделях на кшталт «один потік на один запит», виклик до повільного зовнішнього API змушує серверний потік переходити у стан блокування. В умовах надвисокого навантаження операційна система ініціює тисячі процесів збереження та відновлення станів (Context Switching). Цей процес є надзвичайно ресурсозатратним: центральний процесор (CPU) витрачає левову частку циклів не на виконання бізнес-логіки, а на адміністрування заблокованих потоків (явище т. зв. «thrashing»). Це неминуче призводить до різкого спаду пропускної здатності та зростання часу відгуку до неприпустимих значень.

[2] Науковий керівник – завідувач кафедри комп'ютерних наук та інформаційних технологій Житомирського державного університету ім. Івана Франка, кандидат педагогічних наук, доцент Усата Олена Юріївна

Математичним базисом для дослідження цих станів слугує теорія масового обслуговування (ТМО). Будь-який сервер або API Gateway можна описати як систему масового обслуговування, де вхідний трафік моделюється пуассонівським потоком подій з інтенсивністю λ. Відповідно до закону Літтла, при фіксованій пропускній здатності зовнішньої бази даних (μ), будь-яке зростання користувацького трафіку зумовлює експоненційне збільшення часу затримки [3]. Якщо коефіцієнт завантаження системи перевищує одиницю, система переходить у нестаціонарний стан: виникає нескінченна черга, що споживає оперативну пам'ять сервера аж до критичного збою (Out Of Memory).

Для практичного підтвердження теоретичних викладок було спроєктовано та програмно реалізовано імітаційну модель маршрутизації даних у вигляді спеціалізованого односторінкового вебдодатка (SPA) – «Моніторинг навантаження на API». Архітектура моделі базується на мікросервісних патернах та використовує метод дискретно-подієвого моделювання. Вона включає інтерактивний генератор стохастичного користувацького трафіку, імітатор API Gateway як єдиної точки входу, імітатор повільної відкритої бази даних (з базовою затримкою 50 мс) та підсистему In-Memory кешування (Redis), яка реалізує алгоритм Cache-Aside. Затримка доступу до Redis у моделі зафіксована на рівні 1 мс, оскільки дані обробляються безпосередньо в оперативній пам'яті (RAM) без необхідності парсингу складних SQL-структур [4].

Основним завданням розробленого програмного симулятора є динамічний математичний розрахунок завантаження процесора (CPU) та розміру черг залежно від обраного шляху маршрутизації. Якщо імітована система працює за принципом прямого переспрямування запитів (кеш вимкнено), математичний апарат фіксує перехід у нестаціонарний стан при досягненні порогу у 17 запитів на секунду. В алгоритм розрахунку закладено нелінійний коефіцієнт штрафу за контекстне перемикання, який при переповненні пулу з'єднань підіймає показник утилізації CPU до 85-95%. Час відгуку при цьому деградує відповідно до розміру накопиченої черги.

Активація підсистеми In-Memory кешування докорінно змінює логіку обробки пуассонівського потоку. Застосовано параметр ймовірності влучання в кеш (Cache Hit Ratio - CHR) на рівні 95%. У такому разі алгоритм API Gateway розділяє трафік на два незалежні субпотоки, спрямовуючи абсолютну більшість запитів до надшвидкого вузла Redis. Математичне очікування загального часу відгуку розраховується як середньозважена величина за формулою: New_Latency = (CHR * t_redis) + ((1 – CHR) * t_base). Це рішення усуває фактор тривалих мережевих I/O операцій з основного шляху виконання.

Серія експериментальних стрес-тестів підтвердила адекватність спроєктованої моделі. Під час штатного навантаження (до 10 запитів/с) система демонструвала стабільність із середньою затримкою 50-55 мс та відсутністю черг. При моделюванні раптового сплеску активності (понад 17 запитів/с) без оптимізації була зафіксована стрімка деградація: лічильник черги зростав експоненційно, CPU сягнув 85%, а статус сервера перейшов у стан «КРИТИЧНО». Однак, активація патерну Cache-Aside у режимі реального часу продемонструвала миттєве відновлення. Перенаправлення 95% трафіку на Redis дозволило основній базі оперативно обробити залишки черги. Завдяки ліквідації проблеми контекстного перемикання завантаження процесора впало до безпечного рівня 15-20%, а загальний час відгуку для кінцевого користувача скоротився до мінімальних 1-5 мс.

Висновки. У процесі дослідження було доведено, що пряма інтеграція масштабованих вебсистем із відкритими базами даних є вразливою до стохастичних стрибків трафіку через мережеві затримки та обмежену пропускну здатність зовнішніх API. Розроблена та програмно уможливлена імітаційна модель маршрутизації даних на базі теорії масового обслуговування довела високу дієвість у візуалізації процесів деградації систем під навантаженням. Експериментальні результати симуляції беззаперечно математично підтверджують, що застосування патерну API Gateway у поєднанні з технологією In-Memory кешування (Redis) є критично необхідним архітектурним вибором. Такий підхід кардинально нівелює проблему «пляшкового горлечка», зменшує навантаження на обчислювальні ресурси процесора більш

ніж у чотири рази та забезпечує мілісекундний час відгуку, гарантуючи стійкість сервісу навіть за екстремальних умов функціонування.

Список використаних джерел

1. Таненбаум Е., Бос Х. Сучасні операційні системи. 4-те вид. Київ : Діалектика, 2015. 1120 с.
2. Клеппман М. Розробка інтенсивних даних додатків. Надійні, масштабовані та зручні у підтримці системи. Київ : O'Reilly, 2018. 600 с.
3. Клейнрок Л. Теорія масового обслуговування. Київ : Наукова думка, 1979. 432 с.
4. Ньюмен С. Створення мікросервісів. Київ : Фабула, 2017. 304 с.