УДК 681.12

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПРОЦЕСАМИ НА СЕРВЕРІ

Семеняк Р.В

Барабан М.В.

Анотація

Проаналізовано основні причини зниження продуктивності серверів в умовах високого навантаження. Визначено основні особливості реалізації програмного додатку для аналізу продуктивності серверів.

Ключові слова: *сервер, поток, багатопоточність, продуктивність, трафік*

Abstract

The main reasons for the decrease in the productivity of servers in the minds of high availability have been analyzed. The main features of the implementation of the software add-on for the analysis of the productivity of the servers have been designated.

Keywords: *server, stream, bandwidth, productivity, traffic*

Вступ

Користувачі взаємодіють із системою за допомогою введення якихось даних: дотиків, руху та мови. Всі ці дані обробляються на віддалених комп’ютерах, об’єм даних від багатьох користувачів може занадто відрізнятись, але всі вони у відповідь отримують реакцію, засновану на зоровому, тактильному або слуховому апаратах. Продуктивність – це якість того, як система реагує на дії користувача [1]. Зниження продуктивності може спричинити втрату трафіку, що негативно впливає на бізнес, тому слід розглянути, які саме бувають проблеми.

Результати дослідження

Глобальні проблеми з продуктивністю серверів можна розділити на дві категорії [2]:

- Кількість потоків – завантаження будь-яких ресурсів, необхідних для роботи програми;
- Важкість потоків – робота програми, може бути призупинена для всіх, лише одним користувачем, який відправив занадто великий об’єм даних.

Кожна з цих категорій містить дуже багато нюансів, про які часто не замислюються, але які відрізняють якісні додатки від неякісних.

В той час, коли проблеми великих даних виникають під час апаратних обмежень як зі сторони серверу, так і зі сторони клієнта, проблеми багатопоточного характеру виникають, коли існують помилки під час розробки веб-сервісів. Найбільш вагомою проблемою цього типу – є проблема «вузького місця» [3]. «Вузьке місце» виникає, коли потужність програми або комп’ютерної системи обмежена одним компонентом, наприклад, горловина пляшки, що уповільнює загальний потік води. У випадку з проблемами апаратної частини можливим рішенням буде додання нового сервера або покращення існуючого. А при проблемах з базою даних варто провести оптимізацію запитів [4, 5] та аналіз індексів у ключових таблицях.

Для пошуку таких обмежень проводять стрес тестування системи. Найпопулярнішими рішеннями для проведення стрес тестування є такі [6]:

- JMeter є найпопулярнішим інструментом з відкритим кодом у сфері продуктивності, який допомагає вимірювати час завантаження;
- nGrinder був розроблений, щоб спростити стрес-тестування та забезпечити платформу, яка дозволяє створювати, виконувати та контролювати тести;
- Gatling дозволяє тестувати та вимірювати наскрізну продуктивність додатка та легко розширювати кількість віртуальних користувачів.

Однак у наведених вище найпопулярніших аналогів є ряд недоліків. Головним з них є обмеження в створенні звітів. Особливо це прослідковується у JMeter, у якому звіти реалізовуються через регулярні вирази.

Також більшість популярних аналогів не створюють жодного порівняльного звіту попередніх запусків стрес тестів з останнім. Порівняння зазвичай покладають на користувача, а всі запуски просто лежать у багатьох файлах, що ускладнює пошук того моменту, коли на сервері було створено «вузьке місце». Також у такій архітектурі дуже важко провести аналіз одразу декількох запусків для порівняння результатів.

Тому актуальною задачею є розробка спеціального програмного додатку для підвищення продуктивності серверів в умовах високого навантаження.

Одним з головних модулів для роботи додатку є виконання скрипту. Додаток може використовуватися у великій різноманітності сценаріїв, які зазвичай включають:

1. Запуск спеціальних навантажувальних тестів для окремих API або мікросервісів як частини процесу розробки для вивчення їх характеристик продуктивності та за потреби оптимізації продуктивності (наприклад, запобігання витоку пам’яті або оптимізації коду з великою кількістю процесора).
2. Виконання тестів щодо середовищ постановки/функцій як частини конвеєра CI/CD для раннього виявлення регресій продуктивності.

На рисунку 1 наведено блок-схему алгоритму для виконання користувацького сценарію.

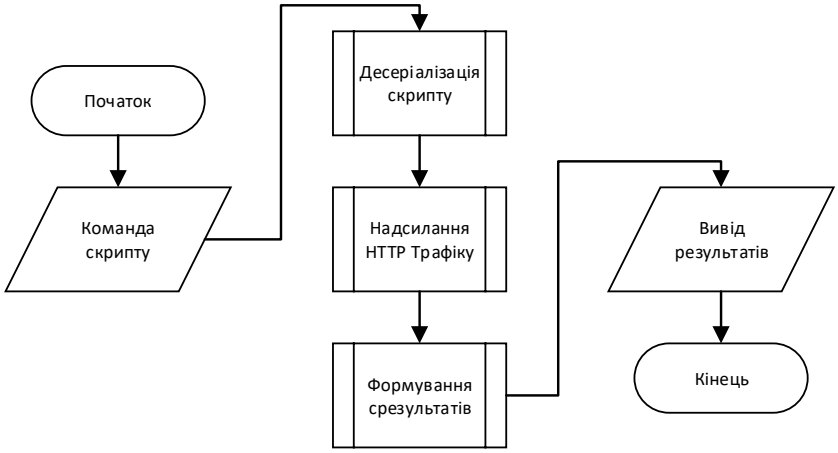

Рисунок 1 – Блок-схема алгоритму виконання користувацького сценарію

Компонент отримує у якості вхідних даних текст скрипту, після чого одразу проводить його десерилізацію. Після чого проводить стрес тестування шляхом симуляції HTTP трафіку. Результатом виконання роботи компоненти є сформований результат стрес тестування, що включає у собі статус код відповіді сервера, а також затрачений на це час у мілісекундах.

Після надсилання всіх запитів, відбувається формування результатів. Для цього використовуються різні агрегатні операції, такі як середнє значення і медіана вибірки часу виконання запиту, а також максимальне та мінімальне значення. Часто ці значення дуже схожі, тому потрібно розглянути детальніше різницю між ними у даній статистиці.

Якщо даних занадто багато, то декілька пікових значень може спотворити результат середнього арифметичного, тому слід використовувати медіану. І навпаки, коли результатів замало, медіана не може дати значну похибку, тому слід формувати обидва значення.

Однією із найважливіших функціональних вимог до додатку є автоматизований аналіз результатів стрес-тестування. Для порівняння результатів з попередніми запусками є необхідним збереження результатів. Існує 2 можливих варіанта збереження даних:

- збереження даних до файлу;
- збереження результатів до бази даних (БД).

Основною перевагою збереження даних до файлу є можливість зберегти дані про кожен запит до веб-сервісу. Якщо за один запуск буде генеруватись забагато запитів, при збереженні даних у БД сховище буде дуже швидко заповнене, тому БД має сенс лише для збереження необхідних результатів агрегатних операцій з даними. У файлах не потрібно зберігати результати агрегатних операцій над даними, оскільки потім можна буде виконати будь-які операції з повним набором даних.

Отже найкращим варіантом для збереження даних є змішаний формат, де повний перелік запитів зберігається у окремому файлі, а результати аналізу зберігаються у БД, для зручного поширення результатів між співробітниками.

Необхідно розробити макет БД. Потрібно забезпечити можливість легко розділяти сценарії виконання та групувати однакові. Для цього варто створити таблицю з різними сценаріями. До неї будуть входити наступні атрибути:

- унікальний ідентифікатор сценарію;
- назва сценарію;
- хеш-сума сценарію – за цим атрибутом можна буде швидко розрізняти різні файли сценарію;
- текст сценарію – крім хеш-суми також буде збережено сам сценарій для можливості легко відрізнити сценарії між собою.

Також необхідно розробити таблицю, у якій будуть збережені результати тестування. До неї будуть входити наступні атрибути:

- унікальний ідентифікатор запуску;
- унікальний ідентифікатор сценарію;
- дата та час проведення тестування;
- результати тестування у форматі JSON, для подальшого оброблення та можливості легко розширити функціонал, додавши нові потрібні параметри.

Кінцеву модель БД зображено на рисунку 2.

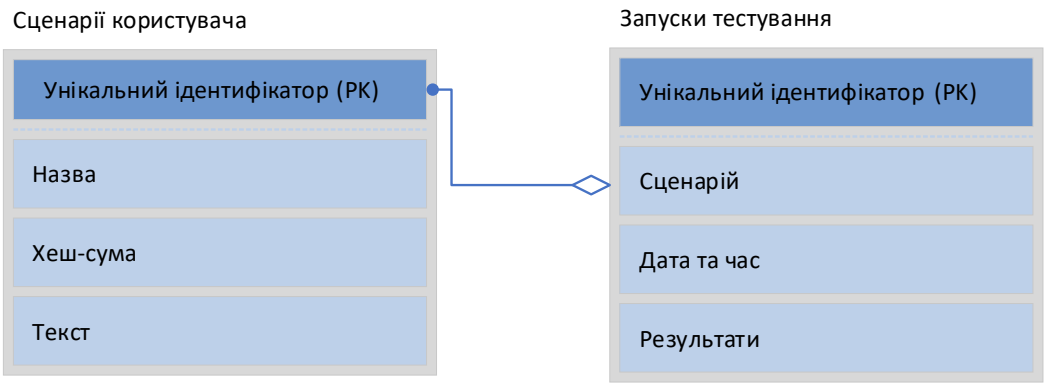

Рисунок 2 – Модель БД для збереження результатів

Висновки

Таким чином, в результаті дослідження було доведено необхідність у розробці спеціального програмного додаток для аналізу продуктивності веб-сервісів під час критичного навантаження, а також визначено основні особливості його реалізації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. MDN Web Docs. Performance fundamentals [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Performance/Fundamentals
2. Что такое производительность веб-приложений? [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/tinkoff/blog/489230/
3. Производительность серверов [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/sql/relational-databases/performance/server-performance-and-activity-monitoring?view=sql-server-ver15/
4. Серверная Оптимизация [Електронний ресурс] – Режим доступу: https://habr.com/ru/hub/server_side_optimization/

5. Оптимизация Сервера сайта [Електронний ресурс] – Режим доступу: https://itfb.com.ua/optimization/
6. 15 Top Load Testing Tools for 2021 (Open Source Guide) [Електронний ресурс] – Режим доступу: https://testguild.com/load-testing-tools/#P2