АРХІТЕКТУРА СИСТЕМИ ВИСОКОНАВАНТАЖЕНИХ WEB-ЗАСТОСУВАНЬ

Санжаровський А.В.
Науковий керівник – д.т.н., проф. Гороховатський В.О.
Харківський національний університет радіоелектроніки
(61166, Харків, просп. Науки,14, каф. Інформатики, тел. (057) 702-14-19)
e-mail: anton.sanzharovskyi@nure.ua.

У наш час є актуальною є розроблення web-застосувань, які розраховані на величезну кількість даних та користувачів. Тобто на web-застосування припадає величезне навантаження (потрібно працювати з мільйонами записів у базі даних та розраховувати систему на тисячі користувачів).

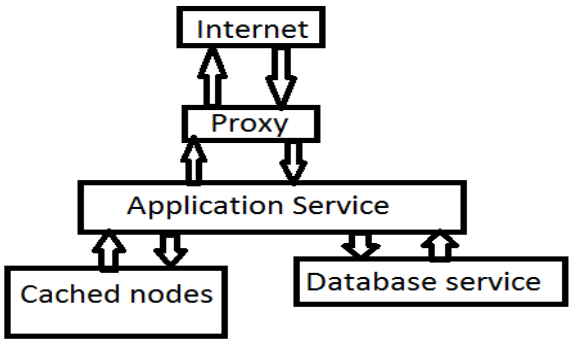

Рис.1. Типова архітектура високонавантаженого web-застосування

Рис. 1 містить архітектуру системи з основними компонентами.

Proxy – використовується для створення власних CDN або при деяких техніках кешування. Однак нечасто використовується на практиці.

Application Service – рівень додатків (серверна частина сайту). Призначений для обробки HTTP-запитів, взаємодії з кешем та БД.

Cached nodes – закешовані вузли (результати запитів до БД, які збережені в кеші). Кешування використовується для того, щоб пришвидшити HTTP - запит. Суть кешування полягає в тому, щоб не здійснювати постійні запити до БД, після першого запиту результат записується на деякий час в кеш. А при здійсненні того ж самого HTTP – запиту, замість запиту до БД ці дані дістаються із кешу, що займає в рази менше часу. Але не варто записувати в кеш дані на нескінченно довгий час, щоб не переповнювати пам’ять сервера. При оновленні або видаленні запису, який збережений у кеші, потрібно видалити відповідний запис із кешу. Найчастіше використовуються такі різновиди кешу: Memcached (на основі оперативної пам'яті сервера); кеш на основі файлової системи; БД Redis у якості кешу.

Database service – рівень баз даних. Призначений для зберігання даних та роботи з ними. (збереження, оновлення, видалення).

Обробка великих масивів даних також є затратною операцією. Це питання переважно вирішується одним із таких способів.

Оптимізація обчислень, пов'язаних з обробкою даних. Наприклад, при отриманні даних для каталогу в онлайн комерційній платформі, треба порахувати рейтинг, кількість замовлень, щоб підкоригувати кількість доступних товарів, відформатувати дати і т.д. Використовуючи звичайні обчислення, цей процес займе багато часу. Тому для скорочення часу обчислення реалізуються процеси оброблення за спеціальними алгоритмами (наприклад швидке сортування). Також код серверної частини пишеться з урахуванням особливостей стеку технологій (наприклад, для PHP не можна робити запити в циклі, виклики функцій треба зводити до мінімуму. А для Node.JS рекомендується робити запити на вибірку та проводити схожі в циклі у зв'язку з асинхронністю, тобто паралельною роботою).

Застосування архітектури клієнт-сервер та виконання обчислень на стороні клієнта. Для цього серверна частина системи реалізується як REST-сервер (обмінюється з клієнтом даними у форматі JSON), а клієнт, як правило, є одно сторінковим додатком (Single Page Application), наприклад Angular, React, Vue JS.

Список використаних джерел:

1. PHP Highload - Блог о разработке высокопроизводительных PHP приложений. [Електронний ресурс]/ http://www.phphighload.com/, 2017 — Режим доступу: http://www.phphighload.com/ - 20.02.2019 р. - Загол. с екрану.

2. Что такое highload. [Електронний ресурс]/ https://ruhighload.com /, 2017 — Режим доступу: https://ruhighload.com/Что+такое+ highload - 20.02.2019 р. - Загол. с екрану

3. High load system and multithreading. [Електронний ресурс]/ https://stackoverflow.com /, 2019 — Режим доступу: https://stackoverflow.com/questions/4651946/high-load-system-and-multithreading - 20.02.2019 р. - Загол. с екрану

4. What is a Highload Project?. [Електронний ресурс]/ http://allyouneedisbackend.com /, 2019 — Режим доступу: http://allyouneedisbackend .com/blog/2017/08/30/what-is-highload - 20.02.2019 р. - Загол. с екрану