УДК 621.391.3

АРХІТЕКТУРНЕ РІШЕННЯ ЩОДО СИСТЕМИ АНАЛІЗУ ДАНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Бугаєнко Ю.М., Ляшенко А.В.
Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, Україна
E-mail: andrey.lyashenko44@gmail.com

ARCHITECTURAL SOLUTION FOR THE SYSTEM OF DATA ANALYSIS OF COMPUTATIONAL PROCESSES

When designing and developing real data analysis systems, a problem arises - high load, due to the large amount of data, on the system. This article explored the effectiveness of a microservice architecture. An architectural solution based on Cloud-computing technologies and micro-services is proposed, which will help to reduce the server load when performing computing processes, in particular, a data analysis system, in which algorithms for cleaning, clustering, building fuzzy logical rules and displaying in the form of a metagraph are performed. The main idea behind this solution was to split the monolithic system into smaller micro-services, each of which is responsible only for its own computational process. As a result, it is possible to use less capacity of each microservice, which will reduce the price, increase the fault tolerance of the system, and these services can be used in the construction of such systems. This architectural solution was tested on the Azure cloud platform as a result of the development of the "Fuzzy Knowledge Base" system for analyzing data on energy efficiency of computing servers. As a result, during the "Performance Testing", a decrease in server load by 37.4% was revealed.

При проектуванні й розробці реальних систем аналізу даних з’являється проблема – високої завантаженості, через великий обсяг даних, на систему. В цій статті проведене дослідження ефективності застосування мікросервісної архітектури. Пропонується архітектурне рішення, яке базується на технологіях Cloud-computing та мікро-сервісів, що допоможе зменшити навантаження сервера під час виконання обчислювальних процесів, зокрема системи аналізу даних, в якій виконуються алгоритми очистки, кластеризації, побудови нечітких логічних правил та відображення у вигляді метаграфа. Головною ідеєю даного рішення було розбиття монолітної системи на менші мікро-сервіси, кожен з яких відповідає тільки за свій обчислювальний процес. В результаті цього можна використовувати меншу потужність кожного мікросервіса, що зменшить ціну, збільшить відмовостійкість системи та ці сервіси можна буде використовувати в побудові схожих систем. Дане архітектурне рішення було опробовано на хмарній платформі Azure, в результаті розробки системи «Fuzzy Knowledge Base», для аналізу даних щодо енергоефективності обчислювальних серверів. В результаті під час «Perfomance Testing», було виявлено зменшення навантаження сервера на 37,4%.

Опис архітектурного рішення. Суть архітектурного рішення полягає в тому, що кожна логічна частина системи повинна бути відокремлена та винесена в окремий мікро-сервіс, який можна легко підключити та інтегрувати до системи, незважаючи на те, яку технологію він використовує в реалізації.

Початкова система мала вигляд моноліту та мала досить багато недоліків, тобто вся усі функції обробки даних було розроблено в одному проекті та цей проект можливо було розгорнути тільки на одному сервері.

Недоліки монолітної системи:

– Систему можна масштабувати тільки вертикально, тобто збільшуючи обчислювальні характеристики сервера (Процесор, операційну пам’ять)
– Існуючий код програмного забезпечення досить важко підтримувати та модифікувати
– Неможливість використовувати окрему частину системи, як одне ціле при інтеграції в інші систему.
– При досить великому навантаженні та під час завантаження файлів, великої розмірності потрібно збільшувати обчислювальні ресурси сервера, що є досить дорого.

Через вище перелічені недоліки при подальшій розробці системи таке архітектурне рішення було не універсальним, тому було вирішено використати мікро-сервісний підхід.

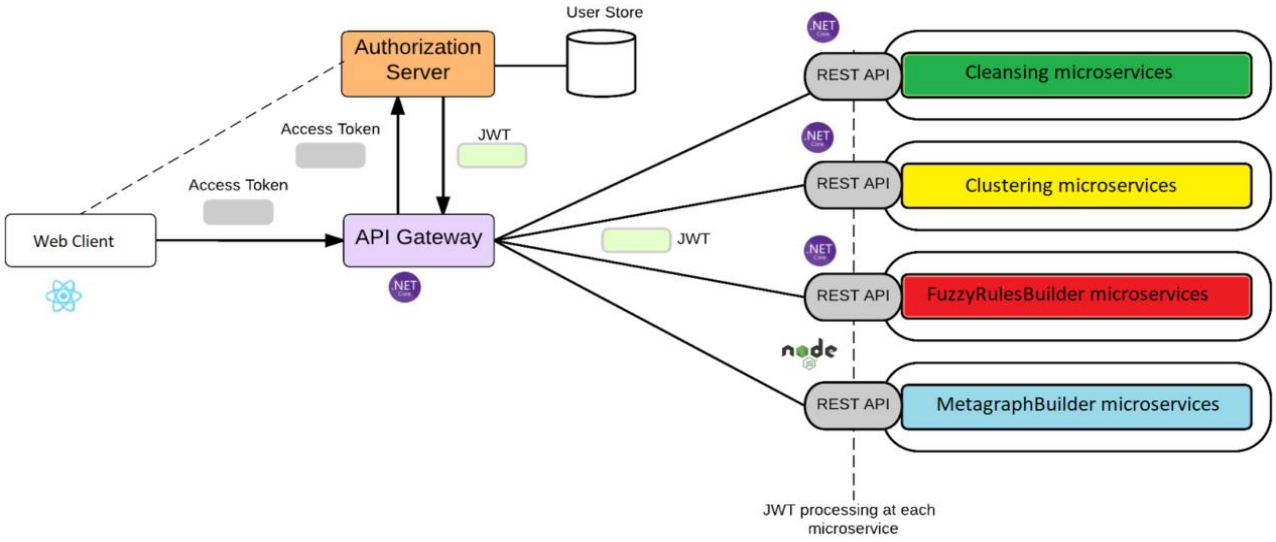

Рис. 1. Архітектура системи на основі мікросервісного підходу.

Кожну частину системи було виділено в окремий проект, який розміщувався на окремому сервері [1]. Система представляє собою веб-сайт, клієнтську частину якого написано на Java-Script із застосуванням бібліотеки React.

Система також має аутентифікацію та авторизацію. Через клієнтську частину під час авторизації клієнт отримує JWT-Token (стандарт токена доступу на основі JSON) від сервера авторизації, за допомогою якого може авторизуватися на кожному із сервісів.

API Gateaway можна назвати головним проектом, яке служить шлюзом між клієнтом там сервісами, виконує агрегацію даних з різних сервісів та відповідає за логіку виконання запитів до сервісів.

Розглянемо наступні сервіси:

Cleansing microservice – сервіс для очищення даних, які завантажує користувач в систему. Сервіс використовує різні алгоритми очистки даних від некоректних даних. Наприклад, в завантаженому файлі можуть бути різноманітні викиди даних або не вірний формат (замість чисел можуть бути слова). Сервіс використовує технологію .Net core та написаний на мові програмування C#.

Clustering microservice – сервіс для кластеризації даних, що надходять після очистки даних. В цьому мікросервісі використовують різні неієрархічні алгоритми кластеризації, для аналізу даних та виділення подібних об'єктів в однорідні групи. Цей крок потрібний для того, щоб побудувати нечіткі логічні правила з даних, що завантажив користувач [2]. Сервіс написаний на мові програмування C#.

FuzzyRulesBuilder microservice – сервіс для побудови нечітких логічних правил із статистичних даних. Використовує в собі метод, який знаходить значення функції Гаусса в точці та відносить цю точку до нечіткого терма лінгвістичної змінної [3], використовує технологію .Net Core.

MetagraphBuilder microservice – сервіс для відображення та візуалізації метаграфа на основі нечітких логічних правил. Цей сервіс відфільтровує дублюючі правила та будує метаграф. Метаграф є вирішенням проблеми прогнозування середньо- та велико-розмірних зв'язків, в якому мета полягає в точному навчанні моделей, які швидко адаптуються до нових даних. Сервіс використовує технологію Node.js.

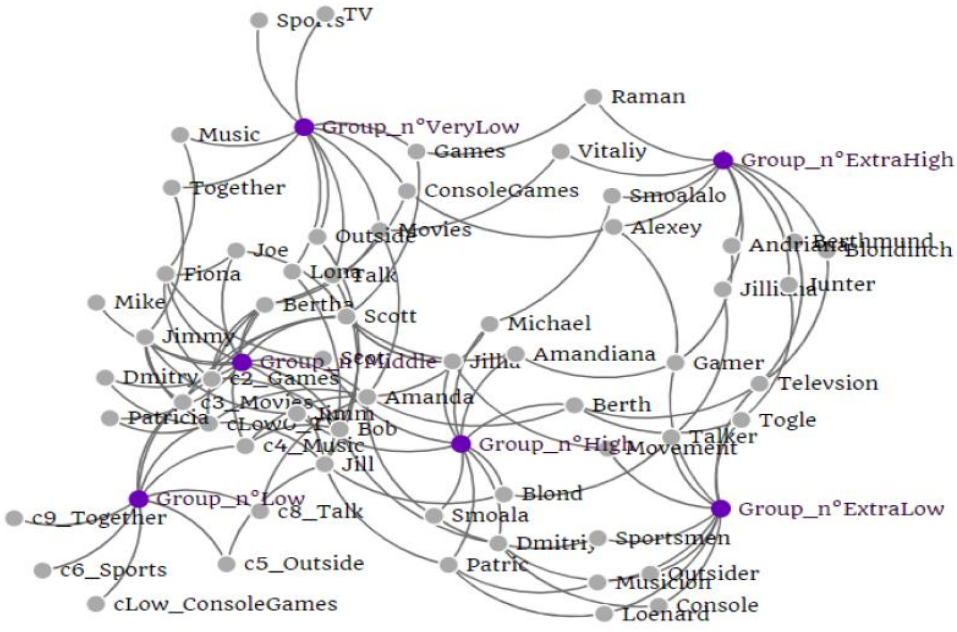

Рис. 2. Побудований метаграф.

Всі сервіси використовують REST – це архітектурне рішення для взаємодії компонентів розподіленої системи в мережі. Взаємодія між сервісами відбувається через HTTP Requests.

Розглянемо результати. Було проведено дослід на однакових даних, з тією самою кількістю користувачів. При мікросервісній архітектурі навантаження менше на 37,4%.

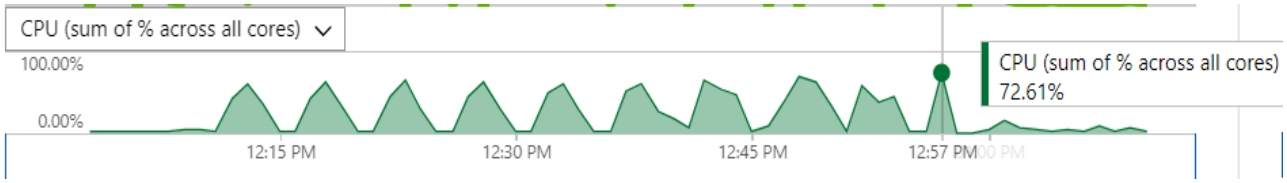

Рис. 3. Навантаження монолітної архітектури.

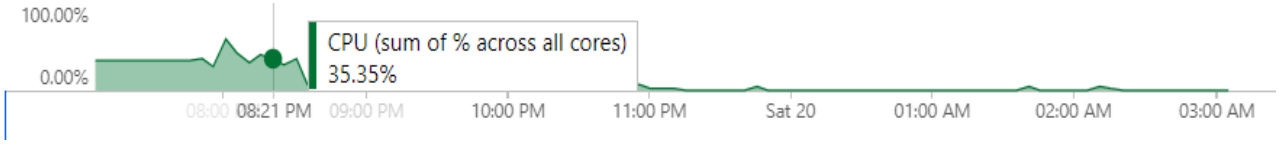

Рис. 4. Навантаження мікросервісної архітектури.

Таким чином, мікросервісна архітектура має перевагу у продуктивності ніж монолітна.

Висновки. В даній статті було запропоноване архітектурне рішення щодо системи аналізу даних обчислювальних процесів, досліджено ефективність мікросервісної архітектури на базі Cloud-computing для його реалізації. Це рішення базується на розбиванні монолітної архітектури на мікросервіси. Кожен сервіс являє собою логічну частину системи, яка може легко інтегруватися з будь-якою системою. Сервіси взаємодіють із головним проектом, який використовується як шлюз, що пересилає запити на відповідний сервіс та агрегує ці дані в одну модель для відображення на клієнтській частині системи, що дозволило зменшити навантаження на систему під час проведення обчислень не менш як на 37,4%.

В майбутньому планується модифікувати дану систему, а саме алгоритми кластеразації та алгоритми очищення даних із статичного набору.

Література

1. https://docs.microsoft.com/ru-ru/dotnet/architecture/microservices/
2. Analysis of clustering algorithms for use in the universal data processing system. Larysa Globa, Yurii Buhaienko, Andrii Liashenko, Maria Grebinichenko National Technical University of Ukraine ”Igor Sikorsky Kyiv Polytechnic Institute” Kyiv.
3. Нечеткое моделирование и управление. А. Пегат с. [506-520].