Інформаційні технології

УДК 004.7

Синкевич Максим Едуардович
студент
Харківського національного університету радіоелектроніки

Лєсна Наталя Совєтівна
кандидат технічних наук,
професор кафедри програмної інженерії
Харківський національний університет радіоелектроніки

ДОСЛІДЖЕННЯ ЗАСОБІВ І ТЕХНОЛОГІЙ МІЖСЕРВІСНОГО ЗВ'ЯЗКУ В МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ

Анотація. *У роботі були розглянуті сучасні засоби міжсервісної комунікації. Основна увага дослідження приділялася питанню продуктивності нової мови запитів GraphQL у порівнянні з REST при використанні JSON у якості формату сериалізації даних.*

Ключові слова: *мікросервіс, зв'язок, продуктивність, REST, GraphQL, JSON.*

Проблема і актуальність дослідження. Актуальність дослідження засобів і технологій міжсервісного зв'язку в мікросервісній архітектурі зумовлена тим, що концепція мікросервісної архітектури полягає у дизайні архітектури програмної системи таким чином, що функціональність розподіляється між сервісами. Кожен із сервісів виконується незалежно і у власному потоці. Тож мікросервіси не повинні мати ніяких залежностей від платформ, інструментів або мов програмування і повинні бути розроблені за допомогою інструментів, які найкраще підходять для

конкретної цілі. Найбільш імовірно, що кожен з сервісів використовується декількома різними клієнтами, серед яких інші мікросервіси системи, що розроблені і працюють на різних платформах. Тому важливо використовувати комунікаційні протоколи та засоби, що можуть ефективно використовуватися як мобільним додатком чи web застосунком, так і компонентами системи.

Огляд поточного стану об'єкта дослідження. У мікросервісній архітектурі сервіси взаємодіють між собою за допомогою передачі повідомлень мережею. Загальноприйнятими методами реалізації такої взаємодії є передача повідомлень за допомогою викликів REST API або моделі підписник-видавець, де сервіси можуть виконувати підписку на канал та отримувати нотифікацію про те, що нове повідомлення було опубліковано до каналу [1]. Проте існує декілька розповсюджених альтернатив, таких як SOAP, gRPC, Thrift, GraphQL та інші. Зв'язок між сервісами багаторазово досліджувався.

Ефективність обміну повідомленнями в розподілених системах є важливою темою, яка висвітлюється у багатьох роботах. Наприклад, Тихоміровс та Гребіс представили дослідження продуктивності веб сервісів при використанні REST та SOAP, у якому REST проявив себе краще [2]. Немає чіткої відповіді на питання, який метод комунікації найкращий для мікросервісів. Хоча фреймворки gRPC і Thrift потенційно пропонують найкращу продуктивність, вони додають залежності до архітектури мікросервісів, змушуючи її розвиватися на певних мовах; список мов, що підтримуються цими фреймворками, обмежений. REST, який є повністю незалежним від платформи, оскільки він є лише архітектурним стилем, може бути реалізований на будь-якій платформі, з будь-якою мовою, використовуючи будь-який транспорт і будь-яку серіалізацію [3]. Проте REST не має структури, яку пропонують фреймворки і протоколи, і часто слід використовувати інструменти третіх

сторін для відстеження його інтерфейсів. SOAP пропонує структурований спосіб відправки повідомлень між службами зі строгими правилами, що визначають структуру повідомлень і способи їх обробки. Основний недолік SOAP полягає в тому, що він прив'язаний до використання XML, який, безумовно, має найгіршу продуктивність із всіх перелічених форматів серіалізації. GraphQL пропонує новий спосіб запиту даних у форматі, який легко зрозуміти. GraphQL пропонує абстрактне уявлення про ресурси даних як вузли у графі, де будь-які вузли можуть бути об'єднані для формування запиту [4]. Якщо фреймворк GraphQL може проаналізувати і серіалізувати дані без великих втрат продуктивності в порівнянні з архітектурою REST з використанням JSON, це може бути життєздатною альтернативою.

Мета дослідження. Дана робота містить дослідження засобів міжсервісної комунікації для визначення ефективного способу міжсервісної взаємодії. Об'єктами дослідження обрано фреймворк GraphQL та архітектурний стиль REST. У якості формату сериализаціх обрано JSON. Метою є створення прототипу мікросервісної системи, призначеної для порівняльного аналізу продуктивності міжсервісної комунікації при використанні REST та GraphQL.

Дослідження продуктивності міжсервісної комунікації при використанні REST та GraphQL. Критерієм продуктивності міжсервісної комунікації є час, який витрачається на транспортування повідомлення за обраним протоколом та технологією. Для порівняння продуктивності GraphQL та REST розроблено тестове середовище та виконано заміри часу проходження запитів через систему.

Тестове середовище містить чотири мікросервіси та сервер баз даних. Кожен з сервісів має підтримку REST та GraphQL запитів. Мікросервіси в системі призначені для виконувати тільки доступ до бази даних і не виконувати жодних фактичних обчислень. Кожний сервіс має

доступ до даних, що доступні тільки для цього конкретного сервісу. Розглянемо тест, що складається з отримання досить невеликої кількості даних від кожного сервісу з використання інтерфейсів REST і GraphQL. Таблиці баз даних містять по 5 записів. Для отримання достовірних результатів надіслано 100 запитів один за одним з секундною затримкою між ними. Затримка впроваджується для того, щоб бути впевненим, що навантаження на систему має мінімальний вплив на результат. За час відповіді взято повний час знаходження запиту у системі, включаючи час, який займає виконання обробки запиту мікросервісом та включаючи час на витяг та обробку даних з бази даних. Час відповіді для кожного запиту до кожного із інтерфейсів зображено на рисунку 1.

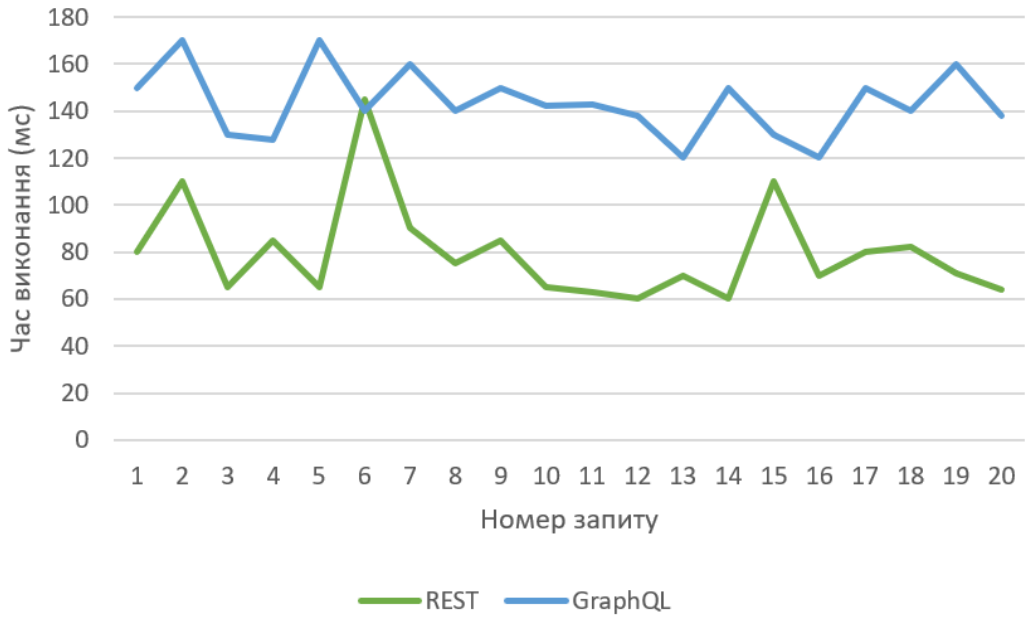

Рис. 1. Час відповіді при запиті даних за допомого GraphQL та REST інтерфейсів

Найшвидший час відповіді для REST становить 58,53 мс, а найповільніший — 151,08 мс. Взаємодія з використанням GraphQL має значно гірший результат з найшвидшим часом відгуку у 119,86 мс і найповільнішим часом відгуку у 177,72 мс. Середній час відгуку для REST дорівнює 78,94 мс, а для GraphQL — 142,92 мс. Таким чином GraphQL є повільнішим в середньому на 81,05%.

Аналогічні тести проведені з використанням більших об'ємів даних: по 25 та 50 запитів у таблицях баз даних. У всіх випадках GraphQL показав довший час відповіді ніж REST.

Висновки. Результати тестів показують, що GraphQL не може конкурувати з архітектурою REST з використанням JSON формату з точки зору часу відгуку. Навіть при отриманні незначного обсягу даних обробка GraphQL викликає просідання часу відповіді. Так як GraphQL прив'язаний до використання JSON у якості формату повідомлень, то він обмежений у техніках серіалізації. Проблема незручності REST інтерфейсу полягає у тому, що кожен ресурс відображається на URI і якщо різні програми-споживачі мають потребу у специфічній структурі даних, то для кожного такого випадку необхідно створювати новий специфічний REST ендпоінт. У противагу, при використанні GraphQL потрібно створити тільки докладну структуру, що описує наявні типи і поля. Використовуючи такий ресурс кожен клієнт-споживач може точно вказати, які поля він бажає отримати. Це виключає великі інтерфейси, які важко підтримувати і які не мають чітких правил щодо того, як обробляти декілька версій одного API. GraphQL є гарною ідеєю для використання у сценарії, коли запитується невеликий обсяг даних. Але якщо продуктивність є важливим критерієм при виборі технології, то GraphQL не є сильним претендентом.

Література

1. C. Richardson. Building microservices: Inter-process communication in a microservices architecture [Електронний ресурс]. — Режим доступу: https://www.nginx.com/blog/building-microservices-inter-process-communication (дата звернення: 23.05.2017).
2. J. Grabis J. Tihomirovs. Comparison of soap and rest based web services using software evaluation metrics. Information Technology and Management Science, 19, December 2016 — 92 c.

3. S. Newman. Building Microservices. O'Reilly Media Inc., USA, 2015. — P. 49.
4. Graphql offcial webpage with documentation [Електронний ресурс]. — Режим доступу: http://graphql.org (дата звернення: 23.05.2017).