УДК 004.4

ОЦІНКА ВПЛИВУ ВИКОРИСТАННЯ ФРЕЙМВОРКУ gRPC НА ПРОДУКТИВНІСТЬ СЕРВІСУ У ПОРІВНЯННІ З REST API

Лисяков М.С.
Науковий керівник – доц., к.т.н. Ворочек О.Г.
Харківський національний університет радіоелектроніки,
каф. Програмної інженерії, м. Харків, Україна
тел. +38(098)57-25-995

The increasing use of cloud computing has led to the widespread adoption of microservices as an architectural pattern. This approach involves breaking down large applications into smaller, independent packages that can be distributed across different environments, with the goal of achieving a cloud-native architecture. However, selecting the most effective communication mechanism between these services can be a complex task. To address this challenge, this thesis presents experimental findings comparing common inter-process communication methods, with a particular focus on REST APIs and Google gRPC.

Стиль мікросервісної архітектури став одним з найпопулярніших архітектурних стилів розробки програмного забезпечення в останні роки. Згідно з дослідженням, проведеним компанією Statista [1], 81,5% компаній вже використовують мікросервісну архітектуру, а 17,5% компаній планують перейти на цей тип архітектури. Лише 1% респондентів задоволені монолітною архітектурою. Згідно з іншими дослідженнями проведеними компанією RedHat [2], 73 % компаній вже використовують мікросервісну архітектуру.

Оскільки системи на основі мікросервісів є розподіленими, однією з ключових проблем при розробці програми є вибір механізму, за допомогою якого сервіси взаємодіють один з одним. Існує декілька підходів для реалізації між процесної взаємодії (англ. inter-process communication, IPC) в мікросервісах, і кожен з них має свої переваги та компроміси.

На сьогодні в галузі вважається, що архітектура REST з використанням текстового протоколу JSON є стандартом для взаємодії в мікросервісній архітектурі. Відносно нова технологія gRPC починає заміщувати REST архітектуру, особливо в великих компаніях, таких як Google, Netflix, Spotify та інші.

В даному дослідженні порівнюється продуктивність двох вебінтерфейсів опираючись на наступні показники:

- використання процесорного часу у відсотках;
- час виконання запиту що включає процес (де)серіалізація даних на клієнті та сервері та час обміну повідомлення мережею;
- пропускна здатність що характеризує скільки веб запитів може обробити система за певний проміжок часу.

Для виконання замірів було розроблено програмну систему що складається з вебсервісу, вебклієнт та допоміжного модуля з підготовки статичних даних мовою програмування Java [3].

Для заміру показників було проведено навантажувальне тестування. Для кожного тесту використовувалось 100 потоків виконання, кожен потік викликав метод сервера 100 разів. Кожен тест проводився 11 разів та за допомогою медіани обирались показники для порівняння. Варто зазначити що абсолютні показники отримані під час дослідження є лише прикладом роботи системи та можуть відрізнятися під час повторного виконання. Результати наведені у таблиці 1.

Таблиця 1 – Результати порівняння продуктивності фреймворків

Розмір повідомлення	Малий		Середній		Великий	
	REST	gRPC	REST	gRPC	REST	gRPC
Кількість запитів за секунду	131521	90334	8842	44853	1029	6017
Швидкість виконання запиту, мс	0,4652	0,6642	6,7853	1,3377	58,2989	9,9702
Відсоток використання процесорного часу	27,4	10,8	28,2	18,8	31,3	27,2

Згідно з таблицею 1 можна зробити висновки що фреймворк REST є більш продуктивним при малому розміру повідомлень. Пропускна здатність REST на 45 відсотків більше ніж у gRPC. Але слід зазначити що REST використовує на 16,6 відсотка більше процесорного часу. При збільшенні розміру повідомлень фреймворк gRPC показує кращі результати. Пропускна здатність зростає на 407 та 484 відсотки при середньому та великому розмірі повідомлень та потребує на 9,4 та 4,1 відсотка менше процесорного часу для обробки повідомлень.

Список використаних джерел:

1. Microservices use in organizations worldwide 2021 | Statista. (б. д.). Statista. https://www.statista.com/statistics/1236823/microservices-usage-per-organization-size/

2. 6 overlooked facts of microservices. (б. д.). RedHat. https://www.redhat.com/architect/microservices-overlooked-facts

3. Mykyta, L. (б. д.). lysiakov/rest-grpc. GitHub. https://github.com/lysiakov/rest-grpc