УДК 004.7

А.В. Миргородський[1]
О.В. Романюк[1]

ВИКОРИСТАННЯ ФРЕЙМВОРКУ GRPC ДЛЯ РОЗРОБКИ ГНУЧКИХ КЛІЄНТ-СЕРВЕРНИХ ДОДАТКІВ

[1]Вінницький національний технічний університет

Анотація

Наведено короткі відомості про сучасні тенденції в розвитку технологій для взаємодії між сервісами, описано переваги фреймворку gRPC, способи використання та можливості щодо захисту й оптимізації швидкості запитів. Наведено приклад архітектури з використання фреймворку.

Ключові слова: IT, RPC, gRPC, HTTP/2, Protocol Buffers, мікросервіси, мікросервісна архітектура.

Abstract

Given brief information about current trends in technology development for inter-service interaction, described the advantages of the gRPC framework, ways to use and opportunities to protect and optimize the speed of requests. An example architecture for using the framework is given.

Keywords: IT, RPC, gRPC, HTTP/2, Protocol Buffers, microservices, microservice architecture.

Вступ

Методи розробки й підтримки сучасних інформаційних систем значно змінились за останні десятиріччя й продовжують стрімко розвиватися. Класична архітектура з одним сервером для обслуговування запитів та декількома додатковими машинами для розгортання бази даних і додаткових сервісів вже поступово відходить у минуле й залишається актуальною лише для невеликих систем з низьким навантаженням. Сучасний бізнес надає перевагу мікросервісній архітектурі [1], а підтримка принципів High Availability та Horizontal Scaling є важливою характеристикою сучасних сервісів.

Довгий час стандартним рішенням для комунікації між ПЗ та виклику Web-API була архітектура REST, яка витіснила протокол SOAP та інші альтернативи, що базувалися на обміні даних через XML [2]. Тепер ситуація знову змінюється, адже підхід до створення розподілених серверних систем та використання мікросервісів для побудови комплексної архітектури відроджують популярність RPC протоколів. Прикладом цього є поширення фреймворку gRPC, який базується на сучасному технологічному стеку та значно спрощує міжсервісну взаємодію.

Метою дослідження є вивчення переваг та особливостей фреймворку gRPC при розробці гнучких клієнт-серверних додатків.

Об'єктом дослідження є процес розробки клієнт-серверних додатків з використанням gRPC.

Предмет дослідження – фреймворк системи віддаленого виклику процедур gRPC.

Основна частина

Техніка виклику віддалених процедур (RPC або Remote Procedure Call) відома ще з минулого сторіччя і раніше набагато частіше використовувалась для взаємодії між різними сервісами або ж окремими серверними машинами. Першочергова концепція RPC полягала в тому, що виклик віддаленої процедури для розробника поверхнево нічим не відрізнявся від виклику локальної процедури створюваної програми. Ця додаткова абстракція дозволяла спростити логіку взаємодії, але подальше ускладнення програмного забезпечення та необхідність розробляти комплексну архітектуру для нього змусила відмовитись від RPC і перейти до концепції веб-сервісів. В цьому випадку розробник знав про клієнт-серверну взаємодію та свідомо розробляв ПЗ у виглядів сервісів або клієнтів для них. Основними функціями таких додатків було створення та обслуговування запитів, які вже не приховувалися за іншими абстракціями.

Сучасні веб-сервіси найчастіше використовують архітектуру REST, яка розвивала попередні думки й ставила за мету спрощення розробки й використання, а також стандартизацію частини технологій. В основі REST лежить протокол HTTP, а найпопулярнішим форматом для серіалізації і передачі даних став JSON. Операції (або ж запити) обмежені існуючими HTTP-методами. Введення додаткових обмежень і використання поширеного технологічного стеку дозволяє сьогодні використовувати REST-API практично з будь-яких пристроїв, які здатні підключатися до мережі.

Фреймворк gRPC першочергово розроблено компанією Google, але зараз це продукт з відкритим кодом, який швидко розвивається і відновлює популярність підходу до виклику віддалених процедур. Фреймворк не є повноцінною заміною REST і не ставить це за мету, але добре підходить для певних сфер використання: наприклад, розробки високонавантажених додатків або ж створення компонентів для мікросервісної архітектури.

Розглянемо приклад розробленої за допомогою gRPC системи: мобільний пристрій (додаток для перегляду новин) з реалізованим на мові програмування Kotlin клієнтом, який спочатку робить запит на сервіс авторизації, після чого починає завантажувати новини в потоковому режимі. Схема взаємодії для такої архітектури показана на рисунку 1.

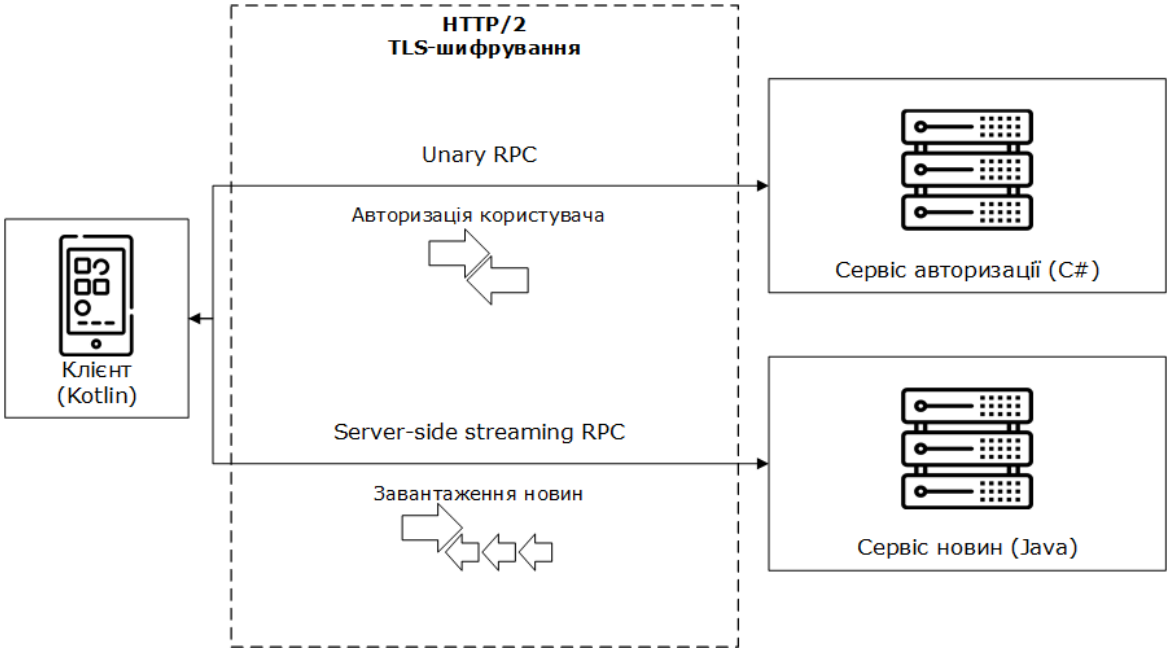

Рисунок 1 – Архітектура клієнт-серверної системи з використанням gRPC

Для кращого розуміння наведеної архітектури необхідно ознайомитися з основними можливостями й перевагами фреймворку. Для виклику процедур через gRPC використовується оновлений протокол HTTP/2, а більш старі версії не підтримуються. Це лежить в основі швидкодії фреймворку: HTTP/2 побудовано в якості бінарного, а не текстового протоколу, він реалізує більш ефективні моделі створення і підтримки з'єднання з сервером, а також підтримує стиснення заголовків та даних. REST-сервіси також можуть використовувати нову версію протоколу, але прив'язка до старої архітектури, що орієнтована на HTTP/1.1, не дає можливості використовувати всі переваги другої версії. Лише близько половини загальнодоступних веб-ресурсів та API використовують HTTP/2 [3].

Використання класичних REST-API вимагає додаткової роботи по створенню запитів до серверу, а також серіалізації та десеріалізації даних в форматі JSON. Крім того, реалізація обробки помилок також залежить від кінцевого розробника. Фреймворк gRPC надає можливість генерації початкового коду для клієнта та сервера, а за замовчуванням використовує формат Protocol Buffers, який зменшує обсяги даних, що передаються по мережі. Крім того, Protocol Buffers пропонує зручний спосіб опису API в proto файлах, а також надає функціонал оновлення формату повідомлень для підтримки зворотної сумісності сервісів. Генерація коду можлива більше ніж для 10 мов програмування і дозволяє поєднувати сервіси та клієнти, що побудовані на різних технологічних стеках.

Використання протоколу HTTP/2 дозволяє реалізувати різні типи RPC:

- unary (simple) – звичайний запит до сервера з очікуванням відповіді;

• server-side streaming – сервер при отриманні запиту від клієнта відповідає довільною кількістю повідомлень (наприклад, надсилає частини інформації під час виконання довготривалої задачі);

• client-side streaming – клієнт надсилає не один запит до сервера, а довільний набір повідомлень, при цьому сервер сам вирішує яку кількість повідомлень приймати й може надіслати лише одну відповідь (підходить для випадків, коли серверні ресурси обмежені й немає необхідності відповідати на кожен запит клієнта);

• bidirectional streaming – комбінація двох попередніх типів RPC, клієнт може надіслати довільну кількість запитів, а сервер – довільну кількість відповідей, при цьому немає вимог щодо порядку обміну повідомленнями.

Введення додаткових просунутих типів RPC допомагає досягти ще більшої ефективності при обміні даних, адже можливість потокового обміну даними значно економить ресурси й час на встановлення з'єднання та дозволяє перевикористовувати існуюче TCP-підключення до серверу.

Важливим аспектом при комунікації між сервісами є шифрування даних та авторизація клієнтів. Фреймворк gRPC надає декілька інструментів для вирішення цих задач. По-перше, наявна підтримка TLS-шифрування, а його ввімкнення зводиться до генерації відповідних сертифікатів та модифікації коду запуску сервера. Для авторизації клієнтів є підтримка mTLS (mutual TLS authentication) за допомогою вже існуючих сертифікатів або ж можливо реалізувати власну систему перевірки клієнтів [4]. Для цього є функціонал передачі додаткових метаданих з усіма створюваними запитами в RPC, які сервер може обробляти відповідно до власної логіки.

Висновок

Дослідження довело, що фреймворк gRPC може спростити взаємодію між створюваними сервісами, допомогти оптимізувати API для їх використання та спростити реалізацію захисних механізмів. Даний фреймворк не може повністю замінити популярний підхід до розробки REST-API, але є надзвичайно корисним при розробці мікросервісної архітектури, сервісів з вимогами до швидкості та зручності інтеграції.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. O'Reilly's Microservices Adoption in 2020 Report Finds that 92% of Organizations are Experiencing Success with Microservices [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.oreilly.com/pub/pr/3307.
2. Levin G. The Rise of REST API [Електронний ресурс] / Guy Levin. – 2015. – Режим доступу до ресурсу: https://blog.restcase.com/the-rise-of-rest-api/.
3. Usage statistics of HTTP/2 for websites [Електронний ресурс] – Режим доступу до ресурсу: https://w3techs.com/technologies/details/ce-http2.
4. Authentication. An overview of gRPC authentication, including built-in auth mechanisms, and how to plug in your own authentication systems. [Електронний ресурс] – Режим доступу до ресурсу: https://grpc.io/docs/guides/auth/.

Миргородський Андрій Вікторович – студент групи 3ПІ-18б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: mirgorodskijav@gmail.com

Романюк Оксана Володимирівна – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: romaniukoksanav@gmail.com

***Myrhorodskyi Andrii* –** student of group 3PI-18b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: mirgorodskijav@gmail.com

Oksana Romaniuk – Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com