International scientific journal «Grail of Science» | № 53 (June, 2025)
ISSN 2710–3056

ARTICLE

DOI 10.36074/grail-of-science.20.06.2025.074

ПРОПОЗИЦІЇ ЩОДО ОПТИМІЗАЦІЇ РОБОТИ САЙДКАРУ В УМОВАХ ВИСОКИХ НАВАНТАЖЕНЬ

НАУКОВО-ДОСЛІДНА ГРУПА:

Коломійцев Олексій Володимирович
д-р техн. наук, професор, Заслужений винахідник України,
професор кафедри
Національний технічний університет «Харківський політехнічний інститут», Україна

Бульба Сергій Сергійович
канд. техн. наук, доцент,
доцент кафедри
Національний технічний університет «Харківський політехнічний інститут», Україна

Носко Сергій Вікторович
аспірант кафедри
Національний технічний університет «Харківський політехнічний інститут», Україна

Скородєлов Володимир Васильович
канд. техн. наук, доцент,
професор кафедри
Національний технічний університет «Харківський політехнічний інститут», Україна

Рєзніков Юрій Вячеславович
канд. техн. наук, ст. наук. сп.
провідний науковий співробітник
Державний науково-дослідний інститут випробувань і сертифікації озброєння та військової технік, Україна

Собора Анатолій Іванович
канд. техн. наук, ст. дослідник
провідний науковий співробітник
Державний науково-дослідний інститут випробувань і сертифікації озброєння та військової технік, Україна

Гейко Геннадій Вікторович
канд. техн. наук,
доцент кафедри
Національний технічний університет «Харківський політехнічний інститут», Україна

Лільчицький Вадим Ігорович
начальник науково-дослідного відділу
Державний науково-дослідний інститут випробувань і сертифікації озброєння та військової технік, Україна

Панченко Володимир Іванович
старший викладач кафедри
Національний технічний університет «Харківський політехнічний інститут», Україна

Олійник Наталія Олександрівна
науковий співробітник науково-дослідного відділу
Державний науково-дослідний інститут випробувань і сертифікації озброєння та військової технік, Україна

Рєзнікова Ольга Геннадіївна
доктор філософії,
старший науковий співробітник
Державний науково-дослідний інститут випробувань і сертифікації озброєння та військової технік, Україна

Анотація. *В статті запропоновано пропозиції щодо оптимізації роботи сайдкару в умовах високих навантажень. Розкрито роботу сайдкару та його основні функції. Доведено, що фреймворк Quarkus має високу продуктивність та гнучкість, що дозволяє розробляти власну бізнес-логіку. Показано розгортання мікросервісу із сайдкаром. Високий рівень безпеки у межах мікросервісної архітектури забезпечує авторизація. Розкрито особливості використання сайдкаром додаткових ресурсів. Наведено графічний мартеріал.*

Ключові слова: *оптимізація, сайдкар, високе навантаження, фреймворк, мікросервіс, патерн, под, платформа, код, запит, застосунок, система, програма, авторизація запитів.*

Вступ. Сайдкар без проведення відповідної оптимізації може споживати багато системних ресурсів, що ускладнює масштабованість та стабільну роботу архітектури. Надмірне використання оперативної пам'яті через витоки пам'яті може знизити продуктивність системи. Методи буферизації, які часто використовуються у сайдкар-компонентах для передавання потоків даних, також можуть призвести до перевантаження ресурсів через незбалансовані швидкості між вхідними та вихідними потоками даних [1-3].

Таким чином, розробка пропозиції щодо оптимізації роботи сайдкару в умовах високих навантажень є актуальним науковим завданням.

Основна частина. Відомо, що сайдкар виступає у якості архітектурного патерну, який передбачає розгортання додаткового компоненту поруч із основним мікросервісом, що працює як проксі між основним мікросервісом та зовнішнім середовищем, доповнюючи його функціональність без модифікації коду самого мікросервісу.

До основних функції сайдкару можливо віднести наступні: передача HTTP запитів між сервісами, авторизація запитів, логування, кешування, забезпечення рівномірного розподілу запитів тощо [1-7].

Найбільш затребуваними є сайдкари на основі: NGINX – як рішення для марштуритазії та кешування, Envoy – як високопродуктивнмй проксі-сервер, який використовується у сервісних mesh-архітектурах, а також рішення, які засновані на фреймворках: Spring Boot, Micronaut, Quarkus тощо.

Сайдкар, який працює на базі фреймворка Quarkus у режимі native із використанням GraalVM для випереджувальної компіляції (Ahead-of-Time, AOT) дозволяє перетворювати код безпосередньо у нативний машинний формат. За таким підхідом забезпечується швидкий запуск програми – за кілька мілісекунд, що дорівнює використанню низькорівневої мови програмування C.

На рис. 1 наведено приклад порівняння витрату часу, який необхідний на запуск застосунку між різними платформами.

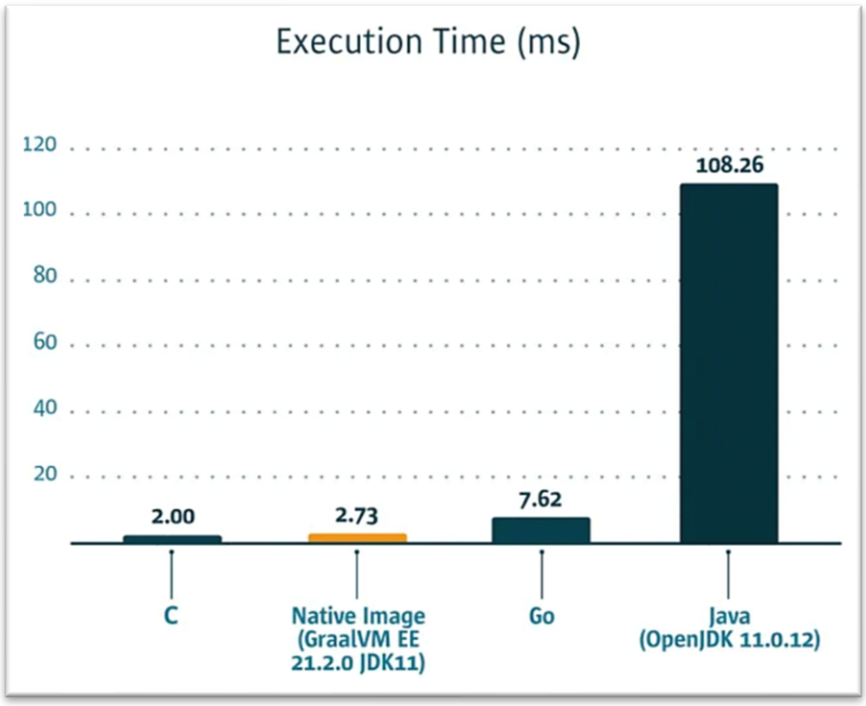

Рис. 1. Порівняння витрату часу, який необхідний на запуск застосунку між різними платформами

Фреймворк Quarkus має високу продуктивність та гнучкість, оскільки, на відміну від стандартизованих рішень таких, як NGINX або Envoy, він дозволяє розробляти власну бізнес-логіку. На приклад, для авторизації запитів інтелектуального кешування (адаптивних алгоритмів буферизації даних) дозволяє створювати ефективні та кастомізовані рішення.

Розгортання мікросервісу із сайдкаром

Kubernates є платформою для оркестрації контейнерів, що забезпечує автоматизоване розгортання, масштабування та управління контейнеризованими застосунками. Отже, Kubernates є базовим будівельним блоком, на якому побудована уся платформа. Поди можуть містити від одного до декілька контейнерів, які пов'язані між собою та виконують спільну задачу. При цьому, контейнер мікросервіса та контейнер сайдкара знаходяться у одному й тому ж самому поді.

На рис. 2 наведено схему розгортання пода з сайдкаром, а на рис. 3 – приклад реального застосування.

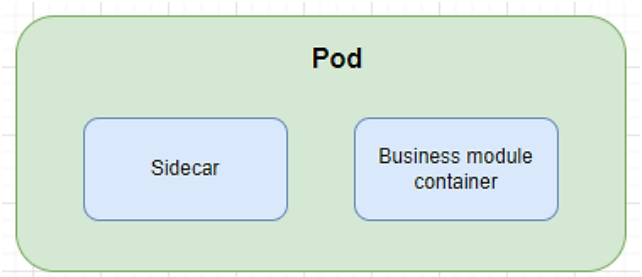

Рис. 2 Схема розгортання пода з сайдкаром

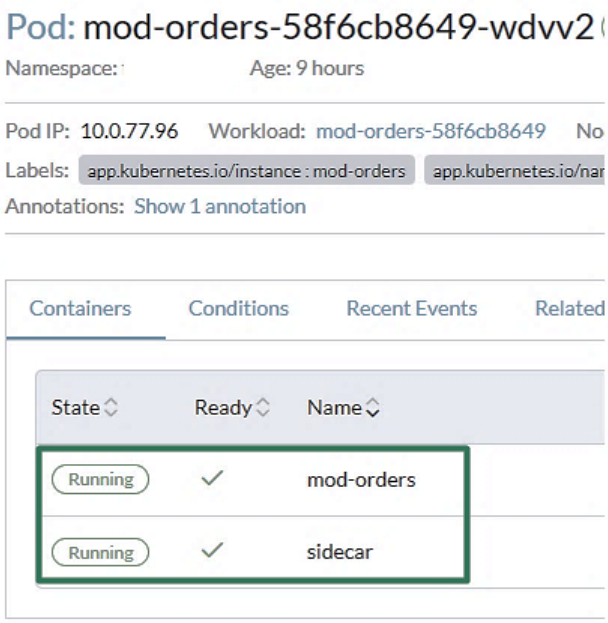

Рис. 3. Контейнер мікросервіса з сайдакром в одному поді

До основних переваг такого рішення можливо віднести наступні:

– спільне управління життєвим циклом. Поди у Kubernates є атомарними одиницями на рівні оркестрації. Управління життєвим циклом пода автоматично впливає на усі контейнери всередині нього. У такому випадку, коли необхідно буде перезапустити контейнер мікросервіса, то відповідний сайдкар буде автоматично перезапущений та правильно оброблений;

– спільне використання ресурсів. Оскільки мікросервіс та сайдкар розміщені у єдиному поді, то вони використовують спільний мережевий простір імен та призначаються до однієї і тієї ж IP-адреси, що спрощує міжконтейнерне спілкування;

– єдність конфігурації та секретів. Контейнери у одному поді можуть спільно використовувати конфігурації та секрети, що спрощує управління налаштуваннями і безпекою;

– оптимізація використання ресурсів. Коли контейнери у поді поділяють загальні ресурси такі, як CPU та оперативна пам'ять, то вони можуть більш ефективно користуватися ними. Kubernetes здатен керувати виділенням ресурсів на рівні пода, що може зменшити загальну кількість вимог до ресурсів порівняно зі сценарієм, де кожен контейнер виконується у окремому поді.

Сайдкари відіграють головну роль у структурі архітектури мікросервісів та забезпечують взаємодію між компонентами. Кожен сайдкар є посередником для свого мікросервісу, розташованого у одному й тому ж поді. Сайдкар ініціює зв'язок зі своїм мікросервісом. У випадку потреби у взаємодії з іншим мікросервісом, запит спочатку обробляється власним сайдкаром поточного мікросервісу. Далі, після первинної обробки, запит пересилається до сайдкара цільового мікросервісу.

На рис. 4 наведено приклад приведеної взаємодії, яка показує принципи комунікації та маршрутизації запитів у межах архітектури мікросервісної системи.

За результатами аналізу рис. 4 можливо стверджувати те, що сайдкар інтегровано із сервером авторизації та забезпечує інкапсуляцію усієї логіки авторизації HTTP-запитів безпосередньо у коді сайдкара. За таким підходом можливо уникнути необхідність дублювання авторизаційної логіки на рівні кожного окремого мікросервісу [3-7].

Отже, важливість авторизації та забезпечення безпеки у межах мікросервісної архітектури важко переоцінити. Центральне розташування такої логіки у одному компоненті, яким є сайдкар, значно знижує ймовірність помилок під час її реалізації, що дозволяє підвищити загальну надійність системи.

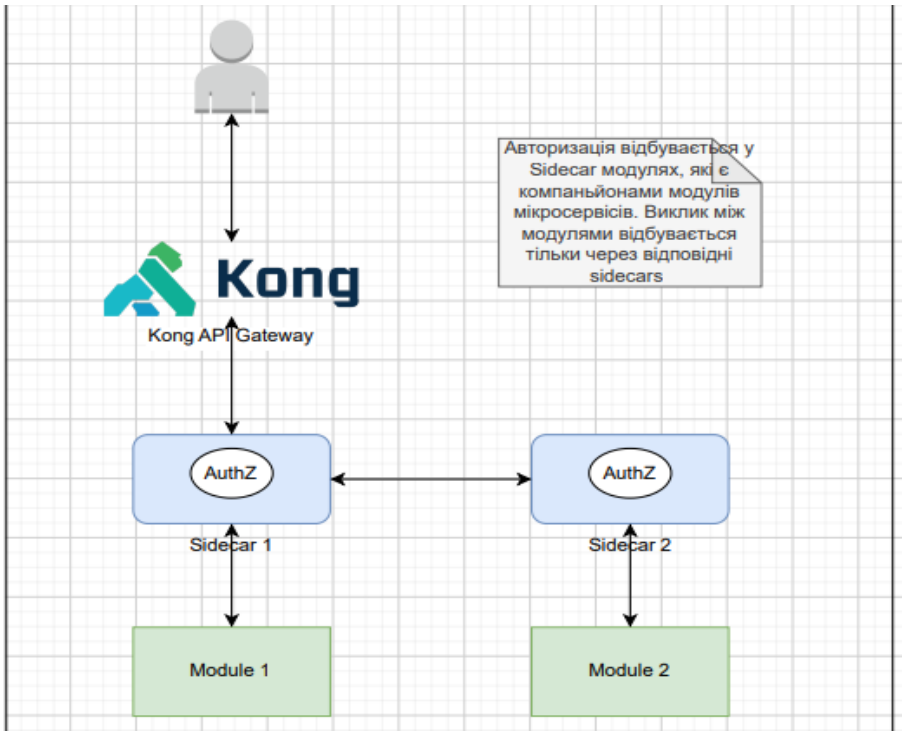

Рис. 4. Комунікація між мікросервісами, використовуючи сайдкари

Розробка авторизаційної логіки потребує високого рівня технічної експертизи. Розміщення такої логіки у межах сайдкара забезпечить коректну та перевірену реалізацію, яку можуть використовувати усі мікросервіси. За таким підходом підвищиться рівень безпеки системи, оскільки усувається ризик до помилок, які пов'язані із окремими неузгодженостями у реалізації авторизації у різних мікросервісах.

Отже, авторизація та аутентифікація є одним із важливих процесів, де сайдкар знаходить своє ефективне застосування.

До основних переваг такого рішення можливо віднести наступні:

– реалізація аутентифікації та авторизації у сайдкарі дозволяє управляти безпекою централізовано для усієї системи та усуває необхідність реалізації цих функцій у кожному мікросервісі окремо;

– використання сайдкару забезпечує однорідне застосування контролю доступу та політик безпеки для усіх мікросервісів, що усуває необхідність у додатковій конфігурації окремих мікросервісів, сприяє як зменшенню кількості похибок, так і полегшенню підтримці;

– масштабованість мікросервісів проходить простіше, оскільки усі налаштування вже інтегровані у сайдкар компонентах, а не у мікросервісах.

Авторизація у розподілених системах, особливо у архітектурах мікросервісів, є однією з найбільш складних та критичних задач, яка вимагає не тільки технічного, але й експертного дослідження у сфері керування безпекою. Складність полягає у забезпеченні надійної та безпечної авторизації без негативного впливу на продуктивність системи. Вибір неправильного рішення може призвести до вразливостей у безпеці та зниження ефективності системи.

Таким чином, уся логіка роботи сайдкара залишається незмінною незалежно від конкретного мікросервісу, з яким він інтегрований. Така архітектура є концепцією у об'єктно-орієнтованому програмуванні, де клас у програмній мові, на приклад Java, можливо порівняти із сайдкаром, а об'єкт, що створений на основі даного класу – з подом, який включає конкретний екземпляр сайдкара.

На рис. 5 наведено приклад розгортання.

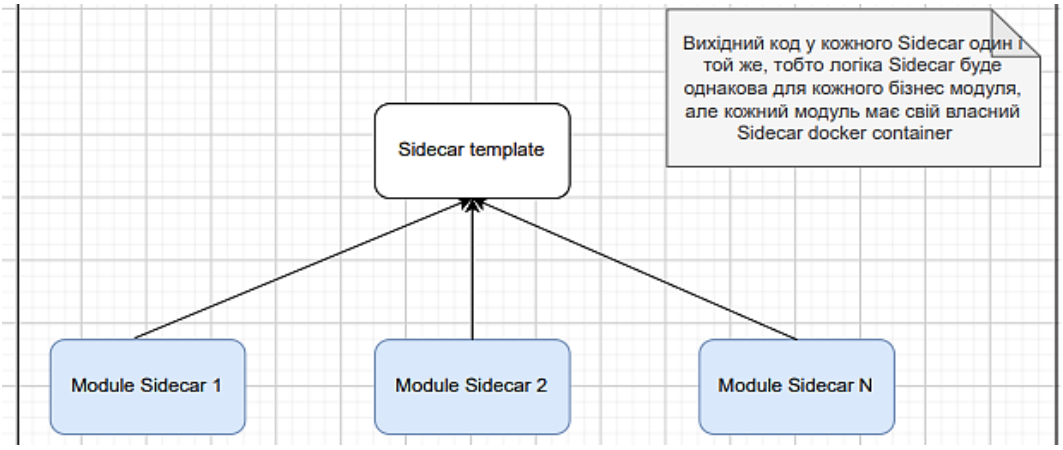

Рис. 5. Приклад коду сайдкара, який однаковий для усіх мікросервісів

Використання сайдкаром додаткових ресурсів

Під час створення високонавантажених систем необхідно враховувати той факт, що кожен сайдкар вимагає додаткових обчислювальних ресурсів для

виконання своїх функцій, що створює відповідне навантаження для систем із десятками та сотнями мікросервісів. Неконтрольоване створення об'єктів (потоків) може перезавантажувати середовище виконання, на приклад JVM, а при великих потоках даних як csv-файли, HTTP запити з великими payloads сайдкар може переповнити оперативну пам'ять та викликати помилки OutOfMemory. До того ж, мережеве середовище може стати нестабільним завдяки втратам пакетів даних та затримці (низька пропускна здатність) можуть порушити синхронізацію між вхідними та вихідними даними.

Такі виклики необхідно враховувати при реалізації високоефективного сайдкару. Тому, одним із перших прийнятих рішень може бути технологічна основа (платформа) на якій базується сайдкар. Quarkus підтримує Vert.x, що є реактивним засобом, який базується на Netty та забезпечує неблокуючих вхід-вихід. Дані характеристики дозволяють ефективно обробляти великі об'єми запитів у режимі реального часу. На рис. 6 наведено приклад традиційного блокуючого підходу для порівняння.

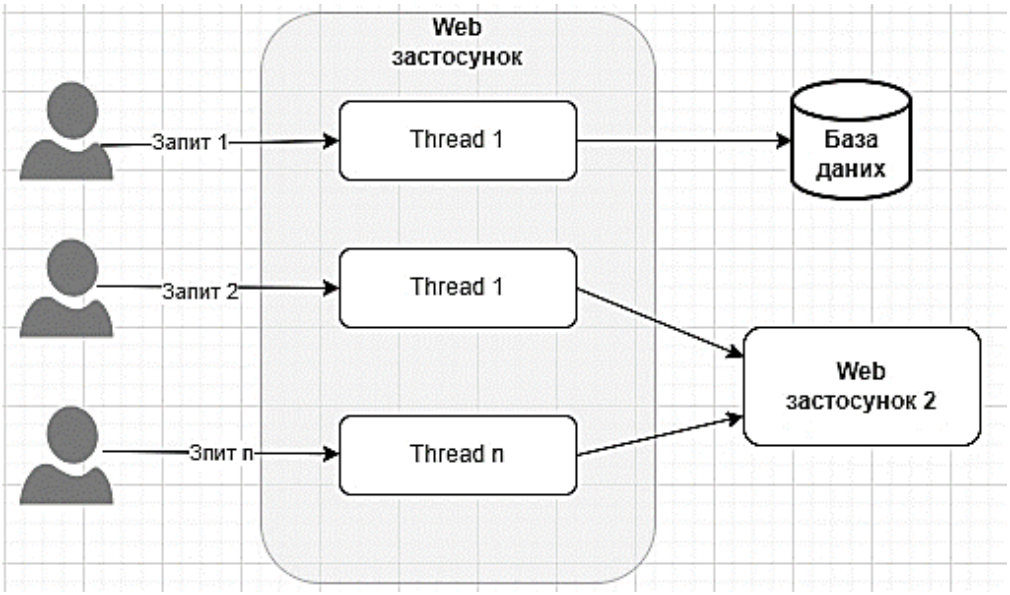

Рис. 6. Традиційний блокуючий підхід

При надходженні запиту веб-сервер створює новий потік, який буде зайнятий обробкою запиту до моменту його завершення. Потік може виконувати тривалі операції такі, як звернення до бази даних (до стороннього API), що забирає певний час. За умови, якщо на сервер надходить новий запит, то він буде призначений іншому доступному потоку, який залишається зайнятим до завершення обробки. Однак, можливості веб-сервера обмежені максимальною кількістю потоків, які він може підтримувати одночасно. На приклад, у Spring Boot – Tomcat використовує пул із 200 потоків. Такий розмір визначає рівень паралелізму програми.

До основного недоліку даного підходу можливо віднести те, що кожен запит блокує окремий потік. Такий потік не може бути використаний для інших задач, доки поточна обробка не завершиться. За умови, якщо усі доступні потоки зайняті, то нові запити чекають у черзі, поки не звільниться хоча б один потік. Крім того, кожен потік має свою ресурсну вартість, споживаючи

оперативну пам'ять та процесорний час. Створення великих пулів потоків призводить до важких та неефективних застосунків, що ускладнює реалізацію легкового та високопродуктивного сайдкара.

Рішенням даного ускладнення є використання неблокуючого асинхронного підходу, де компоненти системи реагують на події замість очікування результатів операцій, що знижує потребу у блокуванні ресурсів.

На рис. 7 наведено один і той самий потік Thread, який переключається між виконанням Запиту 1, Запиту 2 та Запиту n, що характеризує реактивний підхід, коли потік завжди у роботі і ресурси використовуються найбільш ефективно.

Quarkus здійснює низку внутрішніх оптимізацій, які спрямовані на зменшення споживання оперативної пам'яті, що робить його економічно ефективним рішенням для розробки сайдкара. До основних таких оптимізацій можливо віднести наступні:

– дотримання мінімальної моделі виконання – у оперативну пам'ять завантажуються виключно ті компоненти, які необхідні для роботи програми, що дозволяє уникнути додаткових витрат, пов'язаних із завантаженням та підтримкою непотрібних ресурсів, що значно зменшує загальне споживання оперативної пам'яті;

– оптимізація метаданих та налагодження – завдяки стисненню метаданих JVM та видаленню зайвої інформації такої, як некритичні дані про налагодження, Quarkus забезпечує мінімальне споживання оперативної пам'яті, яка потрібна для запуску та функціонування програми;

– вибіркове завантаження класів – у оперативну пам'ять завантажуються лише класи, які реально використовуються під час виконання програми. Такий підхід усуває необхідність завантаження невикористаних бібліотек та класів, що оптимізує споживання оперативної пам'яті.

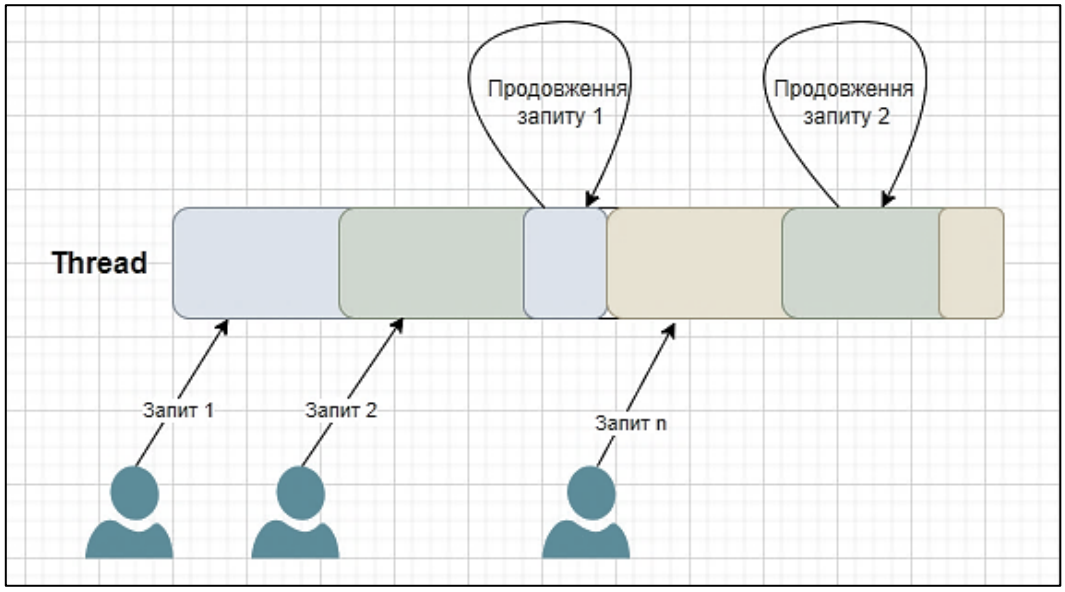

Рис. 7. Робота потоку в асинхронному підході

На рис. 8 представлено діаграму споживання оперативної пам'яті, взятої з офіційного веб-сайту Quarkus.

Отже, механізм Backpressure, реалізований у платформі Vert.x, забезпечує узгодження швидкостей між потоками даних шляхом динамічного управління чергами обробки. Таке узгодження досягається через паузи та відновлення потоків даних для уникнення перевантаження оперативної пам'яті.

Vert.x пропонує неблокуючу реалізацію HttpClient, що спрощує інтеграцію із зовнішніми компонентами, а для асинхронних комунікацій з використанням брокера повідомлень Kafka – є реалізації неблокуючого KafkaClient.

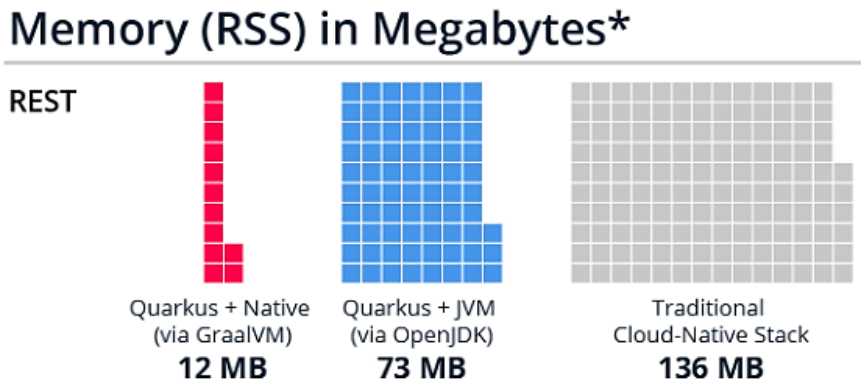

Рис. 8. Порівняння використання пам’яті за різними рішеннями

Висновки. Таким чином, запропоновано пропозиції щодо оптимізації роботи сайдкару в умовах високих навантажень. Розкрито роботу сайдкару та його основні функції. Фреймворк Quarkus має високу продуктивність та гнучкість, що дозволяє розробляти власну бізнес-логіку.

Розміщення авторизації у сайдкарі сприяє підвищенню рівня безпеки, оскільки уся інформація про авторизацію та її логіка управління концентруються у одному місці. Для створення платформи з використанням архітектури мікросервісів, потрібен механізм, спроможний здійснювати керування сотнями і тисячами контейнерів одночасно. Таке завдання може вирішити система з відкритим вихідним кодом Kubernetes.

Розкрито особливості розгортання мікросервісу із сайдкаром та використання сайдкаром додаткових ресурсів. Механізм Backpressure, реалізований у платформі Vert.x, забезпечує узгодження швидкостей між потоками даних шляхом динамічного управління чергами обробки.

Отже, завдяки комплексному підходу щодо проведення оптимізації у конфігурацію та архітектуру сайдкару, можливо досягти значного підвищення його продуктивності.

Список використаних джерел:

[1] Носко С.В., Бульба С.С. Моделі та методи взаємодії між розподіленими мікросервісними системами. *XVII Міжнародна науково-практична конференція магістрантів та аспірантів. Теоретичні та практичні дослідження молодих вчених.* – Х., 2023. – С. 102.

[2] Носко С.В., Бульба С.С., Семеренко Ю.О. Хмарні технології та їх застосування. *XXXI Міжнародну науково-практичну конференція MicroCAD-2023. Інформаційні*

технології: наука, техніка, технологія, освіта, здоров'я. 17-20 травня 2023 р. – Х.: НТУ «ХПІ». – 2023. – С. 1191.

[3] Тарновецька О.Ю., Осадчук Р.Р. Дослідження методів взаємодії між сервісами при мікросервісній архітектурі програмного забезпечення. *Науковий журнал «Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки».* – К.: ТНУ імені В.І. Вернадського, 2024. – Т. 35(74). – № 4 – С. 208-217. – DOI https://doi.org/10.32782/2663-5941/2024.4/31.

[4] Коломійцев О.В., Бульба С.С., Соловйова О.І., Носко С.В. Засоби побудови додаткового рівня системи комунікацій у мікро-сервісній архітектурі. *Грааль науки: міжнар. наук. журнал.* – Вінниця : ГО «Європейська наукова платформа»; НУ «Інститут науково-технічної інтеграції та співпраці», 2024. No 46. С. 651-659. – URL: https://doi.org/10.36074/grail-of-science.29.11.2024.084.

[5] Носко С.В., Бульба С.С., Коломійцев О.В., Лисиця Д.О., Молчанов Г.І. Пропозиції щодо авторизації в сайдкар компоненті мікросервісної архітектури. *Системи управління, навігації та зв'язку*. – Полтава: НУ «ПП», 2025. № 1(7). С. 116–123. – URL: https://journals.nupp.edu.ua/sunz/issue/view/127/68.

[6] Коломійцев О.В., Бульба С.С., Носко С.В. Метод підвищення стійкості сайдкара до зовнішніх навантажень. *XII Міжнародна науково-технічна конференція. Інформатика, управління та штучний інтелект (ІУШІ-2025).* 14-16 травня 2025 року. – Х.: НТУ «ХПІ», 2025. – С. 57.

[7] Kolomiitsev O., Nosko S., Bulba S. A method of increasing the siding's resistance to external loads. *XXXIII міжнародна науково-практична конференція MicroCAD-2025. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: тези доповідей.* 14-17 травня 2025 р. – Х.: НТУ «ХПІ». – С. 1611.

SUGGESTIONS FOR OPTIMIZING THE SIDECAR OPERATION UNDER HIGH LOADS

SCIENTIFIC RESEARCH GROUP:
Oleksii Kolomiitsev
Doctor of Technical Sciences, Professor, Honored Inventor of Ukraine,
Professor of the Department
National Technical University «Kharkiv Polytechnic Institute», Ukraine
Serhii Bulba
PhD, Associate Professor,
Associate Professor of the Department
National Technical University «Kharkiv Polytechnic Institute», Ukraine
Serhii Nosko
Postgraduate Student of the of the Department
National Technical University «Kharkiv Polytechnic Institute», Ukraine
Volodymyr Skorodielov
PhD, associate professor,
Professor of the Department
National Technical University «Kharkiv Polytechnic Institute», Ukraine
Yurii Rieznikov
PhD in Engineering, Senior Researcher
Lead Researcher
State scientific research institute of armament and military equipment testing and certification, Ukraine

Anatolii Sobora
PhD, Senior Research,
Leading researcher
State scientific research institute of armament and military equipment testing and certification, Ukraine
Hennadii Heiko
PhD,
Associate Professor of the Department
National Technical University «Kharkiv Polytechnic Institute», Ukraine
Vadim Lilchytskyi
Head of Scientific Research Department
State scientific research institute of armament and military equipment testing and certification, Ukraine
Volodymyr Panchenko
senior lecturer
National Technical University «Kharkiv Polytechnic Institute», *Ukraine*
Nataliia Oliinyk
Researcher
State scientific research institute of armament and military equipment testing and certification, Ukraine
Olha Rieznikova
PhD in Engineering,
Senior Researcher
State scientific research institute of armament and military equipment testing and certification, Ukraine

Summary. *The article offers suggestions for optimizing the operation of the sidecar under high loads. The work of the sidekick and its main functions are described. It is proved that the Quarkus framework has high performance and flexibility, which allows you to develop your own business logic. The deployment of a microservice with a sidekick is shown. Authorization provides a high level of security within the microservice architecture. The features of using additional resources by the sidekick are revealed. The graphical material is provided.*
Keywords: *optimization, sidechain, high load, framework, microservice, pattern, pod, platform, code, request, application, system, program, request authorization.*

