УДК 004.4, 004.9
DOI https://doi.org/10.32782/IT/2024-2-6

Олександр КИРИЧЕНКО
асистент кафедри математичних проблем управління і кібернетики, Чернівецький національний університет імені Юрія Федьковича, вул. Коцюбинського 2, м. Чернівці, Україна, 58002
***ORCID:** 0009-0001-6982-3342*

Оксана КИРИЧЕНКО
PhD, асистент кафедри математичних проблем управління і кібернетики, Чернівецький національний університет імені Юрія Федьковича, вул. Коцюбинського 2, м. Чернівці, Україна, 58002
***ORCID:** 0000-0003-0282-9958*
***Scopus Author ID:** 57854102400*

Бібліографічний опис статті: Кириченко, О., Кириченко, О. (2024). Кешування даних у додатках з використанням безсерверної архітектури. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 2, 42–49, doi: https://doi.org/10.32782/IT/2024-2-6

КЕШУВАННЯ ДАНИХ У ДОДАТКАХ З ВИКОРИСТАННЯМ БЕЗСЕРВЕРНОЇ АРХІТЕКТУРИ

Час відповіді сучасного вебсервісу є дуже важливою характеристикою, яка має велике значення для задоволення потреб користувачів. Користувачі, які отримують миттєві відповіді від вебсервісу, сприймають його як більш зручний та привабливий для використання. В той самий час, для розробників важливо швидко та ефективно впроваджувати та масштабувати свої додатки з мінімальними витратами на обслуговування та інфраструктуру. Саме тому, широкої популярності набуває використання хмарних технологій для розробки та розгортання різноманітних додатків, а можливість кешувати дані дозволяє значно покращити швидкодію розробленого програмного забезпечення.

***Метою роботи** є розгляд різноманітних наявних опцій кешування даних у хмарному середовищі на прикладі AWS та обґрунтування використання безсерверної архітектури, яка стала новим підходом для розгортання вебсервісів у хмарному середовищі.*

***Методологія** проведеного порівняльного аналізу полягає у наведенні основних принципів роботи, особливостей та обмежень для FaaS (функції як сервісу) на AWS, огляді наявних інструментів для реалізації кешування даних, таких яких Amazon ElastiCache for Redis, API Gateway Caching, CloudFront in front of API Gateway. Розглянуто використання простого вебсервісу для отримання розкладу занять у навчальному закладі на вказаний день тижня, який складається з безсерверних компонент таких як Amazon API Gateway, AWS Lambda, Amazon DynamoDB. Здійснено порівняння швидкодії створеного вебсервісу з використанням різних підходів до кешування даних, для чого розгорнуто чотири різних варіанти додатка з використанням зазначених опцій для кешування даних та підготовлено тести для проведення тестування під навантаженням з використанням фреймворку Locust.*

***Наукова новизна** отриманих у роботі результатів полягає в формулюванні можливих стратегій реалізації кешування для вебсервісів розгорнутих у хмарному середовищі.*

***Висновки.** Застосування кешування даних показало його ефективність для збільшення швидкодії вебсервісу. В подальшому актуальним є обґрунтування вибору безсерверних компонент для реалізації кешування даних.*

Ключові слова: *програмне забезпечення, хмарні технології, безсерверна архітектура, вебсервіс, кешування, тестування під навантаженням.*

Oleksandr KYRYCHENKO
Assistant Professor, Department of Mathematical Problems of Control and Cybernetics, Yuriy Fedkovych Chernivtsi National University, 2, Kotsyubynskoho Str., Chernivtsi, Ukraine, 58002, ol.kyrychenko@chnu.edu.ua
***ORCID:** 0009-0001-6982-3342*

Oksana KYRYCHENKO
PhD, Assistant Professor, Department of Mathematical Problems of Control and Cybernetics, Yuriy Fedkovych Chernivtsi National University, 2, Kotsyubynskoho Str., Chernivtsi, Ukraine, 58002, o.kyrychenko@chnu.edu.ua
***ORCID:** 0000-0003-0282-9958*
***Scopus Author ID:** 57854102400*

To cite this article: Kyrychenko O., Kyrychenko O. (2024). Keshuvannia danykh u dodatkakh z vykorystanniam bezservernoi arkhitektury [Data caching in serverless applications]. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 2, 42–49, doi: https://doi.org/10.32782/IT/2024-2-6

DATA CACHING IN SERVERLESS APPLICATIONS

The response time of a modern web service is a crucial characteristic that has significant importance for meeting user needs. Users who receive instant responses from a web service perceive it as more convenient and appealing to use. At the same time, it is important for developers to quickly and efficiently deploy and scale their applications with minimal costs for maintenance and infrastructure. Therefore, the widespread adoption of cloud technologies for developing and deploying various applications is gaining popularity, and the ability to cache data allows for significant improvements in the performance of developed software.

***The purpose of the work** is to consider various existing options for data caching in the cloud environment using AWS as an example and to justify the use of serverless architecture, which has become a new approach for deploying web services in the cloud environment.*

***The methodology** of the comparative analysis involves presenting the main principles of operation, features, and limitations for FaaS (Function as a Service) on AWS, reviewing existing tools for implementing data caching, such as Amazon ElastiCache for Redis, API Gateway Caching, and CloudFront in front of API Gateway. The use of a simple web service to retrieve a schedule of classes in an educational institution for a specified day of the week, consisting of serverless components such as Amazon API Gateway, AWS Lambda, and Amazon DynamoDB, is considered. A comparison of the performance of the created web service using different approaches to data caching is conducted by deploying four different application variants with the mentioned data caching options and preparing tests for load testing using the Locust framework.*

***The scientific novelty** of the results obtained in the work is formulating possible strategies for implementing caching for web services deployed in a cloud environment.*

***Conclusions.** The implementation of data caching has shown its effectiveness in increasing the performance of the web service. Furthermore, justifying the choice of serverless components for implementing data caching remains relevant in the future.*

***Key words:** software, cloud technologies, serverless architecture, web service, caching, load testing.*

Актуальність проблеми. Безсерверні обчислення (serverless computing) – модель хмарних обчислень, для яких платформа динамічно керує виділенням машинних ресурсів (Adzic, 2017). Використовуючи таку модель розробники відповідають за код та звільняються від будь-якого контролю над ресурсами, на яких буде виконуватись цей код. Це не означає, що серверів взагалі немає, у такому випадку підтримка та управління інфраструктурою, такі як: забезпечення ресурсами, моніторинг, технічне обслуговування, масштабованість і відмовостійкість, є зоною відповідальності постачальника хмарних технологій та дозволяє розробникам зосередитися на розробці функціональності додатків, замість вирішення інфраструктурних питань.

Іноді безсерверні обчислення також іменують «Функція як послуга» (Function as a Service, FaaS), тому що одиницею коду є функція, яка виконується хмарною платформою (Adzic, 2017). Такий підхід був популяризований Amazon у 2014 році і AWS Lambda – першою загальнодоступною платформою, що пропонувала безсерверні обчислення (Miller, 2014). У 2016 році інші постачальники хмарних сервісів впровадили власні реалізації FaaS такі як Google Cloud Functions (3), Microsoft Azure Functions (4). Ідея полягає в тому, що додаток розбивається на набір незалежних функцій для обробки різноманітних подій, таких як, наприклад, HTTP запити. Код розроблених функцій розгортається у середовищі FaaS і постачальник хмарних послуг робить все інше, що необхідно для надання ресурсів, створення екземплярів віртуальних машин, керування процесами (Roberts, 2014). Горизонтальне масштабування є повністю автоматичним, еластичним і керується постачальником. Якщо додатку необхідно обробляти декілька запитів одночасно, це буде зроблено без будь-яких додаткових налаштувань. «Обчислювальні контейнери», в яких запускаються розроблені функції, є ефемерними, і постачальник FaaS створює та знищує їх виключно з огляду на потреби часу виконання. Найважливіше те, що за допомогою FaaS постачальник керує всім основним наданням і розподілом ресурсів – користувачеві взагалі не потрібно керувати кластером або віртуальною машиною (Chadha, 2021).

Таким чином, безсерверна архітектура стає новим та популярним підходом для розгортання вебсервісів у хмарному середовищі. Варто зазначити, що функції FaaS мають певні особливості та обмеження. Зокрема, це відсутність збереження стану між запусками функцій, що призводить до використання бази даних, зовнішнього кешу (наприклад, Redis)

або сховища файлів/об'єктів (наприклад, S3) для збереження стану запитів або додаткових даних, необхідних для обробки запиту.

Іншою важливою особливістю є обмеження тривалості виклику функції. Наразі «тайм-аут» для функції AWS Lambda становить щонайбільше п'ять хвилин перед примусовим припиненням виконання. Microsoft Azure та Google Cloud Functions мають подібні обмеження.

Також необхідно звернути увагу на спосіб підготовки функції для обробки конкретної події. Так, платформі FaaS потрібен деякий час, щоб ініціалізувати екземпляр функції перед запуском. Розрізняють два типи ініціалізації – «теплий запуск» – повторне використання екземпляра функції Lambda та її хост-контейнера з попередньої події або «холодний запуск» – створення нового екземпляра контейнера. Не дивно, що коли розглядати затримку запуску, саме ці холодні запуски викликають найбільше занепокоєння та впливають на загальну швидкодію додатка (Lloyd, 2018).

Для зменшення часу відгуку системи використовуються різні підходи, такі як: зменшення розміру коду, що запускається, оптимізація конфігурації середовища функції, підтримка екземпляра функції у "теплому стані", кешування (Ghosh, 2020).

Метою роботи є розгляд різноманітних наявних опцій кешування даних у хмарному середовищі на прикладі AWS та обґрунтування використання безсерверної архітектури, яка стала новим підходом для розгортання вебсервісів у хмарному середовищі.

Виклад основного матеріалу дослідження. У даній статті нами проведено порівняльний аналіз наявних інструментів для реалізації кешування даних на прикладі AWS Lambda.

Розглянемо простий безсерверний вебсервіс для отримання розкладу занять у навчальному закладі на вказаний день тижня (рис. 1).

Архітектура сервісу використовує Amazon API Gateway, AWS Lambda, Amazon DynamoDB, де Amazon API Gateway – це повністю керована служба, яка дозволяє розробникам легко створювати, публікувати, підтримувати, контролювати та захищати API будь-якого масштабу (9). Amazon DynamoDB – це безсерверна, повністю керована служба бази даних NoSQL із часом відповіді в мілісекундах у будь-якому масштабі, що дає змогу розробляти та запускати сучасні програми, сплачуючи лише за те, що використовується (10).

Зазначений вебсервіс надає RESTful API, створене за допомогою Lambda та API Gateway для ортимання необхідної інформації. DynamoDB забезпечує збереження даних. Lambda функція реалізована мовою програмування Python з використанням набору інструментів розробки програмного забезпечення boto3, який надає можливість взаємодії з хмарними сервісами AWS. Таке поєднання технологій дозволяє підтримувати високу продуктивність сервісу для будь-якої кількості викликів.

Для реалізації кешування розглянемо декілька доступних опцій хмарного сервісу AWS, а саме:

– Amazon ElastiCache for Redis;
– API Gateway Caching;
– CloudFront in front of API Gateway.

Наведемо короткий опис зазначених вище опцій.

Amazon ElastiCache for Redis.

Amazon ElastiCache – це вебсервіс, який спрощує розгортання, експлуатацію та масштабування сховища даних у пам'яті та кешування у хмарі. Служба підвищує продуктивність веб додатків, дозволяючи витягувати інформацію зі швидких керованих сховищ даних у пам'яті, замість того, щоб повністю покладатися на більш повільні дискові бази даних.

Amazon ElastiCache підтримує два механізми обробки даних у пам'яті з відкритим вихідним кодом:

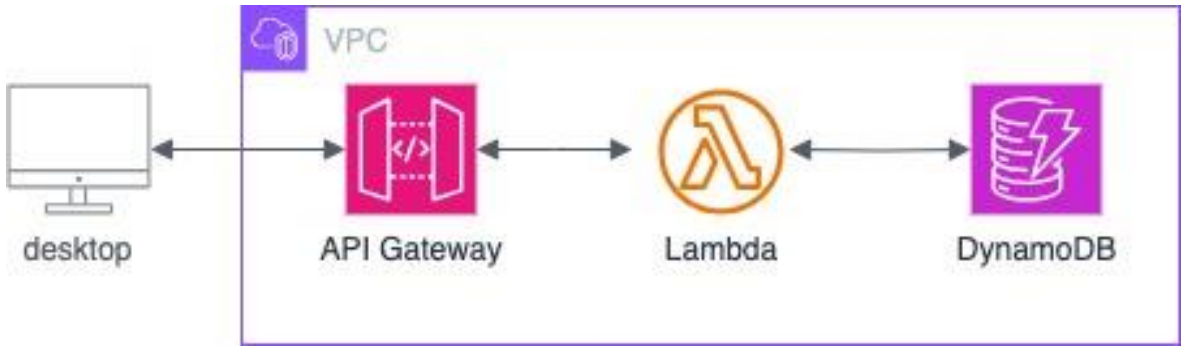

Рис. 1. Архітектура безсерверного вебсервісу

Redis – швидке сховище даних та кеш із відкритим вихідним кодом. Amazon ElastiCache для Redis – це Redis-сумісний сервіс у пам'яті, який забезпечує простоту використання і потужність Redis, а також доступність, надійність і продуктивність, що підходить для найвибагливіших додатків (11).

Використання Amazon ElastiCache дозволяє швидко додати можливість кешування даних до нашого вебсервісу (рис. 2).

У такому випадку для кожного запиту користувача API Gateway запустить виконання Lambda функції. Lambda функція перевірить наявність заздалегідь підготовлених даних в ElastiCache і, якщо дані присутні, отримає їх та підготує відповідь на запит. Якщо ж дані відсутні в кеші, то Lambda функція спробує отримати їх з постійного сховища, оновити кеш та повернути клієнту.

Інтеграція Amazon ElastiCache у вебсервіс надає можливість швидкого отримання функціоналу кешування даних, роботи зі складними типами даних. З іншого боку це досить вартісне рішення – $93 на день (~$1100 на рік) за 1 Гб (мінімальний розмір) даних (12).

API Gateway Caching.

У нашому додатку Amazon API Gateway забезпечує керування API для вебсервісу (рис. 3).

Ця служба підтримує кілька протоколів, включаючи REST API, WebSocket і HTTP/2, і надає такі функції, як аутентифікація та кешування. Кешування на рівні API передбачає кешування відповідей для всіх ресурсів в цьому API. Ця стратегія корисна, коли дані не оновлюються часто і необхідно зменшити навантаження на Lambda функції (Tota, 2023).

Такий підхід гарантує виклик Lambda функцій тільки у випадку відсутності даних, які знаходяться у кеші. Разом з тим API Gateway кешує дані протягом певного періоду часу життя (TTL) у секундах. Значення TTL за замовчуванням для кешування API становить 300 секунд. Максимальне значення TTL становить 3600 секунд.

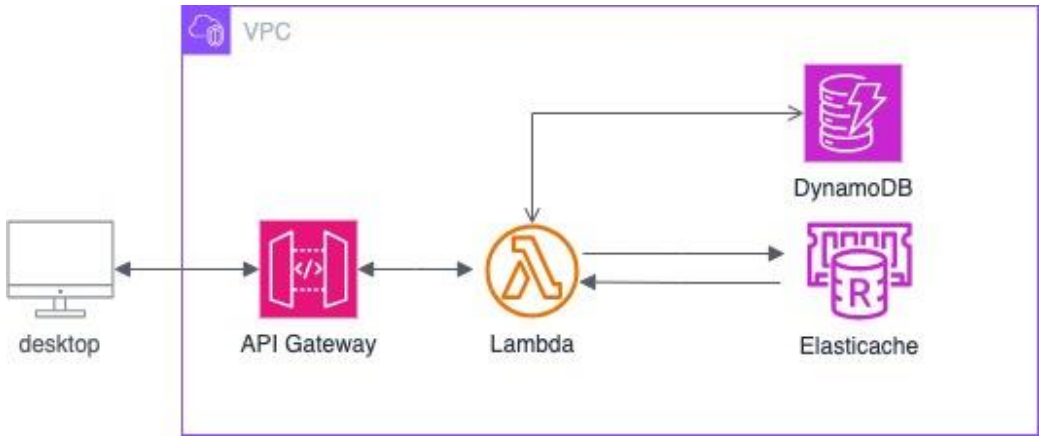

Рис. 2. Архітектура безсерверного вебсервісу з використанням Amazon ElastiCache

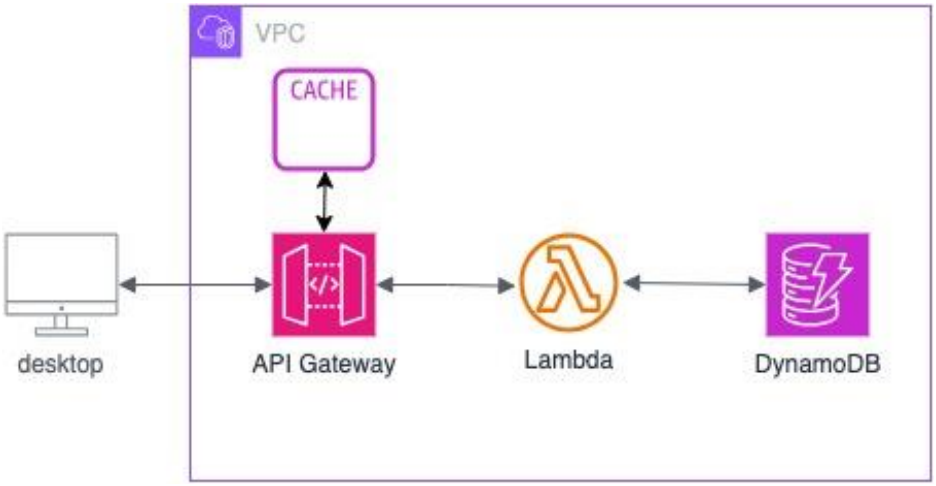

Рис. 3. Архітектура безсерверного вебсервісу з увімкненим кешуванням на рівні API Gateway

TTL=0 означає, що кешування вимкнено. Слід зазначити, що вартість подібного рішення становитиме 0,028 $/год за 0,5 ГБ даних (Shah, 2019).

CloudFront in front of API Gateway.

Відзначимо ще одну опцію для кешування даних – Amazon CloudFront. Amazon CloudFront – це високошвидкісна мережа доставляння контенту (CDN), яка безпечно передає дані клієнтам. Мережа CloudFront має понад 225 точок присутності (PoP), розподілених по всьому світу, і кожна PoP підключена до магістральної мережі Amazon Web Services (AWS), щоб забезпечити низьку затримку, високу швидкість передачі та високу доступність. Кеш є однією з найважливіших функцій CloudFront, оскільки він зберігає статичний вміст на кеш-сервері, розташованому в кожному PoP. Це означає, що коли кінцеві користувачі запитують вміст, вони отримують швидку відповідь без доступу до джерела даних (рис. 4).

Навіть для динамічного вмісту, який не може використовувати кеш, можна прискорити доставляння вмісту, використовуючи магістральну мережу AWS, яка підтримує низьку затримку та високу пропускну здатність (Lee, 2021).

Додавання CloudFront, з одного боку, значно пришвидшує роботу нашого вебсервісу та зменшує затримку доставляння даних, але, з іншого боку, CloudFront має обмеження на розмір вмісту, який можна кешувати, також, хоча CloudFront є економічно ефективним (вартість послуги починається від $0.085 за 10TB), можна легко перевищити ліміти використання, що призведе до неочікуваних витрат (Jeya, 2023).

Для порівняння швидкодії нашого вебсервісу з використанням різних підходів до кешування даних розгорнуто чотири різних варіанти додатка та підготовлено тести для проведення тестування під навантаженням.

Тестування під навантаженням – це практика моделювання реального використання або навантаження на будь-яке програмне забезпечення, вебсайт, веб додаток, API або систему для аналізу та визначення таких факторів, як швидкість реагування, зниження якості та масштабованість. Зазначені тести дають змогу вимірювати час відгуку, пропускну спроможність і рівні використання ресурсів, щоб визначити межу можливостей програмного забезпечення за умови пікового навантаження (17).

Для реалізації тестів використано бібліотеку Locust, яка дозволяє легко запускати тести під навантаженням, що розподілені на кількох машинах. Вона заснована на подіях, що дає змогу одному процесу обробляти багато тисяч користувачів, що працюють одночасно (Heyman, 2024).

Результати проведеного тестування вебсервісу наведені в таблиці 1.

Аналіз отриманих результатів свідчить про ефективність кешування як механізму збільшення швидкодії вебсервісу. Результати проведеного тестування вебсервісу без кешування та з використанням різних механізмів кешування (ElastiCache, API Gateway, CloudFront) наведені на графіку (рис. 5).

Висновки і перспективи подальших досліджень. Вибір підходу для реалізації кешування залежить від особливостей вимог та архітектури додатка. Використання саме безсерверної архітектури дозволяє певною мірою значно зменшити кількість рішень, які необхідно прийняти розробнику під час створення та розгортання програмного забезпечення у хмарному середовищі. Однак правильна взаємодія різних безсерверних компонент, що надаються постачальником хмарних технологій, є важливим питанням, яке може призвести до погіршення якості будь-якої послуги. Саме тому, при виборі стратегії кешування необхідно враховувати не тільки швидкодію, а і багато інших чинників таких як: складність, час збереження даних, вартість, безпеку тощо. Відкритим питанням залишається вибір оптимального рівня багаторівневого додатка, на якому варто здійснювати кешування.

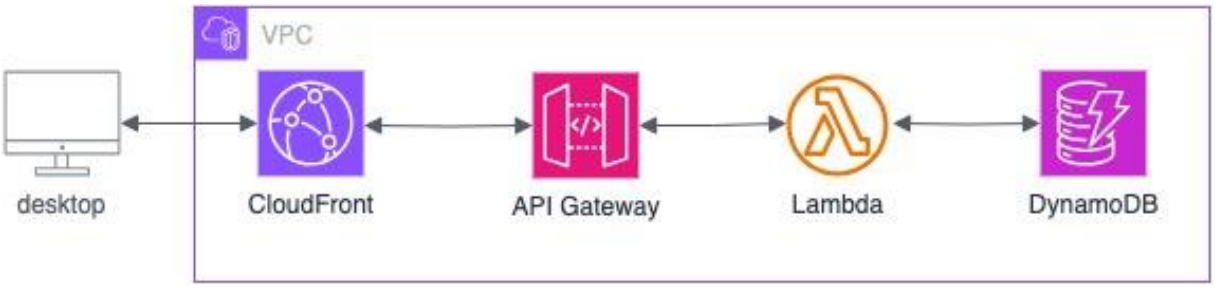

Рис. 4. Архітектура безсерверного вебсервісу з використанням CloudFront

Таблиця 1

Результати тестування під навантаженням вебсервісу без кешування та з кешуванням за допомогою різних підходів

Варіанти реалізації вебсервісу	Кількість потоків	Кількість запитів	Кількість помилок	Кількість запитів за секунду	Середній час відповіді в мілісекундах
Без кешування	10	5338	1	17	391
	20	5268	0	44	415
	30	7602	6	63	512
ElastiCache	10	15997	2	47	132
	20	15245	0	112	158
	30	23787	0	189	144
API Gateway	10	16454	0	54	126
	20	15030	0	125	145
	30	23808	0	198	131
CloudFront	10	15337	0	38	144
	20	14899	0	101	173
	30	23612	0	183	167

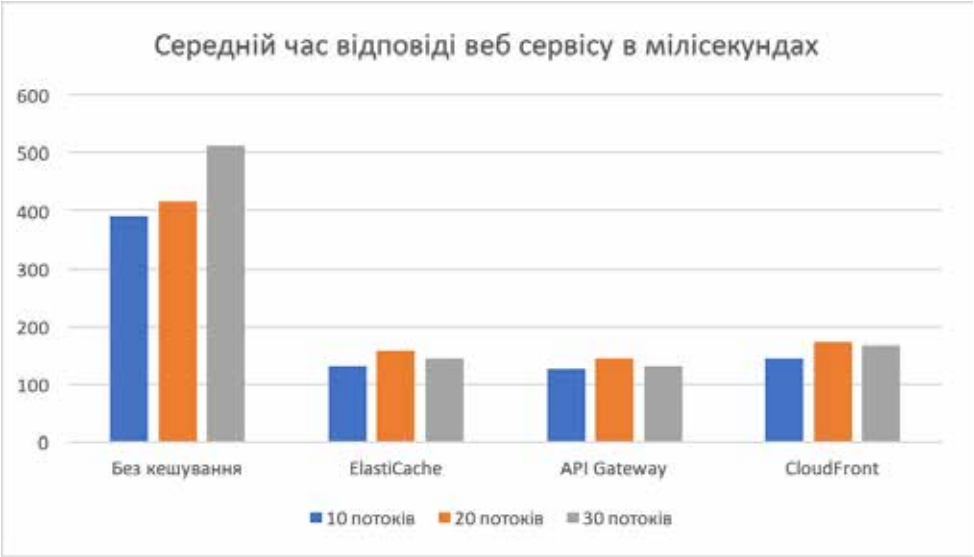

Рис. 5. Графічне представлення результатів тестування вебсервісу різними підходами

ЛІТЕРАТУРА:

1. Adzic, Gojko; Chatley, Robert. Serverless computing: Economic and architectural impact. *In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE'17).* ACM, New York, NY, pp. 884–889.

2. Miller, Ron. Amazon Launches Lambda, An Event-Driven Compute Service: веб-сайт. URL: https://techcrunch.com/2014/11/13/amazon-launches-lambda-an-event-driven-compute-service/ (дата звернення: 19.03.2024).

3. Cloud functions.: веб-сайт. URL: https://cloud.google.com/functions (дата звернення 17.03.24).

4. Azure Functions.: веб-сайт. URL: https://azure.microsoft.com/en-us/products/functions (дата звернення 01.03.24).

5. Roberts, Mike. Serverless Architectures : веб-сайт. URL: https://martinfowler.com/articles/serverless.html (дата звернення 01.03.24).

6. Mohak Chadha, Anshul Jindal, and Michael Gerndt. Architecture-Specific Performance Optimization of Compute-Intensive FaaS Functions. *In 2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. 478–483.

7. W. Lloyd, M. Vu, B. Zhang, O. David and G. Leavesley. Improving Application Migration to Serverless Computing Platforms: Latency Mitigation with Keep-Alive Workloads. *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, Switzerland, 2018, pp. 195-200, doi: 10.1109/UCC-Companion.2018.00056.

8. B. C. Ghosh, S. K. Addya, N. B. Somy, S. B. Nath, S. Chakraborty and S. K. Ghosh. Caching Techniques to Improve Latency in Serverless Architectures. *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, Bengaluru, India, 2020, P. 666-669. doi: 10.1109/COMSNETS48256.2020.9027427.

9. Amazon API Gateway: Create, maintain, and secure APIs at any scale. : веб-сайт. URL: https://aws.amazon.com/api-gateway/ (дата звернення 17.03.24).

10. Amazon DynamoDB: Serverless, NoSQL, fully managed database with single-digit millisecond performance at any scale.: веб-сайт. URL: https://aws.amazon.com/dynamodb/ (дата звернення 17.03.24).

11. AWS Caching Solutions.: веб-сайт. URL: https://aws.amazon.com/caching/aws-caching/ (дата звернення 17.03.24).

12. Amazon ElastiCache pricing.: веб-сайт. URL: https://aws.amazon.com/elasticache/pricing/?nc=sn&loc=5 (дата звернення 17.03.24).

13. Hugo Tota. AWS Gateway Cache Strategy to Improve Performance: веб-сайт. URL: https://www.linkedin.com/pulse/aws-gateway-cache-strategy-improve-performance-hugo-tota (дата звернення 23.03.24).

14. Bhargav Shah. API Gateway & Caching.: веб-сайт. URL: https://shahbhargav.medium.com/api-gateway-caching-3f86034ca491 (дата звернення 23.03.24).

15. Gabin Lee, Byounghwan Oh. Amazon CloudFront Cache Strategy for Successful Global Game Deployment.: веб-сайт. URL: https://aws.amazon.com/blogs/apn/amazon-cloudfront-cache-strategy-for-successful-global-game-deployment/ (дата звернення 21.03.24).

16. Jeiman Jeya. Let's Dive into AWS CloudFront.: веб-сайт. URL: https://community.ops.io/jei/lets-dive-into-aws-cloudfront-3i3k (дата звернення 10.03.24).

17. Load Testing: What is Load Testing & Why is Load Testing Important?: веб-сайт. URL: https://www.loadview-testing.com/learn/load-testing/ (дата звернення 23.03.24).

18. Jonatan Heyman, Lars Holmberg, Andrew Baldwin, Carl Byström, Joakim Hamrén, Hugo Heyman. What is Locust?: веб-сайт. URL: https://docs.locust.io/en/stable/what-is-locust.html (дата звернення 20.03.24).

REFERENCES:

1. Gojko Adzic and Robert Chatley. (2017). Serverless computing: Economic and architectural impact. *In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE'17)*. ACM, New York, NY, 884–889.

2. Ron Miller. (2014). Amazon Launches Lambda, An Event-Driven Compute Service. March 19, 2024, Retrieved from https://techcrunch.com/2014/11/13/amazon-launches-lambda-an-event-driven-compute-service/

3. Cloud functions. March 17, 2024, Retrieved from https://cloud.google.com/functions

4. Microsoft Azure Functions. March 01, 2024, Retrieved from https://azure.microsoft.com/en-us/products/functions

5. Mike Roberts. (2014). Serverless Architectures. March 01, 2024, Retrieved from https://martinfowler.com/articles/serverless.html

6. Chadha, M., Jindal, A., & Gerndt, M. (2021). Architecture-Specific Performance Optimization of Compute-Intensive FaaS Functions. *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 478–483.

7. Lloyd, W., Vu, M., Zhang, B., David, O. & Leavesley, G. (2018). Improving Application Migration to Serverless Computing Platforms: Latency Mitigation with Keep-Alive Workloads, *In 2018 11th IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, Switzerland, 2018, pp. 195-200, doi: 10.1109/UCC-Companion.2018.00056.

8. Ghosh, B. C., Addya, S. K., Somy, N. B., Nath, S. B., Chakraborty, S. & Ghosh, S. K. (2020). Caching Techniques to Improve Latency in Serverless Architectures. *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, Bengaluru, India, 2020, pp. 666-669, doi: 10.1109/COMSNETS48256.2020.9027427

9. Amazon API Gateway: Create, maintain, and secure APIs at any scale. March 17, 2024, Retrieved from https://aws.amazon.com/api-gateway/

10. Amazon DynamoDB: Serverless, NoSQL, fully managed database with single-digit millisecond performance at any scale. March 17, 2024, Retrieved from https://aws.amazon.com/dynamodb/

11. AWS Caching Solutions. March 17, 2024, Retrieved from https://aws.amazon.com/caching/aws-caching/

12. Amazon ElastiCache pricing. March 17, 2024, Retrieved from https://aws.amazon.com/elasticache/pricing/?nc=sn&loc=5

13. Hugo, Tota. (2023). AWS Gateway Cache Strategy to Improve Performance. March 23, 2024, Retrieved from https://www.linkedin.com/pulse/aws-gateway-cache-strategy-improve-performance-hugo-tota

14. Bhargav, Shah. (2019). API Gateway & Caching. March 23, 2024, Retrieved from https://shahbhargav.medium.com/api-gateway-caching-3f86034ca491

15. Gabin Lee & Byounghwan Oh. (2021). Amazon CloudFront Cache Strategy for Successful Global Game Deployment. March 21, 2024, Retrieved from https://aws.am March 23, 2024, Retrieved from azon.com/blogs/apn/amazon-cloudfront-cache-strategy-for-successful-global-game-deployment/

16. Jeiman, Jeya. (2023). Let's Dive into AWS CloudFront. March 10, 2024, Retrieved from https://community.ops.io/jei/lets-dive-into-aws-cloudfront-3i3k

17. Load Testing: What is Load Testing & Why is Load Testing Important? March 23, 2024, Retrieved from https://www.loadview-testing.com/learn/load-testing/

18. Jonatan Heyman, Lars Holmberg, Andrew Baldwin, Carl Byström, Joakim Hamrén & Hugo Heyman. What is Locust? March 20, 2024, Retrieved from https://docs.locust.io/en/stable/what-is-locust.html (дата звернення 20.03.24).