УДК 004.4, 004.9
DOI https://doi.org/10.32782/2663-5941/2024.4/16

Кириченко О.Л.
Чернівецький національний університет імені Юрія Федьковича

Кириченко О.О.
Чернівецький національний університет імені Юрія Федьковича

ВИКОРИСТАННЯ AWS APPSYNC ДЛЯ КОМУНІКАЦІЇ ВЕБДОДАТКІВ У РЕАЛЬНОМУ ЧАСІ

У статті детально проаналізовано можливість використання хмарної технології Amazon Web Services (AWS) AppSync для організації ефективної, надійної та масштабованої комунікації між різними вебдодатками в реальному часі. Розглянуто особливості моделі безсерверних обчислень з точки зору ефективного використання обчислювальних ресурсів та економічної вигоди порівняно з утриманням та налаштуванням постійної серверної інфраструктури. Проведено порівняльний аналіз рішень для реалізації комунікації в реальному часі між сервісами на прикладі AWS AppSync та WebSockets через API Gateway. Розроблено приклади додатків з використанням зазначених технологій. Здійснено вимірювання швидкодії створених вебсервісів з використанням різних підходів до реалізації отримування інформації в реальному часі за допомогою навантажувального тестування. Тестування здійснювалося відповідно до декількох тестових сценаріїв, які емулюють поведінку вебдодатків під час здійснення комунікації в реальному часі. Аналіз результатів тестування дозволяє зробити висновок про оптимальність вибору технології AWS AppSync для розв'язання поставленої задачі. Під час тестування були виявлені та зафіксовані певні обмеження щодо застосування AWS AppSync, такі як використання GraphQL, кількість запитів та підключень, зниження продуктивності при передачі великої кількості даних, а також проаналізовано вартість використання відповідного хмарного сервісу, яка залежить від кількості запитів, з'єднань та передачі даних, що може бути більш вигідним для складних запитів та інтерактивних додатків. Сформулювано основні критерії для вибору AWS AppSync при розробці хмарного додатка для реалізації комунікації в реальному часі, такі як масштабування, інтеграція з іншими сервісами, вбудована підтримка механізмів автентифікації й авторизації. Зроблено висновок стосовно остаточного вибору між AWS AppSync та API Gateway WebSocket, що залежить від особливостей вимог та архітектури додатка для отримування інформації в реальному часі, таких як складність запитів, вимоги до масштабованості, вартість та рівень необхідної інтеграції з іншими сервісами.

Ключові слова: *хмарні обчислення, аналітичні дослідження, хмара, хмарні технології, безсерверна архітектура, AWS (Amazon Web Services), тестування під навантаженням, вебдодаток, вебсервіс.*

Постановка проблеми. У сучасному світі, де передача даних у реальному часі є критично важливою для багатьох вебдодатків, необхідність у швидких та ефективних засобах комунікації стає все більш актуальною. Вебдодатки, які забезпечують оновлення в реальному часі, такі як чати, інформаційні панелі, та сервіси для спільної роботи, вимагають стабільного та надійного способу передачі даних. Одним із таких рішень є використання AWS AppSync.

Аналіз останніх досліджень і публікацій. Хмарні обчислення є дуже популярними завдяки численним перевагам, які вони надають. Зниження витрат на інфраструктуру та обслуговування, гнучке збільшення або зменшення ресурсів відповідно до потреб, висока доступність та відновлення після збоїв, оптимізація роботи та ефективність використання ресурсів, високий рівень захисту даних та інформаційна безпека – це деякі з переваг хмарних обчислень, які приваблюють бізнес [1].

Подібно до хмарних технологій, real-time communication (RTC) є ще однією важливою сферою, яку досліджують компанії для свого зростання. RTC дозволяє миттєво обмінюватися інформацією, що значно скорочує час прийняття рішень і підвищує продуктивність. Завдяки цьому компанії можуть оперативно вирішувати проблеми, що виникають, і ефективно співпрацювати. Це особливо важливо для служб підтримки клієнтів, де швидке реагування на запити клієнтів покращує рівень задоволеності користувачів. Реалізація RTC у вебдодатках дозволяє компаніям легко масштабувати свої сервіси, відповідаючи на зростаючі потреби бізнесу. Наприклад, в е-комерції реальний час допомагає миттєво оновлювати

інформацію про доступність товарів, що зменшує ризик подвійних продажів і покращує управління запасами. Таким чином, real-time communication у вебдодатках не тільки підвищує продуктивність і покращує користувацький досвід, але й надає гнучкі рішення для підтримки бізнес-процесів у різних умовах. Це робить RTC ключовим елементом сучасних вебдодатків і бізнес-рішень [2].

Окремим напрямком у хмарних технологіях є безсерверні обчислення (serverless computing) – модель хмарних обчислень, для яких платформа динамічно керує виділенням машинних ресурсів [3]. Використання безсерверної архітектури для реалізації комунікацій у реальному часі надає значні переваги, що робить її привабливим варіантом для сучасних вебдодатків. Так, використання таких сервісів, як AWS Lambda та API Gateway, дозволяє додаткам автоматично масштабуватися у відповідь на попит. Це означає, що під час пікових навантажень автоматично додаються необхідні ресурси, що забезпечує стабільну роботу та позитивний користувацький досвід. Така масштабованість є критично важливою для додатків, що потребують оновлень у реальному часі, де попит користувачів може значно коливатися [4].

Безсерверні платформи зазвичай працюють за моделлю «оплата за фактичне використання», стягуючи плату лише за реальний час роботи та використані ресурси. Ця модель є більш економічно вигідною порівняно з утриманням та налаштуванням постійної серверної інфраструктури, яка часто може бути недовантаженою. Переваги вартісної ефективності особливо відчутні для додатків із нерівномірними навантаженнями [5].

Безсервені архітектури, у поєднанні з сервісами, такими як AWS AppSync або WebSockets через API Gateway, забезпечують канали комунікації з низькою затримкою. Ці сервіси гарантують, що оновлення надходять до користувачів у реальному часі, підвищуючи швидкість реагування та інтерактивність додатків. Затримка мінімізується завдяки використанню глобально розподіленої інфраструктури та граничних локацій [6].

Постановка завдання. Метою дослідження є можливість та ефективність використання AppSync для забезпечення надійної та масштабованої комунікації між різними вебдодатками у хмарному середовищі.

Об'єкт дослідження: процеси та механізми комунікації між вебдодатками у розподілених системах.

Предмет дослідження: використання технології AppSync для організації комунікації між пристроями, включаючи архітектурні особливості, методи синхронізації даних та забезпечення безперервного обміну інформацією.

Виклад основного матеріалу дослідження. У даній статті проведено порівняльний аналіз рішень для реалізації комунікації в реальному часі між сервісами на прикладі AWS AppSync та WebSockets через API Gateway.

Розглянемо простий вебсервіс, який є частиною системи для масової генерації pdf документів з результатами сесії для кожного студента (рис. 1). Цей сервіс дозволяє в реальному часі отримувати інформацію стосовно статусу чергової генерації.

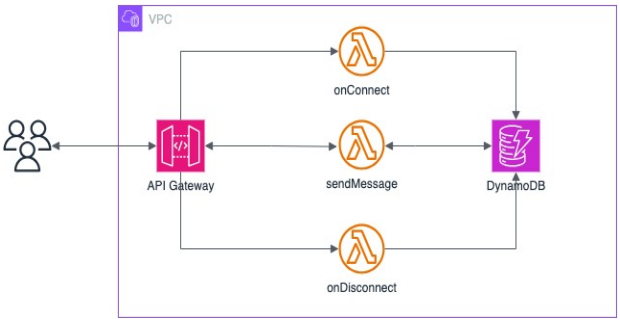

Рис. 1. Архітектура вебсервісу для отримування інформації в реальному часі

Архітектура сервісу використовує Amazon API Gateway, AWS Lambda, Amazon DynamoDB, де Amazon API Gateway – це повністю керований сервіс для розробників, який спрощує публікацію, обслуговування, моніторинг, захист та використання API у будь-яких масштабах [7], Amazon DynamoDB – це повністю керована безсерверна база даних NoSQL на основі пар «ключ-значення», яка створена для запуску високопродуктивних програм у будь-якому масштабі [8].

На основі API Gateway можна створити WebSocket API зі збереженням стану з'єднання в DynamoDB. Такий підхід дозволить WebSocket API викликати інші сервіси на основі вмісту повідомлень, які надходять від клієнтів. На відміну від REST API, який отримує запити та відповідає на них, WebSocket API забезпечує двосторонній зв'язок між клієнтськими додатками та серверною частиною. Це означає, що сервер може надсилати повідомлення безпосередньо підключеним клієнтам [9].

WebSocket API складається з одного або кількох маршрутів. Вибір маршруту здійснюється на основі повідомлень, які повинні містити властивість "action". За замовчуванням є три маршрути, які вже визначені в WebSocket API – $connect, $disconnect і $default. Архітектура розробленого сервісу передбачає створення спеціального марш-

руту – sendMessage. Таким чином, маршрути виконують наступні завдання:

$connect – під час виклику даного маршруту функція Lambda (onConnect) додасть ідентифікатор підключення до DynamoDB.

$disconnect – під час виклику даного маршруту функція Lambda (onDisconnect) видалить ідентифікатор з'єднання з DynamoDB.

sendMessage – під час виклику даного маршруту тіло повідомлення буде надіслано на всі підключені сервіси.

Схема взаємодії стороннього сервісу з нашим додатком буде складатися з наступних кроків (рис. 2):

– спочатку клієнти встановлюють WebSocket з'єднання за допомогою API GateWay;

– клієнт підписується на деякі теми, надсилаючи повідомлення через WebSocket (подія $connect);

– запускається відповідна Lambda функція на маршруті $connect і створює підписку: ідентифікатор підключення WebSocket клієнта та тему, на яку він хоче підписатися;

– потім клієнт робить запит, який містить інформацію для інших клієнтів;

– цей запит обробляється Lambda функцією на маршруті sendMessage, формується повідомлення та надсилається клієнтам, підписаним на вказану в запиті тему через відповідні веб-сокети;

– клієнти отримують повідомлення та оновлюються.

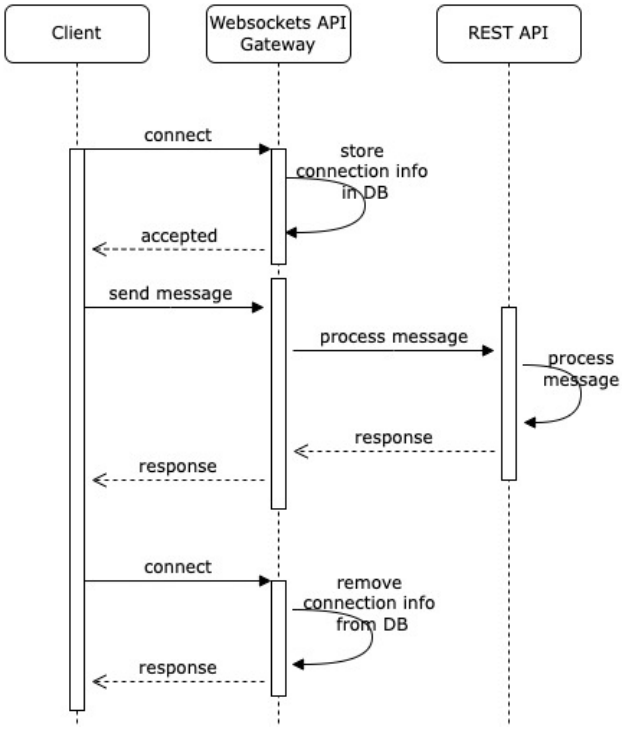

Рис. 2. Схема взаємодії клієнта з WebSocket API

Іншим підходом для реалізації комунікації в реальному часі з застосуванням безсерверних технологій є використання AWS AppSync.

AWS AppSync – це безсерверний GraphQL сервіс для мобільних, веб та корпоративних додатків.

AWS AppSync дозволяє розробникам підключати свої програми та служби до даних і подій за допомогою безпечних, безсерверних і високопродуктивних API GraphQL і Pub/Sub [10].

AWS AppSync використовує переваги підписки GraphQL для виконання операцій у реальному часі, надаючи дані клієнтам, які вирішують прослуховувати певні події з серверної частини. Це означає, що можна без особливих зусиль створити будь-яке підтримуване джерело даних у режимі реального часу в AWS AppSync, а керування з'єднанням автоматично оброблятиметься клієнтом AWS AppSync, використовуючи WebSocket як мережевий протокол між клієнтом і сервісом (рис. 3).

Рис. 3. Підписки на GraphQL у реальному часі в AppSync

Архітектура нашого сервіса із застосуванням AppSync відображена на рис. 4.

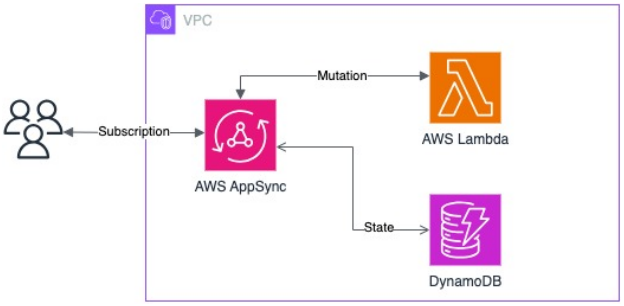

Рис. 4. Архітектура вебсервісу для отримування інформації в реальному часі з застосуванням AppSync

Клієнти вебсервісу підписані на певну мутацію GraphQL, яка змінює стан генерації pdf документів у DynamoDB. Усі вони отримують інформацію, пов'язану з станом поточної генерації, одночасно. Це гарантує своєчасне отримання повідомлення

про успішне завершення генерації або про виникнення будь-яких помилок (рис. 5).

Варто зазначити, що AWS AppSync GraphQL захищено авторизацією IAM на сервері. Для надання доступу кінцевим користувачам використовується пул ідентифікаційних даних Amazon Cognito. AWS Lambda, яка відповідає за обробку бізнес даних, також використовує IAM роль для доступу до AppSync, що дозволяє уникнути несанкціонованого доступу до даних.

Для вимірювання швидкодії нашого вебсервісу з використанням різних підходів до реалізації отримування інформації в реальному часі ми розгорнули 2 різних варіанти додатку та підготували тестові сценарії.

Для реалізації тестів використовувався Locust, який є ефективним інструментом для навантажувального тестування систем з обміном даних у режимі реального часу завдяки своїй масштабованості, підтримці WebSocket, простоті написання сценаріїв та можливостям моніторингу і аналізу [11].

Результати тестування під навантаженням вебсервісу наведені в таблиці 1.

Порівняльний аналіз результатів тестування свідчить про те, що AWS AppSync є конкурентною опцією для побудови систем з комунікацією у реальному часі в хмарному середовищі. Як і інші опції AppSync має певні обмеження у застосуванні. Так, GraphQL запити можуть бути складними для написання та оптимізації. AWS AppSync має обмеження на кількість запитів та підключень, призводить до зменшення кількості активних WebSocket-з'єднань. Також запити, що вимагають значного часу обробки або залучають велику кількість даних, можуть знижувати продуктивність.

З іншого боку, AWS AppSync пропонує вбудовану інтеграцію з іншими сервісами AWS (DynamoDB, Lambda, Cognito), що спрощує налаштування та управління, автоматично масштабується для обробки великої кількості запитів та підключень, зберігаючи високу продуктивність.

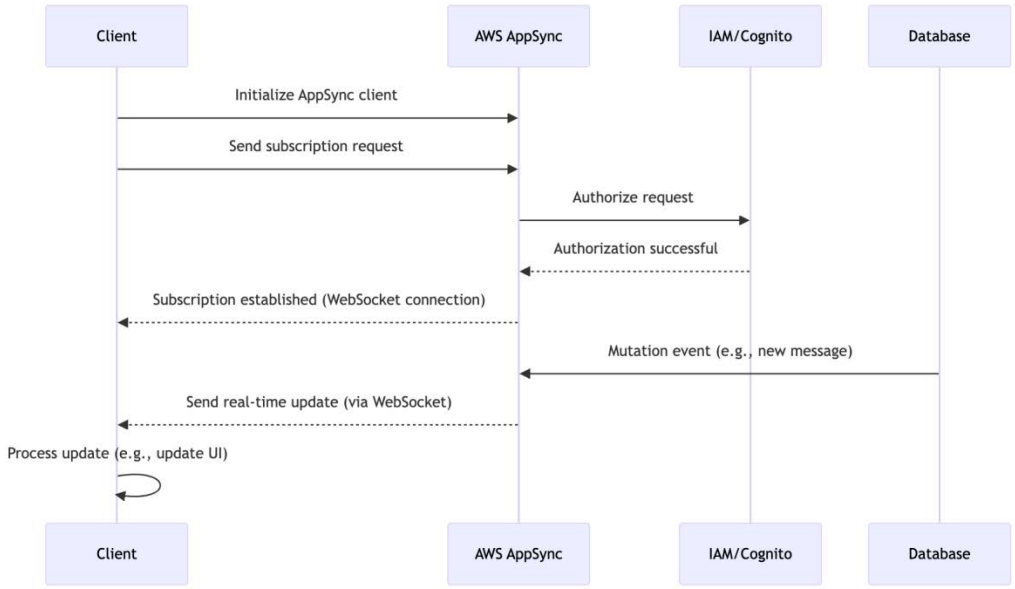

Рис. 5. Схема взаємодії клієнта з AppSync

Таблиця 1

Результати тестування під навантаженням вебсервісу з різними варіантами реалізації отримування інформації в реальному часі

	Кількість потоків	Кількість запитів за секунду	Середній час відповіді в мілісекундах	Мінімальний час відповіді в мілісекундах	Максимальний час відповіді в мілісекундах
WebSocket API	10	81	127	127	131
	20	137	136	135	139
	30	212	141	139	146
AppSync	10	119	121	120	125
	20	148	138	136	137
	30	236	144	142	145

Не менш важливим питанням є питання вартості використання хмарних сервісів. У випадку AWS AppSync ціноутворення засноване на кількості запитів, з'єднань та передачі даних, що може бути більш вигідним для складних запитів та інтерактивних додатків [12]. Тоді як для API Gateway WebSocket ціноутворення засноване на кількості з'єднань та переданих повідомлень, що виглядає більш прийнятним для простих та постійних з'єднань [13].

Таким чином, основними критеріями для вибору AWS AppSync при розробці хмарного додатку для реалізації комунікації в реальному часі є:

- можливість використання GraphQL;
- необхідність швидкої інтеграції з іншими сервісами AWS (DynamoDB, Lambda, Cognito);
- наявність автоматичного масштабування;
- наявність вбудованої підтримки AWS IAM, Cognito та інших механізмів автентифікації та авторизації.

Висновки. У даному дослідженні було здійснено порівняльний аналіз рішень для реалізації комунікації в реальному часі між сервісами на прикладі AWS AppSync та WebSockets через API Gateway.

На прикладі розробленого вебсервісу проведно тестування зазначених вище технологій, проведено вимірювання швидкодії вебсервісів за допомогою навантажувального тестування, зроблено висновок про оптимальність вибору технології AWS AppSync для розв'язання поставленої задачі та виявлені певні обмеження щодо застосування AWS AppSync.

Практична значимість дослідження полягає у формулюванні основних критеріїв для вибору AWS AppSync при розробці хмарного додатка для реалізації комунікації в реальному часі.

За отриманими результатами можна зробити висновок, що остаточний вибір між AWS AppSync та API Gateway WebSocket залежить від особливостей вимог та архітектури додатку для отримування інформації в реальному часі, таких як складність запитів, вимоги до масштабованості, вартість та рівень необхідної інтеграції з іншими сервісами.

Загалом, використання безсерверної архітектури для реалізації комунікації у реальному часі дозволяє створювати надійні, масштабовані та економічно ефективні системи, які відповідають вимогам сучасних вебдодатків, забезпечуючи високу продуктивність та задоволення користувачів. Це робить безсерверні компоненти ключовими елементами у створенні ефективних рішень для комунікацій у реальному часі.

Список літератури:

1. Dr. Masrath Begum, Pratiksha U., Sushmita B., Varshita V., Vinaykumar J. Build A Serverless Real Time Data Processing Application on AWS. *International Journal of Research Publication and Reviews*. 2023. Vol 4, No 6. P. 3592–3596.

2. Demystifying Real-Time Communication in Web Applications: A Comprehensive Guide to WebSockets and Socket.IO. URL: https://www.daillac.com/en/blogue/demystifying-real-time-communication-in-web-applications-a-comprehensive-guide-to-websockets-and-socket-io/ (дата звернення 26.06.2024)

3. Adzic G., Chatley R. 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, 2017. P. 884–889.

4. O'Riordan M., Wiggers S.-J. Using Serverless WebSockets to Enable Real-Time Messaging. URL: https://www.infoq.com/articles/serverless-websockets-realtime-messaging/ (дата звернення 23.06.2024)

5. Roberts M. Serverless Architectures. URL: https://martinfowler.com/articles/serverless.html (дата звернення 01.07.24).

6. Serverless with real-time communication (WebSockets). URL: https://dev.to/oskarkaminski/serverless-with-real-time-communication-websockets-2c6j (дата звернення 01.07.2024)

7. Amazon API Gateway: Create, maintain, and secure APIs at any scale. URL: https://aws.amazon.com/api-gateway/ (дата звернення 17.06.24).

8. Amazon DynamoDB: Serverless, NoSQL, fully managed database with single-digit millisecond performance at any scale. URL: https://aws.amazon.com/dynamodb/ (дата звернення 17.06.24).

9. Overview of WebSocket APIs in API Gateway. URL: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-overview.html (дата звернення 07.07.24).

10. What is AWS AppSync? URL: https://docs.aws.amazon.com/appsync/latest/devguide/what-is-appsync.html (дата звернення 07.07.24)

11. Heyman J., Holmberg L., Baldwin A., Byström C., Hamrén J., Heyman H. What is Locust? URL: https://docs.locust.io/en/stable/what-is-locust.html (дата звернення 20.06.24).

12. AWS AppSync pricing. URL: https://aws.amazon.com/appsync/pricing/ (дата звернення 03.07.24).

13. Amazon API Gateway pricing. URL: https://aws.amazon.com/api-gateway/pricing/ (дата звернення 03.07.24).

Kyrychenko O.L., Kyrychenko O.O. THE UTILIZATION OF AWS APPSYNC FOR REAL-TIME WEB APPLICATION COMMUNICATION

In this article a comprehensive analysis of AWS AppSync cloud technology's capability to enable effective, reliable, and scalable real-time communication between various web applications is conducted. The study emphasizes the benefits of serverless computing models, particularly their efficient resource utilization and cost-effectiveness compared to maintaining a permanent server infrastructure. A comparative analysis of real-time communication solutions, specifically AWS AppSync and WebSockets via API Gateway, is performed. Practical application examples employing these technologies are developed and scrutinized. The performance of the developed web services, utilizing various approaches to real-time data retrieval, is as-sessed through rigorous load testing. The testing adheres to multiple scenarios that simulate web application behaviours during real-time communication. The results underscore the efficacy of AWS AppSync for the given use case, despite certain identified limitations such as GraphQL usage constraints, request and connection caps, and performance declines when handling large data volumes. Additionally, the cost analysis of the cloud service, influenced by the volume of requests, con-nections, and data transfers, reveals potential cost benefits for complex queries and interactive applications. The study delin-eates key criteria for selecting AWS AppSync in the development of cloud applications aimed at real-time communication. These criteria include scalability, seamless integration with other services, and inherent support for authentication and author-ization mechanisms. The investigation also explores the inherent support for security features and the ease of implementation that AWS AppSync offers, making it a suitable choice for developers aiming to build robust and secure real-time applications. Ultimately, the study concludes that the choice between AWS AppSync and API Gateway WebSocket depends on the specific requirements and architecture of the real-time information retrieval application. Factors such as query complexity, scalability needs, cost considerations, and the necessary level of integration with other services are critical in determining the optimal solution. The comparative analysis provides a detailed understanding of both technologies, highlighting the scenarios where AWS AppSync's advantages can be maximized and identifying potential areas where alternative solutions might be more appropriate. The insights gained from this study serve as a valuable resource for developers and organizations considering the implementation of real-time communication capabilities in their web applications, guiding them towards making informed decisions based on their unique requirements and constraints.

Key words: *cloud computing, analytical research, cloud, cloud technologies, serverless architecture, AWS (Amazon Web Services), load testing, web application, web service.*