УДК 004.7(63.06) DOI: 10.31673/2786-8362.2025.016832

Мельник Ю.В., д.т.н.; **Отрох С.І.,** д.т.н.;
Донець А.Г., к.ф.-м.н.; **Колумбет В.П.,** PhD;
Сарафанніков О.В.

ОПТИМІЗАЦІЯ KAPPA-АРХІТЕКТУРИ ПРИ ПОБУДОВІ МАСШТАБОВАНИХ СИСТЕМ

Melnyk Y.V., Otrokh S.I., Donets A.G., Kolumbet V.P., Sarafannikov O.V. Kappa architecture optimization during building scalable systems. The article discusses the design and construction of scalable systems using efficient Lambda and Kappa architectures. Lambda architecture has many advantages, but the main disadvantage of this approach to designing scalable systems is its complexity due to duplication of data processing logic. The main target of the research is to describe and develop an alternative optimized model of the Kappa architecture, which consumes fewer resources but is excellent for real-time event processing, which will provide an effective tool for building scalable systems and simplify the choice. The result of the work is a comprehensive approach to building scalable systems, in which each component of the system will be optimized through event flow analysis, which will allow for significant improvements in real-time data processing systems.

Keywords: Kappa architecture, Lambda architecture, BigData, scalable systems, model optimization

Мельник Ю.В., Отрох С.І., Донець А.Г., Колумбет В.П., Сарафанніков О.В. Оптимізація Kappa-архітектури при побудові масштабованих систем. У статті розглядаються питання проектування та побудови масштабованих систем з використанням ефективних Lambda та Kappa архітектур. Lambda-архітектура, має багато переваг, але головним недоліком цього підходу до проектування масштабованих систем систем вважається його складність через дублювання логіки обробки даних. Основною метою дослідження є опис та розробка альтернативної оптимізованої моделі Kappa-архітектури, яка споживає менше ресурсів, але відмінно підходить для обробки подій у режимі реального часу, що дозволить отримати ефективний інструмент побудови масштабованих систем та спростить вибір. Результатом роботи є комплексний підхід до побудови масштабованих систем, в якому завдяки аналізу потоку подій буде оптимізовуватися кожна складова системи, що дозволить значно вдосконалити систем обробки даних у реальному часі.

Ключові слова: Kappa-архітектура, Lambda-архітектура, BigData, масштабовані системи, оптимізація моделі

Вступ

Kappa-архітектура базується на єдиному потоці даних, який обробляється в реальному часі. Для оптимізації такої системи можна використовувати математичні моделі та графічне представлення, щоб візуалізувати та аналізувати ключові параметри, такі як затримка, пропускна здатність та навантаження на систему. Kappa-архітектура є потужним підходом для обробки даних у реальному часі, але її ефективність залежить від правильного налаштування та оптимізації.

Аналіз останніх досліджень. Через зростання потреб у реальному часі та складності Lambda-архітектури, Kappa-архітектура набуває популярності, яка надає можливості використання єдиного потоку даних для обробки як реального часу, так і існуючих, накоплених історичних даних [1]. Сучасні дослідження та публікації дозволяють виділити такі ключові напрямки при побудові масштабованих систем з використанням Kappa-архітектури: Інтеграція з сучасними технологіями – використання Flink, Spark Streaming для ефективної обробки потоків [1]; вбудована аналітика – інтеграція Kappa з OLAP-системами для одночасного real-time та batch-аналізу [2]; stateful-оптимізації – нові алгоритми управління станом, наприклад, інкрементальні checkpoint у Flink, скорочують накладні витрати на 30-50% та покращують управління станами у Flink для складних workflow [3]; гнучке масштабування та горизонтальне масштабування – досліди з auto-scaling на основі ML передбачають навантаження, уникаючи over-provisioning, динамічне виділення ресурсів для потокової обробки[4]; відмова від надмірності – техніки like "log compaction" у Kafka ефективніше за Lambda-підходи для історичних даних [5]; ефективне кешування – зменшення затримок за ра-

хунок інтелектуального кешування проміжних даних [6]; стрімка еволюція Kafka та Flink – оновлення ядра, такі як Kafka Tiered Storage, зменшують вартість зберігання та підвищують пропускну здатність [7]; edge-оптимізації – обробка даних на периферії з подальшою синхронізацією зменшує затримки [8]. Перспективи: Активно тестуються концепції "Kappa++" з підтримкою транзакцій та крос-кластерної реплікації. Головний виклик – зберегти простоту архітектури при додаванні нових функцій та їх оптимізації.

Метою роботи є оптимізація Kappa-архітектури для системи аналізу логів, яка обробляє мільйони подій щодня. Метою оптимізації є зменшення затримки обробки, підвищення пропускної здатності та забезпечення стабільності системи. В результаті система стає швидшою, стабільною та здатною обробляти великі обсяги даних. Математичне та графічне представлення дозволяє наочно продемонструвати ефективність оптимізації Kappa-архітектури.

Виклад основного матеріалу дослідження

На відміну від Lambda, у Kappa-архітектурі потокові дані проходять одним шляхом. Усі дані приймаються як потік подій у розподіленому та відмовостійкому єдиному журналі – логу подій. Там події впорядковуються і поточний стан події змінюється тільки при додаванні нової події. Аналогічно до рівня прискорення Lambda-архітектури, вся обробка подій виконується у вхідному потоці і зберігається як уявлення в режимі реального часу. Якщо необхідно повторно обчислити весь набір даних, як пакетному рівні в Lambda-архітектурі, потік відтворюється заново. Для своєчасного завершення обчислень використовується паралелізм, можна сказати, що Kappa-архітектура – це спрощення Lambda-підходу до проектування масштабованих систем, коли з моделі видалено рівень пакетної обробки даних. При цьому сховище даних є незмінним журналом тільки для додавання інформації. З журналу дані одразу передаються до системи потокових обчислень, за необхідністю збагачуючись даними з неканонічного сховища

Пропускна здатність. Пропускна здатність системи визначається кількістю подій, які система може обробити за одиницю часу. Для Kappa-архітектури пропускна здатність залежить від продуктивності кожного компонента (Kafka, Elasticsearch, Flink).

Формула для пропускної здатності:

$$P = \min\left(P_{Kafka}, P_{Elasticsearch}, P_{Flink},\right) \quad (1)$$

де:

P_{Kafka}– пропускна здатність Kafka,

$P_{Elasticsearch}$ – пропускна здатність Elasticsearch,

P_{Flink} – пропускна здатність Flink.

Затримка. Затримка – це час, необхідний для обробки однієї події. Вона складається із затримок у кожному компоненті:

$$Z = Z_{Kafka} + Z_{Elasticsearch} + Z_{Flink} \quad (2)$$

де:

Z_{Kafka} – затримка в Kafka,

$Z_{Elasticsearch}$ – затримка в Elasticsearch,

Z_{Flink} – затримка в Flink.

Навантаження на систему. Навантаження на систему можна виразити через кількість подій, які надходять за одиницю часу:

$$\lambda = \frac{N}{t} \quad (3)$$

де:

N – кількість подій,

t – час.

Оптимізація затримки. Для оптимізації затримки необхідно мінімізувати затримку в кожному компоненті. Наприклад, для Flink затримка може бути зменшена шляхом збільшення паралелізму:

$$Z_{Flink} = \frac{O_{sequential}}{T} \quad (4)$$

де:

$O_{sequential}$ – затримка при послідовній обробці,

T – кількість паралельних задач.

Графік затримки. На графіку показано затримку обробки даних у кожному компоненті системи до та після оптимізації (Рис. 1).

До оптимізації:

Kafka: 200 мс
Elasticsearch: 100 мс
Flink: 300 мс
Загальна затримка: 600 мс

Після оптимізації:

Kafka: 100 мс
Elasticsearch: 50 мс
Flink: 50 мс
Загальна затримка: 200 мс

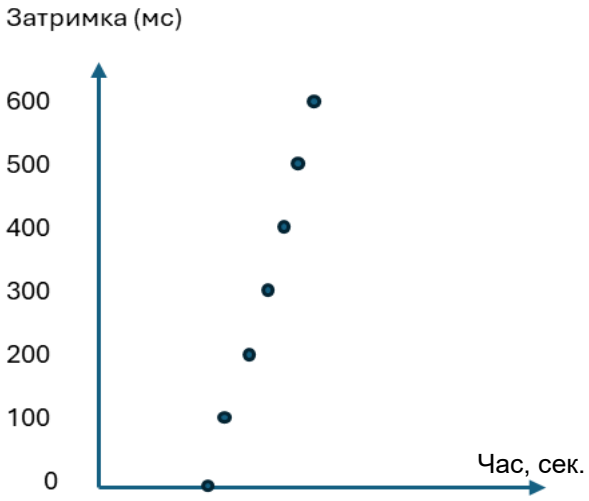

Рис. 1. Затримка обробки даних

Графік пропускної здатності. На графіку показано пропускну здатність системи до та після оптимізації (Рис. 2).

До оптимізації:

Пропускна здатність: 10,000 подій/сек

Після оптимізації:

Пропускна здатність: 50,000 подій/сек

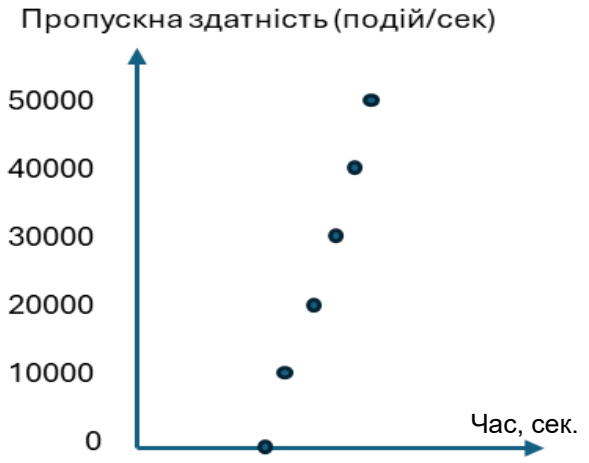

Рис. 2. Пропускна здатність системи

Графік навантаження. На графіку показано навантаження на систему до та після оптимізації (Рис. 3).

До оптимізації:

Навантаження: 15,000 подій/сек

Після оптимізації:

Навантаження: 60,000 подій/сек

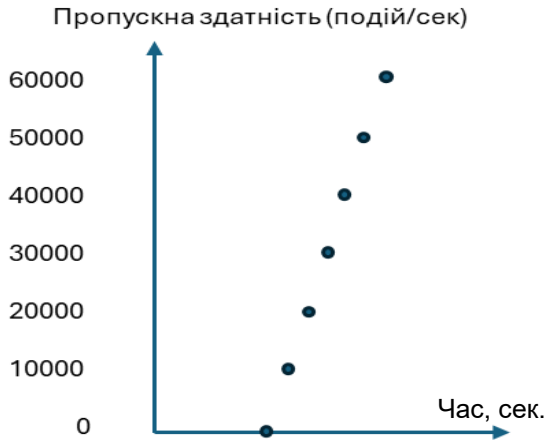

Рис. 3. Навантаження на систему

Висновки

Оптимізація Каппа-архітектури дозволяє створити ефективну та масштабовану систему для обробки даних у реальному часі. У прикладі з аналізом логів ми оптимізували кожен компонент системи: Kafka, Elasticsearch, Flink. В результаті системи стають швидшими, стабільними та здатні обробляти великі обсяги даних. Оптимізація – це безперервний процес, який вимагає постійного аналізу та вдосконалення. Математичне та графічне представлення дозволяє наочно продемонструвати ефективність оптимізації Kappa-архітектури. За допомогою формул можна визначити ключові параметри системи, такі як пропускна здатність, затримка та навантаження. Графіки допомагають візуалізувати результати оптимізації, показуючи, як зменшується затримка та зростає пропускна здатність. Цей підхід є корисним для аналізу та вдосконалення систем обробки даних у реальному часі.

Список використаної літератури:

1. Білоконь А. С., Борисов С. О., Усатенко М. В., Федорченко В. М. Аналіз функціонування розподілених систем обробки та зберігання даних. Системи управління навігації та зв'язку, Збірник наукових праць 2024, т. 3(77), С.84-88 URL: https://doi.org/: 10.26906/SUNZ.2024.3.08

2. Tanenbaum, Andrew S., and Maarten van Steen. Distributed Systems: Principles and Paradigms. 3rd ed., Pearson, 2017.

3. Мокін, В.Б., Крижановський, Є.М., Лучко, А.М., Білецький, Б.С., Жуков, С.О. 2021. Метод оптимізації інформаційних моделей масштабованих у просторі аналітичних веб-систем за критерієм повноти їхньої топологічної спостережуваності. Вісник Вінницького політехнічного інституту. 2021, 131–141. URL: https://doi.org/10.31649/1997-9266-2021-159-6-131-141.

4. D. Y. Paramartha, A. L. Fitriyani, and S. Pramana. Development of Automated Environmental Data Collection System and Environment Statistics Dashboard. Indonesian Journal of Statistics and Its Applications, vol. 5, no. 2, pp. 314-325, 2021. URL: https://doi.org/10.29244/ijsa.v5i2p314-325.

5. Мокін В. Б., Овчаренко І. І., Лучко А. М., Давидюк О. М. Побудова масштабованої інформаційно-пошукової системи для управління річковим басейном на основі реєстрів та онтологічних моделей. Математичне моделювання в економіці, № 2 (15), 2019.

6. Рудніченко М.Д. Методичні вказівки до розрахунково-графічної роботи з дисципліни «Моделі обробки структурованих и слабо структурованих масивів даних» для студентів

спеціальності - 126 Інформаційні системи та технології / Укл.: М.Д. Рудніченко, Н.В. Бут. – Одеса: ОНПУ, 2020. – 10 с. (Електронна версія), Реєстраційний номер №7534-РС-2020 (МВ11512)

7. Schuster, Werner. Nathan Marz on Storm, Immutability in the Lambda Architecture, Clojure. www.infoq.com. Nathan Marz interview, 2014.

8. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, Incorporated, 2017. 616 p.

Автори статті

Мельник Юрій – доктор технічних наук, професор, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ORCID: 0000-0002-5028-8749

Отрох Сергій – доктор технічних наук, професор, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

ORCID: 0000-0001-9008-0902

Донець Андрій – кандидат фізико-математичних наук, доцент, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

ORCID: 0000-0002-8122-051X

Колумбет Вадим – PhD, старший викладач, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

ORCID: 0000-0002-0871-9402

Сарафанніков Олександр – аспірант, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

ORCID: 0000-0002-8373-4629

Authors of the article

Melnyk Yurii – Doctor of Sciences (technical), Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

ORCID: 0000-0002-5028-8749

Otrokh Serhii – Doctor of Sciences (technical), Professor, National Technical University of Ukraine “Ihor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

ORCID: 0000-0001-9008-0902

Donets Andrii – Candidate of Sciences (physics and mathematics), Associate Professor, National Technical University of Ukraine “Ihor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

ORCID: 0000-0002-8122-051X

Kolumbet Vadym – PhD, senior lecturer, National Technical University of Ukraine “Ihor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

ORCID: 0000-0002-0871-9402

Sarafannikov Oleksandr – postgraduate, National Technical University of Ukraine “Ihor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

ORCID: 0000-0002-8373-4629