https://doi.org/10.31891/2307-5732-2025-359-57
УДК 004.6:004.75

СУПРУН ПАВЛО
Хмельницький національний університет
https://orcid.org/0009-0002-4903-1337
e-mail: pasha1apple@gmail.com
ПРАВОРСЬКА НАТАЛІЯ
Хмельницький національний університет
https://orcid.org/0000-0001-6001-3311
e-mail: margana2000007@gmail.com

ПОДІЄВО-КЕРОВАНІ КЕШОВАНІ ПРЕДСТАВЛЕННЯ ДАНИХ У ВИСОКОНАВАНТАЖЕНИХ ІНТЕГРАЦІЙНИХ ПОТОКАХ З АРХІТЕКТУРНИМ ЗАБЕЗПЕЧЕННЯМ ТА ГАРАНТІЯМИ УЗГОДЖЕНОСТІ

У сучасних розподілених інформаційних системах інтеграційні потоки характеризуються високою частотою змін та жорсткими вимогами до швидкодії. Прямий доступ до джерел істини створює ризик перевантаження операційних сховищ і нестабільності у роботі всієї інфраструктури. Це породжує необхідність у використанні кешованих представлень даних, здатних забезпечити швидкий доступ до інформації без постійного звернення до основних систем.

Разом з тим, застосування кешованих подань без механізмів актуалізації призводить до накопичення розбіжностей із джерелом істини. Проблема особливо загострюється у середовищах із високочастотними подіями та у випадка відмов окремих компонентів. Для підтримки цілісності бізнес-процесі необхідні архітектурні рішення, що зможуть гарантувати узгодженість між кешем і джерелом незалежно від характеру навантаження.

Подієво-керований підхід дозволяє формувати кешовані представлення на основі послідовності подій, що описують зміни стану даних. Кожна подія є незалежним повідомленням із визначеним ідентифікатором, часовою міткою та типом операції, що робить можливим відтворення канонічного стану системи. Така модель підтримує ідемпотентність обробки та впорядкування подій, а також створює умови для контролю вікна застарілості.

Архітектурне забезпечення включає використання спільної черги для приймання повідомлень, алгоритмів нормалізації та дедуплікації, трансакційне оновлення стану, а також політики розв'язання конфліктів між різними джерелами. Важливу роль відіграють механізми відновлення даних через повторне відтворення журналу подій та моніторинг якості синхронізації за допомогою інструментів спостереження. Це дозволяє інтеграційним потокам залишатись стабільними навіть під час пікових навантажень.

Таким чином, подієво0орієнтовані кешовані представлення даних із архітектурним забезпеченням та гарантіями узгодженості формують цілісний підхід до побудови високонавантажених інтеграційних систем. Поєднання масштабованості, стійкості до збоїв і передбачуваності поведінки робить їх придатним інструментом для сучасних розподілених архітектур, коли вимоги до продуктивності та правильності даних є критичними.

Ключові слова: *подієво-орієнтована архітектура, кешовані представлення даних, бази даних, узгодженість, інтеграційні потоки, журнал подій, масштабування, спостережність.*

SUPRUN PAVLO
PRAVORSKA NATALYA
Khmelnitsky national university, Ukraine

EVENT-DRIVEN CACHED DATA REPRESENTATIONS IN HIGH-LOAD INTEGRATION FLOWS WITH ARCHITECTURAL SUPPORT AND CONSISTENCY GUARANTEES

In modern distributed information systems, integration flows are characterized by a high frequency of changes and strict requirements for speed. Direct access to sources of truth creates the risk of overloading operational storage and instability in the operation of the entire infrastructure. This creates the need to use cached data representations that can provide fast access to information without constant access to the main systems.

However, the use of cached representations without update mechanisms leads to the accumulation of discrepancies with the source of truth. The problem is especially acute in environments with high-frequency events and in the event of failures of individual components. To maintain the integrity of the business process, architectural solutions are needed that can guarantee consistency between the cache and the source, regardless of the nature of the load.

The event-driven approach allows you to form cached representations based on a sequence of events that describe changes in the data state. Each event is an independent message with a defined identifier, time stamp, and type of operation, which makes it possible to reproduce the canonical state of the system. This model supports idempotence of event processing and ordering, and also creates conditions for controlling the obsolescence window.

Architectural support includes the use of a shared queue for receiving messages, normalization and deduplication algorithms, transactional state updates, and conflict resolution policies between different sources. Data recovery mechanisms through event log replay and synchronization quality monitoring using observation tools play an important role. This allows integration flows to remain stable even during peak loads.

Thus, event-driven cached data representations with architectural support and consistency guarantees form a holistic approach to building high-load integration systems. The combination of scalability, fault tolerance, and predictability of behavior makes them a suitable tool for modern distributed architectures when performance and data correctness requirements are critical.

Keywords: *event-driven architecture, cached data representations, databases, consistency, integration flows, event log, scaling, observability.*

Стаття надійшла до редакції / Received 03.10.2025
Прийнята до друку / Accepted 15.11.2025

Проблематика та її зв'язок із науковими чи практичними завданнями

У сучасних розподілених інформаційних системах наявне суттєве зростання кількості інтеграційних потоків між сервісами, які обмінюються даними з високою частотою та жорсткими вимогами щодо часових обмежень. У таких сценаріях прямий доступ до джерел істини - сховищ операційних даних - призводить до перевантаження інформаційної система, нестабільності читання та зниження загальної продуктивності в межах визначеного середовища [8]. Як наслідок, виникає потреба в застосуванні альтернативних представленнях даних, щоб зменшити навантаження на джерело істини і забезпечити доступ до інформації.

Однак, застосування кешованих представлень без належного механізму їх актуалізації призводить до проблеми десинхронізації з базовою системою [5]. Зокрема, у випадках високо-частотних змін або відмов в межах інфраструктури, кеш може містити застарілі або некоректні значення, що унеможливлює підтримку цілісності бізнес-процесів. Це породжує потребу в побудові відповідних архітектури, які могли б гарантувати керовану узгодженість між кешованим представленням та джерелом істини [9].

Постає завдання формалізації принципів побудови кешованих подань із заданими обмеженнями на затримку оновлення, правилами обробки подій, ідемпотентністю дій та здатністю до відновлення.

Необхідно створити структуру, що поєднує реактивність, узгодженість, масштабованість і стійкість до відмов, що вирішить питання розподілених обчислень, трансакційної цілісності, організації подієвих потоків та побудови високо-доступних сервісів та джерел інформації.

Сучасний стан досліджень

Проблематика кешування в розподілених системах є об'єктом дослідження протягом останніх двох десятиліть [1, 3]. Найбільш розповюджені рішення включають використання періодичного опитування (polling), реплікованих баз даних, систем на основі Change Data Capture (CDC) та Read Replicas [4, 11]. Усі вищезазначені підходи містять обмеження в контексті високої навантаженості та вимог до узгодженості [9].

Polling-підходи, в більшості випадків, забезпечують просту реалізацію але мають високу затримку між оновленнями та очевидну неефективність використання ресурсів [5]. Вони створюються інтервали неактуальності, в яких кешована інформація не відображає актуальний стан системи [6]. Тим часом, read replicas забезпечують лише часткову масштабованість, не дозволяючи змінювати структуру представлення даних без втрати узгодженості [7].

CDC-підходи (change data capture) дають можливість трансакційного потокового оновлення, але працюють на рівні СУБД, що ускладнює інтерпретацію змін у бізнес-контексті [4]. Крім того, CDC-системи важко адаптуються до складних сценаріїв перетворення або агрегації даних [11]. Таким чином, вище перелічені пункти спонукають зростання застосування підоєво-орієнтованих підходів (event-driven), які дозволяють реєструвати логічні події високого рівня відповідаючи змінам у межах визначених доменних моделей [6].

Під час застосування CQRS (Command Query Responsibility Segregation) наявне поєднання підходу подієво-орієнтованих змін із побудовою окремих моделей читання (read models) [5]. Останні часто реалізуються, як денормалізовані кешовані представлення [2]. Проте в більшості реалізацій CQRS не визначається чітких формальних меж станом кешу та його відхиленням від істини - це ускладнює побудову із передбачуваною поведінкою [9].

Подієві журнали (event logs) також можуть використовуватись для розподілених систем і потокової обробки, як єдине джерело істини, з можливістю відтворення поточного стану системи через послідовне повторне застосування подій. Таким чином забезпечується відновлення кешу при відмовах, хоча це вимагає побудови ідемпотентної функції обробки подій та узгодження порядку дій у розподіленому середовищі за допомогою CRDT чи інших [1, 9].

В більшості існуючих рішень відсутнє формалізоване обґрунтування меж застарілості представлення даних та немає визначених метрик оцінювання узгодженості кешу з джерелом [2, 9]. Також немає визначених архітектурних моделей, які могли б забезпечити відтворюваність, масштабованість і цілісність одночасно - ця прогалина і визначає актуальність дослідження подієво-керованих кешованих подань, як окремого об'єкта проєктування та аналізу [11]. Попри наявність окремих фрагментарних рішень, сучасний стан досліджень не пропонує єдиного цілісного підходу до побудови керованих кешованих представлень, що узгоджуються з джерелом істини за допомогою подієвої моделі в умовах високонавантажених інтеграційних потоків [6, 7].

Формулювання мети та цілей статті

Метою статті є обґрунтування та формалізація архітектурного підходу для побудови подієво-керованих кешованих представлень даних у високонавантажених інтеграційних потоках (integration flows) із гарантованим рівнем узгодженості між кешом та джерелом істини [3]. У межах досягнення цієї мети передбачається створити рішення, що поєднує високу продуктивність, відмовостійкість та контрольовану застарілість даних у системах з асинхронною обробкою та доставкою подій. Особлива увага поділяється тому, щоб запропонований підхід надавав можливість забезпечити передбачувану поведінку системи навіть за умов втрати чи дублюванні подій з врахування випадків настання затримки їх обробки.

Цілями дослідження, по-перше, є формалізація моделі кешованого представлення, як результату застосування послідовності подій, що надходять від зовнішніх систем [2]. По-друге, визначення та обґрунтування допустимого вікна застарілості кешу; розробка відповідних метрик для оцінки його узгодженості з джерелом істини [7]. По-третє, опис вимог до ідемпотентності функцій обробки подій та механізмів стійкості до втрати повідомлень чи повторної доставки повідомлень. Крім того, дослідження

передбачає побудову узагальненої архітектури з реактивним механізмом оновлення подань, розробку відповідних сценаріїв відновлення кешу після збоїв чи затримок. Важливим є опис підходів до спостереження та моніторингу стану узгодженості кешу в режимі реального часу .

Створення формалізованої архітектури подієво-керованих кешованих представлень даних, що здатна функціонувати в умовах високого навантаження та складних інтеграційних сценаріїв, дасть змогу забезпечити стабільну та вимірювану узгодженість даних.

Визначення архітектурної моделі подієво-керованих кешованих представлень даних у високонавантажених інтеграційних потоках

В межах сучасних високонавантажених середовищах критичними визначаються такі параметри, як швидкість та актуальність доступу до даних за допомогою інтеграцій [2, 7]. Програмні системи, що об'єднують декілька джерел інформації, повинні обробляти великі обсяги змін майже в реальному часі, без створення надмірного навантаження на джерела істини [9]. Це вимагає особливої архітектури, здатної одночасно підтримувати узгодженість даних та забезпечувати їхню високу доступність.

В умовах, де класичні методи обробки даних за допомогою прямого доступу до бази даних джерела істини стають вузьким місцем, стає прийнятим застосування концепції кешованих представлень даних [1]. Основна функція таких - надавати готові для використання, оптимізовані під конкретні запити копії інформації, які оновлюються синхронно з подіями у джерелі істини [5].

Принцип подієво-керованого оновлення полягає в тому, що кожна зміна стану даних у джерелі істини генерує подію, яка описує відповідний перелік змін цього стану. Подія є повноцінним повідомленням, що містить унікальний ідентифікатор, часову мітку, тип операції над даними (створення, оновлення, видалення) та повний або частковий опис зміненої одиниці даних [10].

Кожен обробник подій визначається окремим програмним компонентом, що обробляє визначений тип повідомлень [11], чим і визначає себе, як центральною контекстною складовою потоку даних між ключовими компонентами системи, що і продемонстровано на Рисунку 1. Отримавши подію для обробки, він виконує трансформацію даних у формат, придатний для кешованого представлення, після чого записує результат у цільове сховище. Трансформація може включати об'єднання інформації з декількох джерел; агрегацію числових показників чи створення денормалізованих структур.

Рис.1. Організація потоків даних між ключовими компонентами

Щоб підтримувати дані у кеші в актуальному стані, кожне представлення зберігає маркер останньої обробленої події. Це дозволяє у будь який момент визначити ступінь синхронізації з джерелом істини та, під час настання потреби, ініціювати завантаження пропущених змін [8].

Якщо при появі збоїв при оновленні компоненту в кеш потрапляють не усі результати обробки подій - система виконує автоматичне відновлення та перезапис останнього актуального стану даних, використовуючи джерело істинних даних. Воно, також, може базуватись на повторному відтворенні історії подій, збережених в журналі брокера повідомлень. Такі механізми дозволяють відновити дані без втручання користувачів та без зупинки основних інтеграційних потоків [11].

Особливе місце в такій архітектурі займає контроль застарілості кешу. Якщо маркер конкретної останньої події у кеші суттєво відстає від маркера у брокері, система позначає представлення як неактуальне та ініціює його синхронізацію [8]. Це гарантує, що інтеграційні потоки завжди працюють з максимально свіжими даними .

Під час обробки змін однієї сутності, коли події надходять з кількох джерел, узгодженість досягається за допомогою використання логічних часових міток або пріоритетності джерел. Таким чином уникається настання суперечливих оновлень, зберігається хронологічна та структурна цілісність інформації.

Додатковою характеристикою, яку можна визначити за допомогою трансакцій є атомарність оновлень у кеші. Якщо подія стосується кількох взаємопов'язаних сутностей, оновлення обробляються як єдиний блок використовуючи версіонування запису. Це запобігає появі частково оновлених станів, які могли б призвести до логічних помилок в роботі інтеграцій [4]. Додатково, еволюція схеми представлень, також, повинна використовувати версіонування задля запуску нових форматів паралельно зі старими, поступово переводячи інтеграційні потоки на оновлені структури без простої і ризику втрати даних [10].

В такій архітектурі наявні два місця для потенційного масштабування: додавання нових обробників для кожного нового переліку подій та горизонтальне масштабування самого кеш-сховища – зображено на Рисунку 2. В обох випадках зберігається принцип рівномірного розподілу навантаження між усіма компонентами. Реплікація критичних елементів систем забезпечується безперервна робота архітектури у разі відмови частини інфраструктури [9]. Брокер подій та кешовані сховища, після налаштування, можуть функціонувати в кластерному режимі, що в результаті дозволяє продовжувати обробку запитів навіть при відмові частини вузлів.

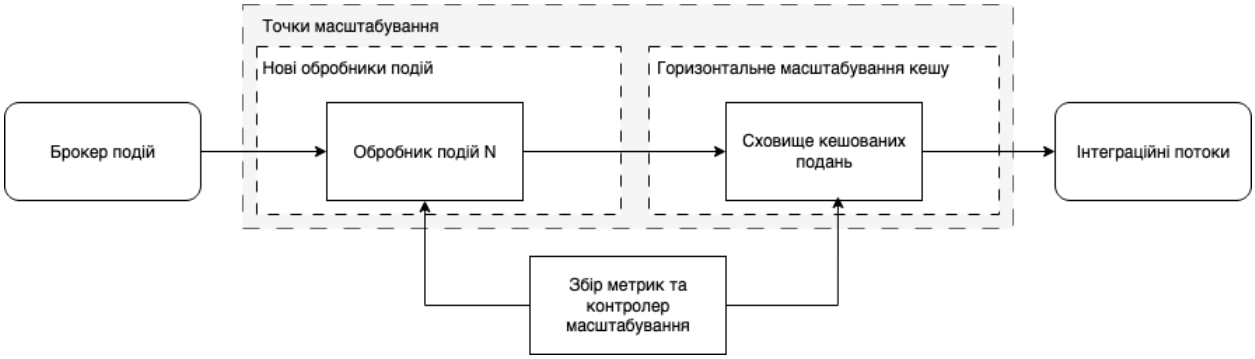

Рис. 2. Визначення точок масштабування

Моніторинг помилок дозволяє зберегти перелік частин даних, які могли бути пошкодженні, та виконати логіку відновлення, щоб, у результаті, встановити актуальний стан визначених сутностей. За допомогою кількісних ознак та їх інтервального періодичного співставлення можна визначати моменти для масштабування, щоб передбачити настання моментів критичності [7].

У межах високонавантажених інтеграційних сценаріїв подієво-керовані кешовані представлення стають центральним елементом архітектури. Вони забезпечують узгодженість даних, зменшують затримку доступу та мінімізують вплив інтеграційних потоків на джерела істини. Це дозволяє масштабувати систему без шкоди для продуктивності та забезпечити передбачувану поведінку навіть у пікові періоди навантаження, що робить його ключовим інструментом для сучасних розподілених систем.

Алгоритм обробки подій змін даних у подієво-керованій архітектурі

У подієво-керованих інтеграційних системах зміни даних часто надходять одночасно з декількох незалежних джерел. Це створює потребу в уніфікованій логіці обробки подій, яка дозволяє зберігати послідовність, узгодженість і цілісність стану [3]. Для цього всі повідомлення спочатку потраплять у спільну чергу, при обробці якої, отримують стандартизоване представлення. Кожна подія містить ідентифікатор запису сутності, тип операції (створення, оновлення, видалення), часову мітку чи версію запису, ідентифікатор джерела та корисні дані. Наочно це, в межах послідовності обробки подій, можна побачити на Рисунку 3. Прийняте повідомлення проходить валідацію схеми та перевірку полів, і лише валідні події потрапляють до подальшої обробки. Зазвичай не валідними вважають ті події, обробка яких не підтримується системою чи подія містить не достатню кількість даних для подальшої обробки [11].

Далі, подія проходить нормалізацію: поля форматуються до канонічного вигляду для внутрішньої системи, Часові мітки перетворюються до єдиного стандарту, а значення ключів уніфікуються. На цьому ж етапі виконується уніфікація за унікальним ключем події для запобігання повторної подальшої обробки.

Після цього, відбувається маршрутизація події у послідовність сутності - сховище, де усі події з однаковим entityId підлягають упорядкуванню обробляються серійно [5]. Для цього можуть бути застосовані різні методи, найчастіше event sourcing. Упорядкування виконується за допомогою часових міток чи версіонування документу.

До внесення змін потрібно перевірити стан кешу разом із метаданими. Якщо запис відсутній, потрібно ініціалізувати запис, якщо запис застарілий - запис помічається як застарілий.

Для оновлення запису порівнюються версії чи часові позначки. Якщо подія, що обробляється, новіша - застосовується оновлення. Перезаписуються лише поля, що містяться у корисному навантаженні, після чого перераховуються похідні поля, агрегати та індекси [7]. Якщо оновлення часткове і вимагає знання попереднього значення, використовується трансакційне оновлення для запобігання одночасного запису.

Події видалення відбуваються відповідно до визначеної доменної політики: м'яке видалення, встановлення нової версії з пустим значенням чи почне видалення з відповідних індексів та сховищ.

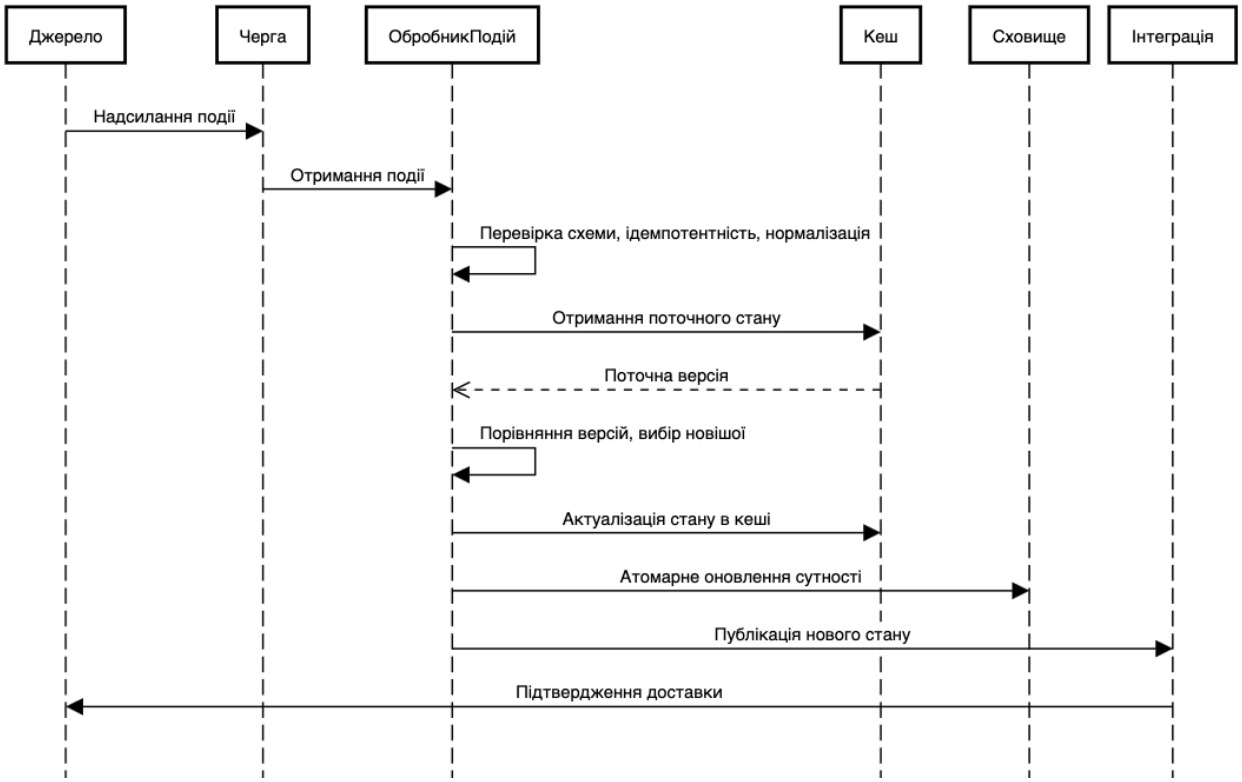

Рис. 3. Послідовність обробки події в інтеграційній системі

У результаті застосування будь-якої вищезгаданої події формується новий канонічний стан сутності, оновлюється версія чи часова позначка, перераховуються денормалізовані проекції і оновлюються наявні ключі доступу до актуального запису. Усі зміни виконуються в межах атомарної трансакції сховища кешу, щоб унеможливити часткові оновлення [9].

Щоб запобігти настанню паралельної обробки, встановлюється серіалізація за ключем для кожного запису під час його модифікації. При запиті обробки за однаковим ключем, лише один збереже зміни, усі інші - отримають відмову і виконають повтор із повторним читанням щойно актуалізованого стану.

У випадку конфліктів появи між джерелами, коли стани відповідних записів не співпадають, слід ввести поняття політики розв'язання конфліктів відповідно до покажчику найпізнішої версії, чи пріоритетності джерела оновлення [1]. Обрана політика має бути реалізована у вигляді умови (детермінованого компаратора), який визначатиме відповідне рішення в межах реального виконання.

Після фіксації нового стану відбувається публікація повідомлення про утворення нового стану запису, щоб усі залежні чи підпорядковані системи-споживачі внесли відповідні зміни станів для свого контексту [2]. Таке повідомлення надсилається, також, може надсилатись у вихідний інтеграційний канал для подальшої синхронізації зовнішніх систем.

Кожна оброблена подія завершується підтвердженням доставки за допомогою повернення позитивного статусу доставки події в точку початкової відправки повідомлення. Якщо операція не відбулась чи надсилання перервалось, подія не підтверджується, та відповідно обробляється повторно за допомогою відокремленої повторної черги (dead-letter queue) чи за допомогою внутрішніх компонентів обробника [6]. Ідемпотентність забезпечується перевіркою стану обробки унікального ключа події.

Усі помилкові події, що були відправленні до dead-letter queue - канал із повним контекстом про набір вхідних даних, стан до та після, а також причину помилки. Визначений відповідний процес повинен обробляти цю чергу відновлювати події після виправлення причин помилок, або ініціювати виклик компенсуючих дій.

Для гарантій відновлюваності існує журнал застосованих подій на рівні сутностей. Таким чином зберігаються ідентифікатори останніх прийнятих подій і контрольні стани чи суми станів [1, 7]. Під час реконструкції актуального стану кешу відбувається послідвне програвання подій починаючи з останнього зафіксованого маркера, що забезпечує детерміноване відтворення.

Завершальним кроком є створення умов спостережності: логування кожної стадії (отримання, нормалізація, серіалізація, застосування даних та публікація нового стану) обробки даних і подій над ними з кореляційними ідентифікаторами [5]. Важливим є визначення метрик часу обробки подій, обрахунок процентилів відхилених подій, частоти конфліктів і довжини черги - усе це дозволяє автоматично масштабувати обробники подій та тримати вікно застарілості в межах очікуваних показників.

Підсумкові положення та їх застосування в високонавантажених системах

Архітектурна модель подієво-керованих кешованих представлень даних забезпечує можливість відокремлення контуру читання від джерел істини та створення оптимізованих подань, що витримають високе

навантаження. Реалізація через подієвий механізм оновлення дозволяє підтримувати актуальний стан даних без надмірного звернення до операційних сховищ.

Узгодженість досягається завдяки визначенню вікна застарілості, ідемпотентній обробці подій, використанню механізмів упорядкування та розв'язання конфліктів. Відновлюваність системи забезпечується журналом подій та можливістю повторного відтворення змін, що унеможливлює втрату інформації навіть у випадках відмов.

Архітектурне забезпечення включає масштабування обробників і сховищ, кластеризацію брокера подій, підтримку трансакційної цілісності під час оновлення; засоби спостережності, що дозволяються контролювати якість синхронізації у реальному часі. Таким чином досягається поєднання високої продуктивності, стабільності та передбачуваної поведінки на рівні інтеграційних потоків.

Подієво-керовані кешовані представлення даних із архітектурним забезпечення та гарантіями узгодженості формують цілісних підхід до побудови високонавантажених інтеграційних систем, у яких забезпечення швидкодії неможливе без одночасного контролю вірності та відтворюваності стану.

Література

1. Д'яков С., Зубрей Т., Самоїдюк А. Застосування джерел подій і шаблонів CQRS в розподілених системах. Адаптивні системи автоматичного управління (ASAC), 2019, Т. 1(34). doi:10.20535/1560-8956.1.2019.178224.

2. Завалій Т. І. Задачі та алгоритми опрацювання потокових даних. Вісник Хмельницького національного університету. Технічні науки, 2023, Ч. 2, № 5(327): 42–48. doi:10.31891/2307-5732-2023-327-5-42-42.

3. Копанський І., Григор'єв М. Подієві журнали як засіб забезпечення відновлюваності станів у розподілених системах. Наукові вісті НТУУ «КПІ», 2023, Т. 5, № 2: 65–74. doi:10.20535/nvki2023.5(2).2331.

4. Олійник Д. А. Система аналізу текстових потоків даних. Прикладні питання математичного моделювання, 2020, № 1–3. doi:10.32782/2618-0340/2020.1-3.15.

5. Ружинський В. Г., Бондарчук А. П., Кузьміних В. О., Отрох С. І., Тараненко Р. А. Подійно-орієнтована мікросервісна архітектура системи збору та обробки бібліографічної інформації. Телекомунікаційні та інформаційні технології, 2022, № 4. doi:10.31673/2412-4338.2022.049098.

6. Chen Y., Ramasamy K. Потокова обробка: концепції та застосування в аналітиці реального часу. – Springer, 2020. – 310 с. doi:10.1007/978-3-030-49424-7.

7. Fragkoulis M., Carbone P., Kalavri V., Katsifodimos A. Огляд еволюції систем потокової обробки даних. The VLDB Journal, 2024, 33: 507–541. doi:10.1007/s00778-023-00819-8.

8. Helland P. Незмінність усе змінює. Communications of the ACM, 2016, 59(1): 64–70. doi:10.1145/2844112.

9. Overeem M., Spoor M., Jansen S., Brinkkemper S. Емпірична характеристика систем із event sourcing та їх еволюції схем. Journal of Systems and Software, 2021, 178: 110970. doi:10.1016/j.jss.2021.110970.

10. Vinh N. T. Q., Phuong L. H., Linh T. N. Рішення для синхронного інкрементального супровадження матеріалізованих подань на основі рекурсивних SQL-запитів. Eastern-European Journal of Enterprise Technologies, 2019, 2/6(98): 26–33. doi:10.15587/1729-4061.2019.180226.

11. Winter C., Schmidt T., Neumann T., Kemper A. Розділене супровадження безперервних подань. Proceedings of the VLDB Endowment (PVLDB), 2020, 13(11): 2620–2633. doi:10.14778/3407790.3407849.

References

1. Diakov, S., Zubrei, T., & Samoidiuk, A. (2019). Applying event sourcing and CQRS patterns in distributed systems. Adaptive Systems of Automatic Control (ASAC), 1(34). doi:10.20535/1560-8956.1.2019.178224.

2. Zavaliĭ, T. I. (2023). Tasks and algorithms for stream data processing. Herald of Khmelnytskyi National University. Technical Sciences, Part 2, 5(327), 42–48. doi:10.31891/2307-5732-2023-327-5-42-42.

3. Kopanskyi, I., & Hryhoriev, M. (2023). Event logs as a means of ensuring state recoverability in distributed systems. Scientific News of NTUU "KPI", 5(2), 65–74. doi:10.20535/nvki2023.5(2).2331.

4. Oliinyk, D. A. (2020). A system for text stream data analysis. Applied Issues of Mathematical Modelling, 1–3. doi:10.32782/2618-0340/2020.1-3.15.

5. Ruzhynskyi, V. H., Bondarchuk, A. P., Kuzminykh, V. O., Otrokh, S. I., & Taranenko, R. A. (2022). Event-oriented microservice architecture for collecting and processing bibliographic information. Telecommunication and Information Technologies, (4). doi:10.31673/2412-4338.2022.049098.

6. Chen, Y., & Ramasamy, K. (2020). Stream processing: Concepts and applications in real-time analytics. Springer. doi:10.1007/978-3-030-49424-7.

7. Fragkoulis, M., Carbone, P., Kalavri, V., & Katsifodimos, A. (2024). A survey on the evolution of stream processing systems. The VLDB Journal, 33, 507–541. doi:10.1007/s00778-023-00819-8.

8. Helland, P. (2016). Immutability changes everything. Communications of the ACM, 59(1), 64–70. doi:10.1145/2844112.

9. Overeem, M., Spoor, M., Jansen, S., & Brinkkemper, S. (2021). An empirical characterization of event sourced systems and their schema evolution—Lessons from industry. Journal of Systems and Software, 178, 110970. doi:10.1016/j.jss.2021.110970.

10. Vinh, N. T. Q., Phuong, L. H., & Linh, T. N. (2019). A solution for synchronous incremental maintenance of materialized views based on SQL recursive query. Eastern-European Journal of Enterprise Technologies, 2/6(98), 26–33. doi:10.15587/1729-4061.2019.180226.

11. Winter, C., Schmidt, T., Neumann, T., & Kemper, A. (2020). Meet me halfway: Split maintenance of continuous views. Proceedings of the VLDB Endowment (PVLDB), 13(11), 2620–2633. doi:10.14778/3407790.3407849.