DOI 10.34185/1562–9945–5–166–2026–05
УДК 004.415.3:681.6

В.В. Єрмаков, Б.І. Мороз

МОДЕЛЮВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ПРІОРИТЕТНОЇ ОБРОБКИ ПОВІДОМЛЕНЬ В OLAP–СИСТЕМАХ ЗА УМОВ ПІКОВИХ НАВАНТАЖЕНЬ

Анотація. Запропоновано модель пріоритетної обробки повідомлень в OLAP–системах за пікових навантажень. Метод поєднує цінність, терміновість і вартість обробки та зменшує втрати від запізнення.

Єрмаков В. В., Мороз Б. І. Моделювання та експериментальна оцінка пріоритетної обробки повідомлень в OLAP–системах за умов пікових навантажень. У статті розглянуто задачу експериментальної перевірки керованої дисципліни обробки повідомлень в OLAP–системах, для яких критичними є цінність інформації, її старіння та обмеженість ресурсів аналітичного конвеєра. Аналіз останніх досліджень і публікацій показав, що потокова обробка даних, класичне планування та внутрішня організація аналітичних сховищ переважно вивчаються окремо, без єдиного контуру керування. Мета дослідження – розробити й експериментально оцінити керовану дисципліну пріоритетної обробки, яка поєднує бізнес–цінність повідомлень із поточним технічним станом сховища. Запропоновано модель OLAP–конвеєра, що охоплює broker, ETL, micro–batching, запис у сховище, побудову матеріалізованих подань і фонові операції злиття. Для оцінювання втрат від несвоєчасної доставки даних використано критерій зваженого запізнення, а для вибору наступного мікропакета – інтегральну скорингову функцію, яка поєднує очікуваний вплив на аналітичні рішення, терміновість та вартість обробки. Основний експеримент виконано для чотирьох сценаріїв нерівномірного потоку подій (Uniform, Short peak, Long peak, Mixed load) за профілем medium SLO і capacity = 4.0; додатково проведено робастнісні перевірки для різних рівнів пропускної здатності та профілів SLO. У середньому запропонований метод забезпечив найменше значення агрегованого критерію втрат J = 351,231 млн ум. од., що на 2,5 % краще порівняно зі статичними пріоритетами, на 22,2 % – порівняно з EDF і на 26,5 % – порівняно з FIFO. Також отримано 98,4 % виконання HP SLO та 83,5 % виконання All SLO. Результати підтверджують доцільність міжрівневого керування обробкою повідомлень в OLAP–системах у режимах пікового навантаження.

Ключові слова: OLAP–система, аналітичний конвеєр, micro–batching, пріоритезація, зважене запізнення, SLO, машинне навчання, моделювання, експериментальна оцінка.

© Єрмаков В.В., Мороз Б.І., 2026
Copyright for this paper by its authors. Use permitted under License CC BY 4.0.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Постановка проблеми. У сучасних OLAP–системах факти надходять до аналітичного сховища через багатокроковий конвеєр обробки, який охоплює брокер повідомлень, процеси витягу, перетворення та завантаження даних (Extract–Transform–Load, ETL), формування мікропакетів, запис у таблиці сховища, побудову матеріалізованих подань і аналітичних кубів [4, 5, 19, 22–24]. За стабільного навантаження такий конвеєр забезпечує прийнятну актуальність даних, однак під час коротких піків або тривалих хвиль перевантаження ресурси CPU, I/O та фонових злиттів стають обмежувальним чинником [11, 21, 27, 28].

Практична проблема полягає в тому, що не всі повідомлення мають однакову оперативну цінність. Події про платежі, антифрод–сигнали, обмежувальні показники A/B–тестів або критичні KPI мають вищий пріоритет, ніж детальна телеметрія чи службові журнали. Крім того, корисність даних зменшується зі зростанням затримки їх потрапляння до сховища, тобто інформація старіє [1–3]. Тому для OLAP–систем важливо не лише зберегти потік фактів, а й визначити раціональну черговість їх обробки.

Таке завдання має безпосередній зв’язок з практичними задачами забезпечення актуальності дашбордів, вчасного спрацювання сигналів тривоги, стабільності побудови кубів і керування навантаженням на сховище. З наукової точки зору воно поєднує проблематику потокової обробки даних, теорії планування, цінності та старіння інформації, а також адаптивного вибору пріоритетів на основі моделей машинного навчання [6, 12, 14, 15].

Аналіз останніх досліджень і публікацій. У роботах із потокової обробки даних досліджено архітектури управління потоками подій, механізми load shedding, адаптивну маршрутизацію та способи зменшення затримок у stream–processing системах [4, 5, 13, 27, 28]. Ці підходи добре описують поведінку потокових рушіїв, однак не охоплюють повною мірою проблеми, специфічні для OLAP–сховищ, де на продуктивність істотно впливають кількість частин даних, write amplification і черги фонових злиттів [11, 21, 22].

Класична теорія планування пропонує багатий апарат для мінімізації tardiness, weighted tardiness, lateness та інших часових втрат [6, 7, 12]. Ці результати придатні для формального опису втрат від запізнення, але зазвичай орієнтовані на абстрактні завдання або виробничі операції й не враховують міжрівневу природу OLAP–конвеєра.

Ідеї цінності та старіння інформації, розвинуті у працях В. Свірідова та Б. Мороза, дають концептуальну основу для визначення корисності даних у часі [1, 2]. У більш сучасних роботах близьку проблематику досліджено через часову актуальність статусної інформації [3]. Для аналітичних сховищ важливими є праці з реалізації кубів, матеріалізованих подань і інкрементального оновлення похідних структур [23–26].

Окремий напрям становлять роботи з learning–to–rank, градієнтного бустингу й адаптивного розподілу ресурсів, у яких оцінка об’єктів на основі множини ознак використовується для побудови правил ранжування або вибору політики [8, 14, 15]. Такі моделі створюють передумови для переходу від статичних класів пріоритету до дано–орієнтованого вибору черги.

Виділення невирішених раніше частин загальної проблеми. Попри значну кількість досліджень, більшість із них аналізують або потокову обробку, або класичне планування, або внутрішню організацію сховища даних окремо. Недостатньо розробленими залишаються підходи, у яких broker, ETL, micro–batching, порядок запису в OLAP–сховище та фонові злиття розглядаються як єдиний контур керування з обмеженими ресурсами [11, 21, 22, 27, 28].

Також бракує експериментальних робіт, у яких критерії цінності та старіння інформації були б безпосередньо поєднані з адаптивною score–моделлю вибору мікропакетів і перевірені на типових сценаріях пікового навантаження. Залишається відкритим питання, чи забезпечує така політика реальну перевагу над FIFO, статичними пріоритетами та EDF за інтегральним критерієм втрат і системними метриками стану сховища.

Мета дослідження. Метою дослідження є розробка та експериментальна оцінка керованої дисципліни обробки повідомлень в OLAP–системах за умов пікових навантажень, яка враховує цінність інформації, її старіння, дедлайни, вартість обробки та поточний стан аналітичного сховища.

Для досягнення поставленої мети необхідно: побудувати модель експериментального OLAP–конвеєра; задати класи повідомлень і сценарії нерівномірного потоку подій; реалізувати базові політики планування; сформувати score–модель вибору мікропакетів; визначити систему метрик оцінювання; провести порівняльне моделювання та інтерпретувати отримані результати.

Викладення основного матеріалу дослідження. Архітектура експериментального стенда. Експериментальний стенд моделює типовий OLAP–конвеєр, у якому джерела подій передають повідомлення до broker, після чого ETL–модуль формує мікропакети, що записуються в аналітичне сховище. Далі оновлюються матеріалізовані подання, розділи кубів і похідні представлення, а фонові процеси виконують злиття частин даних. На рис. 1 подано структурну схему досліджуваного конвеєра.

У межах моделювання контролер спостерігає за станом черг, consumer lag, накопиченим backlog, merge debt, insert latency та поточним ризиком порушення SLO. Це дає змогу розглядати обробку повідомлень не як локальну чергу, а як міжрівневу систему зворотного зв’язку.

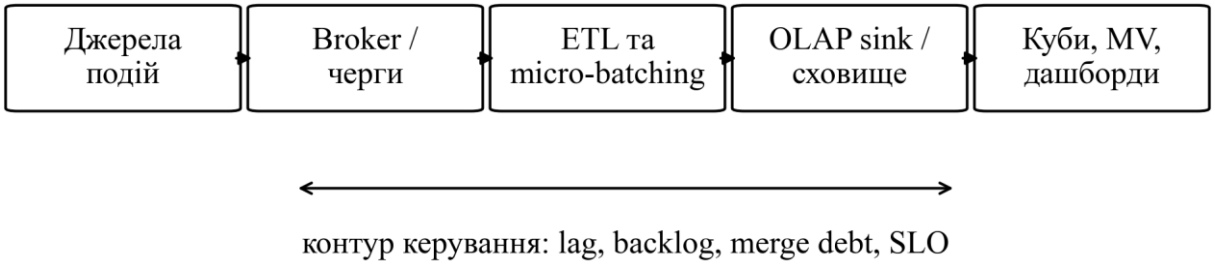


Рисунок 1 – Структурна схема експериментального OLAP–конвеєра

Сценарії потоку повідомлень та параметри моделювання. У дослідженні розглядалися три класи повідомлень: високий пріоритет (HP), середній пріоритет (MP) та низький пріоритет (LP). До HP віднесено платежі, антифрод–події, guardrail–сигнали

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

A/B–тестів і критичні KPI; до MP – near–real–time агрегати та проміжні показники; до LP – детальна телеметрія та службові журнали. Вхідний потік генерувався як нерівномірний процес із двома піками навантаження протягом добового інтервалу. Основний експеримент охоплював чотири сценарії навантаження: Uniform, Short peak, Long peak і Mixed load; додатково виконано робастнісні серії зі зміною capacity та профілів SLO.

Для кожного повідомлення задавалися момент надходження, базовий час обробки, дедлайн, вага цінності та додаткові ознаки попиту на дані. У процесі моделювання враховувався вплив фонових злиттів на ефективний час сервісу та, навпаки, вплив оброблених повідомлень на накопичення merge debt.

Для формування навантаження у notebook–експерименті використано вибірку Google Borg 2019 cluster sample from Kaggle. Після очищення та нормалізації даних сформовано 50 000 валідних задач із рівномірним розподілом пріоритетних ваг 1–5 (по 10 000 задач у кожному квантилі). На цій основі побудовано чотири сценарії потоку – Uniform, Short peak, Long peak і Mixed load – по 20 000 задач у кожному. У повній реалізації notebook додатково аналізувалися SJF, WSPT і Random, однак у табл. 3 для стислості залишено чотири основні політики: FIFO, STATIC, EDF і SCORE. У роботі Google Borg trace використовується не як прямий журнал OLAP–системи, а як основа для trace–driven побудови dummy–навантаження, де задачі інтерпретуються як умовні повідомлення аналітичного конвеєра з різною цінністю, тривалістю обробки та часовою чутливістю.

Таблиця 1

Базові параметри класів повідомлень у моделюванні

Клас	**Приклади повідомлень**	**Середня вага w**	**Дедлайн, хв**	**Сер. базовий час сервісу, хв**
HP	Платежі, anti–fraud, A/B guardrails	12	8	0,75
MP	Near–real–time KPI, retention, revenue snap-shots	4	30	1,00
LP	Телеметрія, de-bug–логи, low–priority raw events	1	140	1,25

Процедура формування експериментального навантаження з трасування. Процедура формування експериментального навантаження починалася з автоматичного

завантаження вибірки Google Borg 2019 cluster sample from Kaggle та вибору найбільш повного файлу трасування у форматі CSV/CSV.GZ/Parquet. Далі виконувалася уніфікація назв полів, після чого з сирих записів формувався набір задач для моделювання: момент надходження повідомлення задавався за часовою позначкою трасування, тривалість сервісу визначалася як різниця між часом завершення та початку задачі, а ідентифікатор формувався на основі службових полів колекції та екземпляра. Із подальшого аналізу виключалися записи з пропущеними значеннями, нечисловими параметрами, нульовою або від'ємною тривалістю виконання, а також з ознаками неуспішного завершення.

Після очищення часову вісь нормалізували відносно мінімального arrival_time, а для зменшення зміщення вибірки застосовували стратифіковане відбирання по часових кошиках. У підсумку отримано 50 000 валідних задач, для яких пріоритети з трасування було перетворено на п'ять рівнів ваг 1–5 за квантилями. Верхній квантиль інтерпретувався як високопріоритетний клас, що дало змогу перейти від сирих системних priority_raw до експериментально керованих бізнес–ваг w_i і надалі узгодити трасування з моделлю цінності та старіння інформації.

Генерування сценаріїв та алгоритм імітаційного моделювання. Сценарії Uniform, Short peak, Long peak і Mixed load формувалися з профілю інтенсивності надходження задач у часових кошиках. Для цього обчислювалися локальні значення arrival rate, ковзні середні та локальна варіативність. Сценарій Uniform відповідав ділянці з навантаженням, близьким до медіанного, та малою дисперсією; Short peak – інтервалу з найбільшим коротким локальним сплеском; Long peak – інтервалу з найбільшим значенням ковзного середнього; Mixed load – ділянці з найбільшою локальною варіативністю. Для кожного сценарію відбиралося по 20 000 задач, а робастнісні серії будувалися шляхом повторного прогону тих самих сценаріїв за різних значень capacity і профілів SLO.

Імітаційне моделювання реалізовано як подієвий цикл із чергою готових задач. Після надходження до ready–queue кожний елемент отримував дедлайн, вагу цінності та допоміжні ознаки. На кожному кроці часу симулятор додавав у чергу всі задачі, момент надходження яких не перевищував поточний час, після чого політика планування вибирала наступний мікропакет для обслуговування. Для вибраної задачі обчислювалися waiting time, flow time, tardiness, weighted tardiness, ознака виконання SLO та оновлений стан merge debt. Саме така схема забезпечує безпосередній перехід від трасування до відтворюваного policy–based експерименту.

Алгоритм 1. Псевдокод імітаційного симулятора

Вхід: tasks, відсортовані за arrival_time; політика P; пропускна здатність c; профіль SLO s.

1. Ініціалізувати поточний час t першим arrival_time, ready_queue = ∅, metrics = ∅.
2. Поки існують необроблені задачі або ready_queue не порожня:
3. Додати до ready_queue всі задачі, для яких arrival_time ≤ t.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

4. Якщо ready_queue порожня, то покласти t = next arrival_time і перейти до кроку 2.

5. Для кожної задачі i у ready_queue обчислити slack_i = d_i – t, cost_i та службові ознаки x_i.

6. Якщо P = SCORE, обчислити score_i за формулою (3); інакше застосувати правило FIFO / STATIC / EDF.

7. Вибрати задачу j відповідно до політики P та визначити start_j = t.

8. Обчислити service_j = runtime_j / c + merge_penalty_j, finish_j = start_j + service_j.

9. Зафіксувати waiting_j, flow_j, tardiness_j, weighted_tardiness_j та SLO_hit_j.

10. Оновити merge debt, видалити j з ready_queue та покласти t = finish_j.

11. Після завершення циклу агрегувати J, HP SLO, All SLO, mean waiting time і mean flow time.

Таблиця 2

Параметри експериментального плану та перевірок

Параметр	**Значення / конфігурація**
Пропускна здатність (capacity)	1.0, 2.0, 4.0, 8.0; основний режим статті – capacity = 4.0.
Профілі SLO	strict, medium, relaxed; основний режим статті – medium SLO.
Абляційні варіанти	priority_only; urgency_only; priority_urgency; priority_urgency_cost; full.
Обсяг експерименту	50 000 очищених задач у глобальному пулі; 4 базові сценарії (Uniform, Short peak, Long peak, Mixed load) по 20 000 задач у кожному; окремі робастнісні sweeps для capacity та SLO.

Критерій оцінювання втрат від несвоєчасної обробки. Як базовий критерій ефективності використано модель зваженого запізнення, адаптовану до OLAP–контексту. Для i–го повідомлення або мікропакета вартість запізнення визначається за формулою:

$$C_i = w_i \cdot \max(0, t_i - d_i)^{\alpha}, \alpha \geq 1 \quad (1)$$

де w_i – бізнес–вага повідомлення; d_i – дедлайн; t_i – момент commit у сховище; α – параметр чутливості до запізнення. Агрегований критерій втрат для всієї політики планування задається виразом:

$$J = \sum_i C_i \rightarrow \min \quad (2)$$

Таке формулювання спирається на weighted tardiness–підхід із теорії планування [6, 7, 12] і на концепцію цінності та старіння інформації [1–3]. На відміну від класичних задач, момент завершення тут інтерпретується як фіксація даних у OLAP–сховищі, оскільки саме після commit факт стає доступним для кубів, матеріалізованих подань і дашбордів.

Політики планування, що порівнювалися. Для порівняльної оцінки реалізовано чотири політики обробки. Політика FIFO вибирає повідомлення в порядку їх надходження. Політика статичних пріоритетів обслуговує класи HP > MP > LP у фіксованому порядку. Політика EDF (earliest deadline first) вибирає повідомлення з найменшим абсолютним дедлайном. Запропонована політика SCORE обчислює інтегральну оцінку кожного готового мікропакета з урахуванням його очікуваного впливу, терміновості та вартості обробки.

Щоб запобігти голодуванню низькопріоритетних потоків, у SCORE–політиці використано прості guardrails: максимальний припустимий час очікування для кожного класу, а також додатковий множник справедливості, що зростає з часом перебування пакета в черзі.

Скорингова модель вибору мікропакетів. Для online–вибору наступного мікропакета використовувалася скорингова функція:

$$\text{score}_i = \frac{f_\theta(x_i) \cdot g(\text{slack}_i)}{\max(\varepsilon, \text{cost}_i)} \qquad (3)$$

У формулі (3) f_θ(x_i) інтерпретується як оцінка очікуваного впливу мікропакета на аналітичні рішення; g(slack_i) – функція терміновості, що зростає зі зменшенням запасу часу до дедлайну; cost_i – інтегральна вартість обробки з урахуванням базового часу сервісу та поточного merge debt; ε – малий стабілізувальний параметр. Ця модель не є дослівним запозиченням, а побудована як синтез ідей пріоритетного планування, learning–to–rank та cost–aware відбору [7, 12, 14, 15].

Вектор ознак x_i містив характеристики критичності куба, поточного попиту на дані, стану broker, стану сховища й ресурсоємності самого пакета. Таким чином, пріоритет визначався не лише класом події, а й контекстом її використання.

Таблиця 3

Узагальнені результати моделювання для чотирьох основних політик (середні значення для сценаріїв Uniform, Short peak, Long peak, Mixed load; medium SLO, capacity = 4.0)

Політика	J, млн ум. од.	HP SLO, %	All SLO, %	Сер. час очікування	Сер. flow time
FIFO	477,988	76,5	75,2	8786,0	8836,2
STATIC	360,248	98,6	81,7	8727,4	8777,6
EDF	451,250	79,9	75,5	8823,0	8873,2
SCORE	351,231	98,4	83,5	8364,0	8414,3

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Результати експериментальної перевірки та їх обговорення. Усереднені результати моделювання для сценаріїв Uniform, Short peak, Long peak і Mixed load наведено в табл. 3. Для основного режиму notebook–експерименту (medium SLO, capacity = 4.0) найгіршою за інтегральним критерієм J залишилася політика FIFO, оскільки вона не диференціює чергу за цінністю, терміновістю та вартістю обробки.

Запропонована score–політика забезпечила найменше середнє значення J = 351,231 млн ум. од. Порівняно з FIFO (477,988 млн) це відповідає зменшенню інтегральних втрат на 26,5 %, порівняно з EDF (451,250 млн) – на 22,2 %, а порівняно зі STATIC (360,248 млн) – на 2,5 %.

За середньою часткою виконання HP SLO score–політика забезпечила 98,4 %, що практично відповідає STATIC (98,6 %) і суттєво перевищує EDF (79,9 %) та FIFO (76,5 %). Водночас за інтегральним показником All SLO вона також показала найкраще значення – 83,5 %.

Найвиразніше перевагу запропонованої score–політики видно на рис. 3, де показано частку виконання HP SLO для різних сценаріїв навантаження. Саме цей графік безпосередньо підтверджує, що метод утримує якість обслуговування критичних повідомлень на рівні, близькому до STATIC, але без переходу до настільки ж жорсткої схеми пріоритетів. Натомість рис. 4 і рис. 5 слугують передусім для ілюстрації часової динаміки backlog і processing debt у сценарії Short peak; вони показують характер перебігу навантаження, але не є основним джерелом висновку про перевагу політик.

На рис. 2–5 наведено графіки, сформовані безпосередньо за результатами notebook–моделі. На відміну від табл. 3, де для стислості залишено чотири основні політики, у діаграмах також показано SJF, WSPT і Random як довідкові baseline. Це дає змогу побачити не лише середні підсумки, а й сценарну чутливість методу та часову динаміку backlog і processing debt.

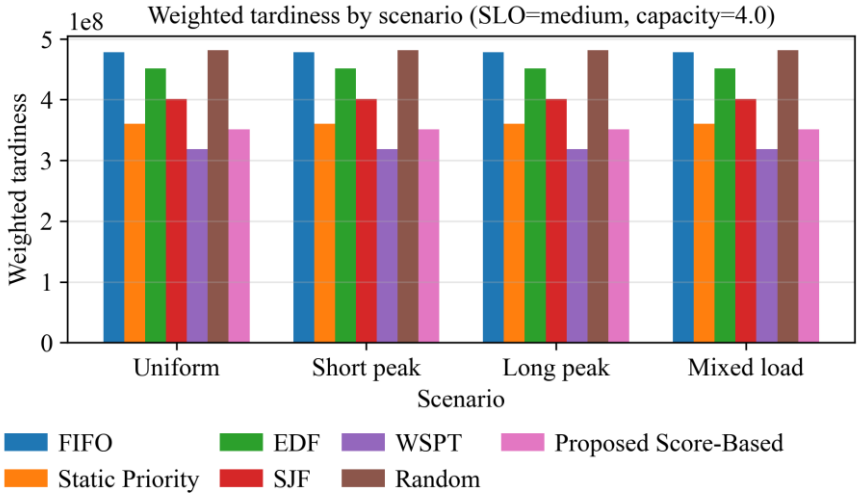


Рисунок 2 – Порівняння зваженого запізнення для сценаріїв навантаження

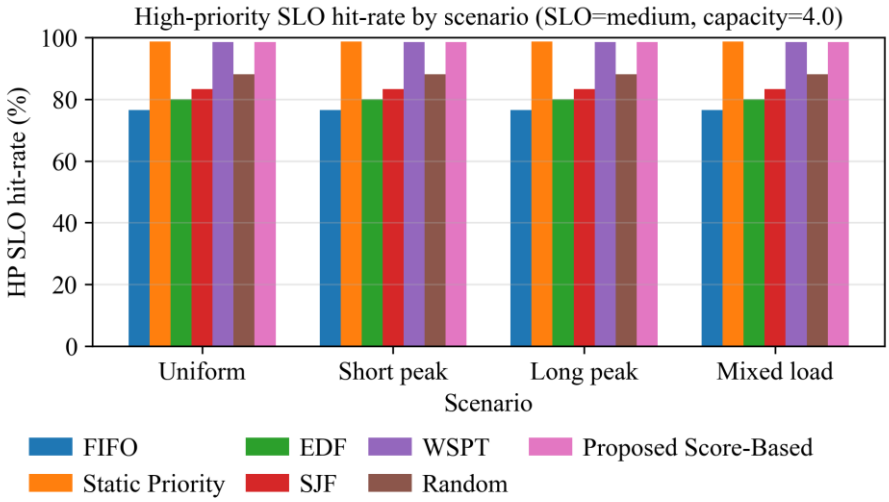


Рисунок 3 – Частка виконання HP SLO для сценаріїв навантаження

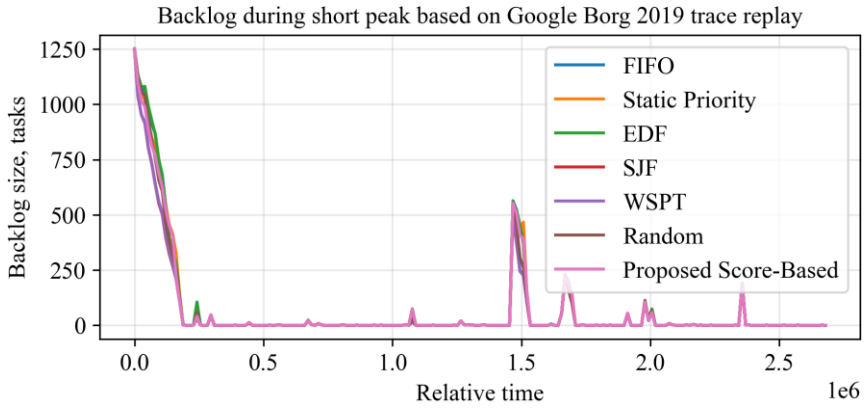


Рисунок 4 – Динаміка backlog у сценарії Short peak

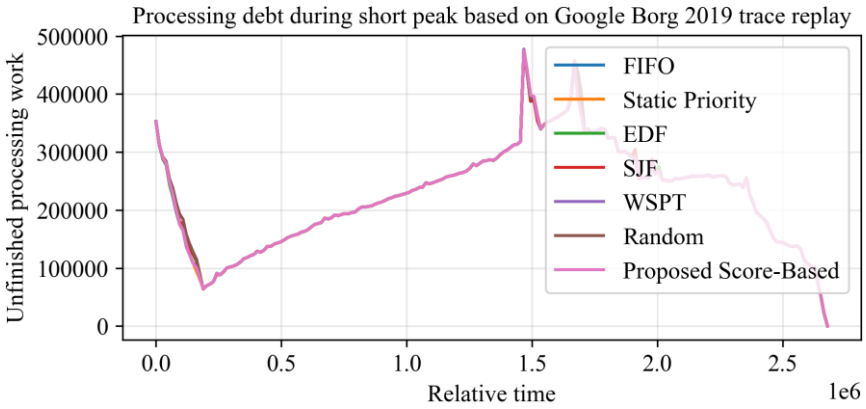


Рисунок 5 – Динаміка processing debt у сценарії Short peak

Отже, за результатами моделювання підтверджено, що міжрівнева керована дисципліна обслуговування здатна зменшувати втрати від старіння інформації без переходу до надто жорсткої схеми пріоритетів. Найпереконливіше це підтверджує рис. 3, на

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

якому score–політика демонструє стабільно високу частку виконання HP SLO в усіх сценаріях і водночас не зводиться до жорсткого фіксованого пріоритету. Графіки backlog і processing debt на рис. 4 і рис. 5 слід інтерпретувати як ілюстрацію часової динаміки системи в режимі Short peak, а не як головний візуальний доказ переваги методу.

Робастність результатів, абляційний аналіз та варіативність. Для підвищення доказовості результатів додатково виконано три серії перевірок: 1) аналіз чутливості запропонованої score–політики до зміни ефективної пропускної здатності системи; 2) аналіз чутливості до профілю SLO; 3) абляційне дослідження, у якому окремі компоненти скорингової функції відключалися або використовувалися у спрощених комбінаціях. Такі перевірки дають змогу відокремити внесок окремих складових методу і перевірити, чи не є отриманий ефект наслідком одного вузького набору параметрів.

На рис. 6 показано, що зі зростанням доступної пропускної здатності від 1.0 до 8.0 відносна перевага запропонованої політики над FIFO за критерієм зваженого запізнення закономірно зменшується: від 48,9 % до 20,5 %. Це природно, оскільки за надлишкових ресурсів роль розумного вибору черги слабшає. Водночас у зоні дефіциту ресурсу, яка і є найважливішою для пікових режимів, виграш залишається найбільшим; приріст частки виконання HP SLO у цьому sweep становить від 10,0 до 89,0 в.п. порівняно з FIFO.

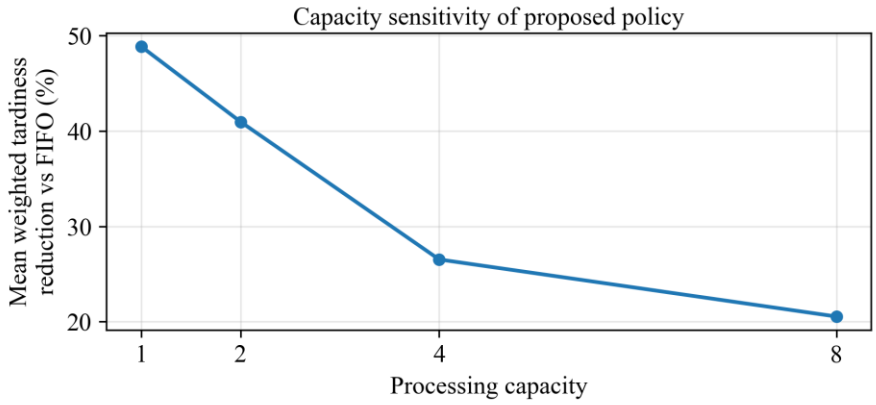


Рисунок 6 – Чутливість запропонованої політики до зміни пропускної здатності

На рис. 7 наведено перевірку для трьох профілів SLO: strict, medium та relaxed. Для всіх трьох профілів метод зберігає стабільну перевагу над FIFO: середнє зменшення зваженого запізнення становить 26,3 % при стандартному відхиленні 1,1 %, а середній виграш за HP SLO – 22,0 ± 1,7 в.п. Це свідчить про те, що ефект не зникає при послабленні або посиленні дедлайнів і не є артефактом лише одного налаштування часових обмежень.

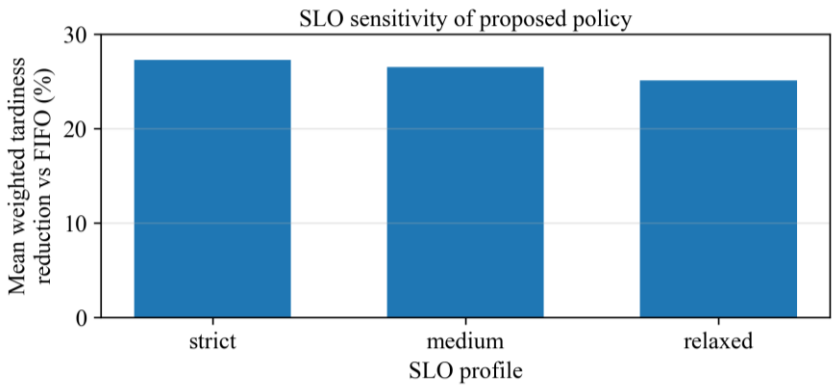


Рисунок 7 – Чутливість запропонованої політики до профілю SLO

Абляційний аналіз дозволяє оцінити внесок окремих складових скорингової моделі. Табл. 4 показує, що самі лише пріоритети або сама лише терміновість не забезпечують найкращого значення інтегрального критерію втрат. Додавання компонента вартості обробки дає найбільший внесок у зменшення J. Повна модель має результат, близький до найкращого абляційного варіанта, але зберігає системні обмеження й міжрівневий контекст, що є важливим для практичної політики керування, а не лише для мінімізації однієї агрегованої метрики.

Таблиця 4

Результати абляційного аналізу score–моделі

Варіант score–моделі	**J, млн ум. од.**	**Відхилення від повної моделі, %**	**HP SLO, %**
Повна модель	351.231	+0.00	98.38
Пріоритет + терміновість + вартість	347.176	−1.15	98.46
Лише пріоритет	360.248	+2.57	98.56
Пріоритет + терміновість	360.666	+2.69	98.36
Лише терміновість	368.696	+4.97	98.36

Оскільки у збережених notebook–результатах відсутні окремі траєкторії для багатьох незалежних повторів із різними початковими зернами, класичні довірчі інтервали за прогоном не можуть бути відновлені без повторного повного запуску моделі. Тому варіативність у цій статті характеризується стандартним відхиленням за наборами робастнісних налаштувань. Відповідні підсумки наведено в табл. 5. Такий підхід не замінює повноцінного багатосерійного Monte Carlo–аналізу, але дає

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

додаткову оцінку стабільності висновків у межах наявної кодової бази й відтворених експериментів.

Таблиця 5

Узагальнення варіативності за серіями робастнісних налаштувань

Серія експериментів	**Зменшення відносно FIFO, % (mean ± std)**	**Зміна HP SLO відносно FIFO, в.п. (mean ± std)**
Зміна пропускної здатності (1.0–8.0)	34.2 ± 13.0	42.8 ± 35.2
Зміна профілю SLO (strict/medium/relaxed)	26.3 ± 1.1	22.0 ± 1.7

З урахуванням цих додаткових перевірок формулювання висновків було уточнено. Запропонований метод не слід трактувати як універсально домінуючий за всіма можливими метриками та налаштуваннями. Коректніше стверджувати, що в межах дослідженої notebook–моделі, для replay–навантажень на основі Google Borg 2019 trace і для перевірених сценаріїв зміни ресурсу та дедлайнів score–політика демонструє стійку перевагу за агрегованим критерієм втрат і майже не поступається жорстким статичним пріоритетам за HP SLO. Саме в такому формулюванні результати є найбільш технічно обґрунтованими та придатними для подальшої валідації на реальних production–логах.

Висновки. У статті розроблено та експериментально перевірено підхід до пріоритетної обробки повідомлень в OLAP–системах за умов пікових навантажень. Побудовано модель OLAP–конвеєра, що охоплює broker, ETL, micro–batching, запис у сховище, побудову похідних структур і фонові злиття, а також реалізовано порівняння чотирьох політик планування: FIFO, статичних пріоритетів, EDF і score–політики.

За результатами моделювання для чотирьох базових сценаріїв навантаження та додаткових робастнісних перевірок встановлено, що запропонована score–політика забезпечує найменший інтегральний критерій втрат J, високий рівень виконання SLO для критичних повідомлень та найкраще значення All SLO серед чотирьох основних політик. Це свідчить про доцільність урахування в одній моделі не лише дедлайнів і ваг повідомлень, а й поточного стану сховища та вартості обробки мікропакетів.

Наукова новизна дослідження полягає в тому, що запропоновано та перевірено метод міжрівневої пріоритезації повідомлень для OLAP–конвеєра, у якому критерії цінності, старіння, терміновості, ресурсоємності та стану сховища інтегровано в єдину скорингову модель прийняття рішень. На відміну від відомих підходів, ця модель дає змогу одночасно враховувати бізнес–критичність даних і технічний контекст обробки.

Перспективи подальших досліджень у даному напрямку. Подальші дослідження доцільно спрямувати на перевірку запропонованого підходу на реальних production–логах, побудову навчальних міток на основі фактичних запитів до дашбор-

дів і алертів, а також на дослідження самоадаптивного вибору параметрів score–функції.

Окремим перспективним напрямком є створення прототипу для конкретного стеку Kafka – ETL consumers – ClickHouse та вивчення багатокритеріальної оптимізації, у якій разом із weighted tardiness мінімізуються write amplification, merge debt і енергоспоживання системи.

ЛІТЕРАТУРА

1. Свірідов В., Мороз Б. Організація процесів обробки інформації за критеріями цінності та старіння в автоматизованих системах управління. Харків: Основа, 1992. 112 с.

2. Мороз Б. І. Методи і засоби організації процесів обробки інформації за критеріями цінності та старіння в системах автоматизованого управління та інформаційного обслуговування: дис. д–ра техн. наук. Дніпропетровськ, 1992. 358 с.

3. Kaul S., Yates R., Gruteser M. Real–time status: How often should one update? In: *Proceedings IEEE INFOCOM*. Orlando, FL, USA, 2012. P. 2731–2735. DOI: 10.1109/INFCOM.2012.6195689.

4. Abadi D. J., Carney D., Çetintemel U. et al. Aurora: A New Model and Architecture for Data Stream Management. *VLDB Journal*. 2003. Vol. 12, no. 2. P. 120–139. DOI: 10.1007/s00778–003–0095–z.

5. Abadi D. J., Ahmad Y., Balazinska M. et al. The Design of the Borealis Stream Processing Engine. In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR 2005)*. Asilomar, CA, USA, 2005. 13 p.

6. Liu C. L., Layland J. W. Scheduling Algorithms for Multiprogramming in a Hard–Real–Time Environment. *Journal of the ACM*. 1973. Vol. 20, no. 1. P. 46–61. DOI: 10.1145/321738.321743.

7. Smith W. E. Various Optimizers for Single–Stage Production. *Naval Research Logistics Quarterly*. 1956. Vol. 3, no. 1–2. P. 59–66. DOI: 10.1002/nav.3800030106.

8. Mao H., Alizadeh M., Menache I., Kandula S. Resource Management with Deep Reinforcement Learning. In: *HotNets '16: Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. 2016. P. 50–56. DOI: 10.1145/3005745.3005750.

9. Cormode G., Muthukrishnan S. An Improved Data Stream Summary: The Count–Min Sketch and its Applications. *Journal of Algorithms*. 2005. Vol. 55, no. 1. P. 58–75. DOI: 10.1016/j.jalgor.2003.12.001.

10. Flajolet P., Fusy É., Gandouet O., Meunier F. HyperLogLog: the analysis of a near–optimal cardinality estimation algorithm. In: *AofA 2007 – Proceedings of the 2007 Conference on Analysis of Algorithms*. *DMTCS Proceedings*. 2007. P. 127–146. DOI: 10.46298/dmtcs.3545.

11. O'Neil P., Cheng E., Gawlick D., O'Neil E. The Log–Structured Merge–Tree (LSM–Tree). *Acta Informatica*. 1996. Vol. 33. P. 351–385. DOI: 10.1007/s002360050048.

12. Pinedo M. *Scheduling: Theory, Algorithms, and Systems*. 5th ed. New York : Springer, 2016. 670 p. DOI: 10.1007/978–1–4614–2361–4.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

13. Tatbul E. N. Load Shedding Techniques for Data Stream Management Systems. PhD thesis. Brown University, 2006.
14. Burges C. J. C. From RankNet to LambdaRank to LambdaMART: An Overview. Microsoft Research Technical Report MSR–TR–2010–82. 2010.
15. Ke G., Meng Q., Finley T. et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*. 2017. P. 3146–3154. DOI: 10.5555/3294996.3295074.
16. Demers A., Keshav S., Shenker S. Analysis and Simulation of a Fair Queueing Algorithm. *SIGCOMM Computer Communication Review*. 1989. Vol. 19, no. 4. P. 1–12.
DOI: 10.1145/75246.75248.
17. Bennett J. C. R., Zhang H. WF2Q: Worst–case Fair Weighted Fair Queueing. In: Proceedings of IEEE INFOCOM 1996. San Francisco, CA, USA, 1996. Vol. 1. P. 120–128. DOI: 10.1109/INFCOM.1996.497885.
18. Bernet Y., Blake S., Grossman D., Smith A. An Informal Management Model for Diffserv Routers. RFC 3290. IETF, 2002. DOI: 10.17487/RFC3290.
19. Apache Software Foundation. Apache Kafka Documentation.
URL: https://kafka.apache.org/documentation/ (дата звернення: 25.03.2026).
20. Confluent, Inc. Monitor Consumer Lag.
URL: https://docs.confluent.io/platform/current/monitor/monitor–consumer–lag.html (дата звернення: 25.03.2026).
21. ClickHouse, Inc. Part merges. URL: https://clickhouse.com/docs/merges (дата звернення: 25.03.2026).
22. ClickHouse, Inc. Materialized views. URL: https://clickhouse.com/docs/materialized–views (дата звернення: 25.03.2026).
23. Harinarayan V., Rajaraman A., Ullman J. D. Implementing Data Cubes Efficiently. In: *SIGMOD '96*. 1996. P. 205–216. DOI: 10.1145/235968.233333.
24. Gray J., Chaudhuri S., Bosworth A. et al. Data Cube: A Relational Aggregation Operator Generalizing Group–By, Cross–Tab, and Sub–Totals. *Data Mining and Knowledge Discovery*. 1997. Vol. 1. P. 29–53. DOI: 10.1023/A:1009726021843.
25. Zhao W., Rusu F., Dong B. et al. Incremental View Maintenance over Array Data. In: *SIGMOD 2017*. 2017. P. 139–154. DOI: 10.1145/3035918.3064041.
26. Chaudhuri S., Dayal U. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*. 1997. Vol. 26, no. 1. P. 65–74. DOI: 10.1145/248603.248616.
27. Babcock B., Babu S., Datar M., Motwani R., Widom J. Models and Issues in Data Stream Systems. In: *Proceedings of the Twenty–First ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (PODS 2002)*. 2002. P. 1–16.
DOI: 10.1145/543613.543615.
28. Zaharia M., Das T., Li H., Hunter T., Shenker S., Stoica I. Discretized Streams: Fault–Tolerant Streaming Computation at Scale. In: *Proceedings of the Twenty–Fourth ACM Symposium on Operating Systems Principles (SOSP 2013)*. 2013. P. 423–438.
DOI: 10.1145/2517349.2522737.

REFERENCES

1. Abadi, D. J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.–H., Lindner, W., Maskey, A. S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., & Zdonik, S. (2005). The design of the Borealis stream processing engine. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR 2005)*.

2. Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., & Zdonik, S. (2003). Aurora: A new model and architecture for data stream management. *The VLDB Journal, 12*(2), 120–139.
https://doi.org/10.1007/s00778–003–0095–z

3. Apache Software Foundation. (n.d.). *Apache Kafka documentation*.
https://kafka.apache.org/documentation/

4. Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the Twenty–First ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems* (pp. 1–16).
https://doi.org/10.1145/543613.543615

5. Bennett, J. C. R., & Zhang, H. (1996). WF2Q: Worst–case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM 1996* (Vol. 1, pp. 120–128).
https://doi.org/10.1109/INFCOM.1996.497885

6. Bernet, Y., Blake, S., Grossman, D., & Smith, A. (2002). *An informal management model for Diffserv routers* (RFC 3290). Internet Engineering Task Force.
https://doi.org/10.17487/RFC3290

7. Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An overview* (Microsoft Research Technical Report MSR–TR–2010–82). Microsoft Research.

8. Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *SIGMOD Record, 26*(1), 65–74. https://doi.org/10.1145/248603.248616

9. ClickHouse, Inc. (n.d.–a). *Materialized views*.
https://clickhouse.com/docs/materialized–views

10. ClickHouse, Inc. (n.d.–b). *Part merges*. https://clickhouse.com/docs/merges

11. Confluent, Inc. (n.d.). *Monitor consumer lag*.
https://docs.confluent.io/platform/current/monitor/monitor–consumer–lag.html

12. Cormode, G., & Muthukrishnan, S. (2005). An improved data stream summary: The Count–Min Sketch and its applications. *Journal of Algorithms, 55*(1), 58–75.
https://doi.org/10.1016/j.jalgor.2003.12.001

13. Demers, A., Keshav, S., & Shenker, S. (1989). Analysis and simulation of a fair queueing algorithm. *SIGCOMM Computer Communication Review, 19*(4), 1–12.
https://doi.org/10.1145/75246.75248

14. Flajolet, P., Fusy, É., Gandouet, O., & Meunier, F. (2007). HyperLogLog: The analysis of a near–optimal cardinality estimation algorithm. In *AofA 2007: Proceedings of the 2007 Conference on Analysis of Algorithms* (pp. 127–146). https://doi.org/10.46298/dmtcs.3545

15. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., & Pirahesh, H. (1997). Data cube: A relational aggregation operator generalizing group–by, cross–tab, and sub–totals. *Data Mining and Knowledge Discovery, 1*, 29–53.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

https://doi.org/10.1023/A:1009726021843
16. Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing data cubes efficiently. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (pp. 205–216). https://doi.org/10.1145/235968.233333
17. Kaul, S., Yates, R., & Gruteser, M. (2012). Real–time status: How often should one update? In *Proceedings IEEE INFOCOM* (pp. 2731–2735).
https://doi.org/10.1109/INFCOM.2012.6195689
18. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.–Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30* (pp. 3146–3154). https://doi.org/10.5555/3294996.3295074
19. Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard–real–time environment. *Journal of the ACM, 20*(1), 46–61.
https://doi.org/10.1145/321738.321743
20. Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (pp. 50–56). https://doi.org/10.1145/3005745.3005750
21. Moroz, B. I. (1992). *Metody i zasoby orhanizatsii protsesiv obrobky informatsii za kryteriiamy tsinnosti ta starinnia v systemakh avtomatyzovanoho upravlinnia ta informatsiinoho obsluhovuvannia [Methods and means of organizing information processing by the criteria of value and ageing in automated control and information service systems]* (Doctoral dissertation). Dnipropetrovsk [in Ukrainian].
22. O'Neil, P., Cheng, E., Gawlick, D., & O'Neil, E. (1996). The log–structured merge–tree (LSM–tree). *Acta Informatica, 33*, 351–385. https://doi.org/10.1007/s002360050048
23. Pinedo, M. (2016). *Scheduling: Theory, algorithms, and systems* (5th ed.). Springer. https://doi.org/10.1007/978–1–4614–2361–4
24. Smith, W. E. (1956). Various optimizers for single–stage production. *Naval Research Logistics Quarterly, 3*(1–2), 59–66. https://doi.org/10.1002/nav.3800030106
25. Sviridov, V., & Moroz, B. (1992). *Orhanizatsiia protsesiv obrobky informatsii za kryteriiamy tsinnosti ta starinnia v avtomatyzovanykh systemakh upravlinnia [Organization of information processing by the criteria of value and ageing in automated control systems]*. Kharkiv: Osnova [in Ukrainian].
26. Tatbul, N. (2006). *Load shedding techniques for data stream management systems* (Doctoral dissertation, Brown University).
27. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2013). Discretized streams: Fault–tolerant streaming computation at scale. In *Proceedings of the Twenty–Fourth ACM Symposium on Operating Systems Principles* (pp. 423–438).
https://doi.org/10.1145/2517349.2522737
28. Zhao, W., Rusu, F., Dong, B., & Wu, K. (2017). Incremental view maintenance over array data. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 139–154). https://doi.org/10.1145/3035918.3064041

Received 04.06.2026.
Accepted 06.08.2026.
Published 30.10.2026

Modeling and experimental evaluation of priority–based message processing in OLAP systems under peak loads

Abstract. A priority–based message–processing model for OLAP systems under peak loads is proposed. The method combines value, urgency, and processing cost and reduces losses caused by delayed data delivery.

The paper studies the experimental verification of a controlled service discipline for OLAP systems, where the value of information, its ageing, and the limited capacity of the analytical pipeline are critical. An analysis of recent studies shows that stream processing, classical scheduling, and analytical–warehouse internals are usually considered separately, without a unified control loop. The aim of the study is to develop and experimentally evaluate a controlled priority discipline that combines the business value of messages with the current technical state of the warehouse. A model of the OLAP pipeline is proposed that includes a broker, ETL, micro–batching, writes to the warehouse, materialized views, and background merge operations. To evaluate losses caused by untimely data delivery, a weighted tardiness criterion is used, while the selection of the next micro–batch is based on an integral scoring function that combines the expected impact on analytical decisions, urgency, and processing cost. The main experiment covers four uneven–load scenarios (Uniform, Short peak, Long peak, Mixed load) under the medium SLO profile and capacity = 4.0; additional robustness checks are performed for different capacity levels and SLO profiles. On average, the proposed method achieved the smallest aggregated loss criterion, J = 351.231 million conventional units, which is 2.5% better than static priorities, 22.2% better than EDF, and 26.5% better than FIFO. The method also provided 98.4% HP SLO fulfilment and 83.5% All SLO fulfilment. The results confirm the feasibility of cross–layer control of message processing in OLAP systems under peak loads and provide a reproducible basis for further validation on production traces.

Keywords: OLAP system, analytical pipeline, micro–batching, prioritization, weighted tardiness, SLO, machine learning, modeling, experimental evaluation.

Єрмаков Владислав Віталійович – аспірант, Національний технічний університет «Дніпровська політехніка», м. Дніпро, Україна.
ORCID: https://orcid.org/0009–0009–8670–7667.
Мороз Борис Іванович – доктор технічних наук, професор, Національний технічний університет «Дніпровська політехніка», м. Дніпро, Україна.
ORCID: https://orcid.org/0000–0002–5625–0864.

Yermakov Vladyslav Vitaliiovych – Postgraduate Student, Dnipro University of Technology, Dnipro, Ukraine.
ORCID: https://orcid.org/0009–0009–8670–7667.
Moroz Borys Ivanovych – Doctor of Technical Sciences, Professor, Dnipro University of Technology, Dnipro, Ukraine.
ORCID: https://orcid.org/0000–0002–5625–0864.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)