ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2025-82-30
УДК 621.391

ПАРХОМЕЙ Ігор
Київський національний університет імені Тараса Шевченка
https://orcid.org/0000-0002-9510-7657
e-mail: i_parhomey@ukr.net
БОЙКО Юлій
Хмельницький національний університет
https://orcid.org/0000-0003-0603-7827
e-mail: boiko_julius@ukr.net
ЛЕМЕШКО В'ячеслав
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
https://orcid.org/0009-0008-1495-715X
e-mail: slava.lemeshko@gmail.com
ЄРЬОМЕНКО Олександр
Хмельницький національний університет
https://orcid.org/0000-0001-5110-3761
e-mail: yeromenko_s@ukr.net

ОПТИМІЗАЦІЯ ЛОГУВАННЯ В ІНФОРМАЦІЙНИХ СИСТЕМАХ НА ОСНОВІ МАТЕМАТИЧНОЇ МОДЕЛІ УПРАВЛІННЯ БУФЕРОМ У ШАБЛОНІ PRODUCER-CONSUMER

У статті розглядається підхід до оптимізації процесів логування в інформаційних системах шляхом застосування математичної моделі управління буфером у межах шаблону взаємодії Producer–Consumer. Запропоновано модель, що дозволяє балансувати навантаження між генератором логів (producer) і підсистемою зберігання (consumer), з урахуванням обмежень буферної пам'яті та параметрів системи. Аналіз ефективності моделі демонструє зменшення втрат логів, зниження затримки обробки та покращення стабільності логування в умовах високого навантаження. Представлено результати моделювання, що підтверджують доцільність використання розробленого підходу для побудови надійних систем аудиту та моніторингу подій.
Ключові слова: логування, керування буфером, інформаційні системи, шаблон Producer–Consumer, моделювання, аудит, моніторинг, оптимізація, високонавантажене середовище, теорія черг

PARKHOMEY Igor
Taras Shevchenko National University of Kyiv
BOIKO Juliy, EROMENKO Oleksander
Khmelnytskyi National University
LEMESHKO Viacheslav
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

OPTIMIZATION OF LOGGING IN INFORMATION SYSTEMS BASED ON A MATHEMATICAL MODEL OF BUFFER MANAGEMENT IN THE PRODUCER–CONSUMER PATTERN

The article presents a comprehensive approach to optimizing logging processes in information systems by utilizing a mathematical model for buffer management within the widely used Producer–Consumer interaction pattern. The study addresses the challenges associated with high-frequency logging, such as data loss, increased latency, and resource overload, which are common in distributed and high-load environments. To mitigate these issues, a dynamic buffer management model is proposed that adaptively regulates the interaction between log-generating components (producers) and log-handling or storage subsystems (consumers). The model takes into account critical system parameters, including buffer size, event generation rate, and consumer processing speed. It enables dynamic adjustment of logging strategies depending on the current load and buffer state.

The research includes the formalization of the proposed model using discrete-time mathematics, with particular attention to queue theory and finite-buffer constraints. Simulation experiments conducted in the study demonstrate that the model significantly reduces log loss, optimizes system responsiveness, and ensures stable operation of logging mechanisms even under extreme workloads. The findings suggest that implementing such a model contributes to the design of resilient auditing and monitoring subsystems, especially in cybersecurity-sensitive and mission-critical infrastructures. The approach can be integrated into various architectural layers of modern information systems, improving reliability, maintainability, and traceability of operations in real time. Recommendations for practical implementation and possible extensions of the model for adaptive load prediction are also provided.
Keywords: logging, buffer management, information systems, Producer–Consumer pattern, modeling, audit, monitoring, optimization, high-load environment, queueing theory

Стаття надійшла до редакції / Received 12.04.2025
Прийнята до друку / Accepted 07.05.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Логування стану інформаційних систем є критично важливим компонентом платформи проєктування та виконання бізнес-сервісів, оскільки це забезпечує моніторинг, аналіз та діагностику роботи системи [1]. Логи дозволяють оперативно визначати причину збоїв або помилок, містять інформацію про те, на якому етапі бізнес-сервісу виникла проблема, які параметри передавалися та яким був результат виконання. Вони також дозволяють відстежувати швидкодію системи, виявляти вузькі місця та оптимізувати роботу. Крім того, логи дають змогу аналізувати взаємодію користувачів із системою, виявляти неефективності й знаходити шляхи для вдосконалення функціоналу.

У сучасних умовах високонавантажених, масштабованих або розподілених інформаційних середовищ функція логування перестає бути лише допоміжним інструментом і набуває статусу ключового елементу інфраструктури. Водночас збільшення кількості подій, обсягу даних, що підлягають запису, та вимог до швидкодії породжує низку проблем, пов'язаних із втратами логів, перевантаженням буферів, затримками в обробці та зниженням стабільності підсистем логування під час пікових навантажень.

Суттєвим фактором, що впливає на ефективність логування, є механізм взаємодії між компонентами, які генерують події (producer), та компонентами, що їх споживають (consumer). У типовій архітектурі така взаємодія реалізується через буфери обмеженого розміру, і саме спосіб керування цим буфером визначає рівень надійності та стабільності логування [2].

Отже, постає науково-практичне завдання — розробити математичну модель керування буфером у шаблоні producer–consumer, яка б забезпечувала динамічну адаптацію до змін у навантаженні в реальному часі. Така модель дозволить уникнути перевантаження буферів, забезпечити повноту та своєчасність логування без зайвого споживання ресурсів системи, а також зробить логування масштабованим і стійким до пікових навантажень. Розв'язання цієї проблеми має високу практичну цінність для розробників програмного забезпечення, архітекторів інформаційних систем, а також фахівців з інформаційної безпеки, оскільки дає змогу істотно підвищити якість моніторингу, зменшити втрати критичної інформації й покращити загальну продуктивність і стійкість системи.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Для того щоб забезпечувати належний рівень виконання бізнес-сервісів, необхідно постійно моніторити стан системи. До реалізації та інтеграції підсистеми логування можна віднести ряд проблем, які пов'язані із великим обсягом даних, підвищеним впливом на продуктивність системи та складнощами аналізу цих даних.

В даній роботі увага приділяється підсистемі логування, яка зберігає останній актуальний стан виконання бізнес-сервісу, не зберігаючи детальний історичний перебіг. При цьому задля оптимізації пропонується вдосконалена модель producer-consumer із часовою агрегацією (T_{merge}).

Аналіз останніх наукових робіт [3-14] свідчить про значний інтерес до тематики оптимізації підсистем логування та обробки потокових даних, зокрема в контексті зменшення навантаження на систему управління базами даних (СУБД) і покращення масштабованості. У сучасних потокових та розподілених системах дедалі частіше застосовуються «батчові» або мікробатчові підходи [3-13], що дають змогу зменшувати обсяг транзакцій і підвищувати продуктивність. Дослідження, присвячені моделі producer-consumer [4,9], наголошують на важливості підтримання балансу між швидкістю формування та швидкістю обробки даних, а також на гнучкому підході до розподілу ресурсів у режимі реального часу. Застосування адаптивних інтервалів агрегації [7,15] доводить свою ефективність у сценаріях з високим обсягом логів і нерівномірною частотою надходження даних, зумовлюючи помітне зменшення загальних витрат на запис і підвищення стійкості до пікових навантажень.

Інші дослідження [5,6,8,10,11] фокусуються на аспектах реалізації розподілених та масштабованих систем логування, пропонуючи методи структурованого зберігання, розвантаження центральних вузлів і швидкого пошуку. Зокрема, наголошується на важливості оптимізованих буферів (еластичних, із можливістю адаптації розміру) та алгоритмів узгодження швидкостей запису й читання.

Таким чином, у більшості досліджень підкреслюється потреба у збалансованій реалізації буферизації та агрегації для уникнення переповнення системи частими одиночними транзакціями. Водночас відсутня цілісна модель, що поєднує класичний шаблон producer-consumer з регульованим агрегаційним інтервалом, зокрема для реляційних СУБД, які активно використовуються у великій кількості бізнес-сервісів.

МОДЕЛЬ СИСТЕМИ ЛОГУВАННЯ

Розглянемо процес логування кроків виконання бізнес-процесу. При виконанні кожного кроку бізнес-процесу відбувається виклик функції-реєстрації даних у деякої абстракції, реалізованої за допомогою шаблону проектування "репозиторій" [16]. Реалізація даної абстракції формує запит до реляційної СУБД на створення нового запису в реляційні таблиці. Після завершення виконання кроку бізнес-процесу відбувається

виклик функції-оновлення даних в абстракції “репозиторій”. Реалізація абстракції формує запит до реляційної СУБД на оновлення існуючого запису у реляційні таблиці.

Дана реалізація має ряд переваг та недоліків. До переваг можна віднести, що логування кожного кроку бізнес-процесу дозволяє відстежувати його виконання в деталях та в режимі реального часу, що сприяє діагностиці проблем та підвищенню прозорості роботи системи. Використання патерну "репозиторій" забезпечує чітке розділення бізнес-логіки та роботи з базою даних, що дозволяє легше змінювати чи вдосконалювати компоненти системи. Абстракція через "репозиторій" спрощує підтримку та тестування, оскільки дає змогу змінювати внутрішню реалізацію доступу до даних без впливу на бізнес-логіку. Використання реляційної СУБД як сховища даних є стандартним підходом, що робить рішення зрозумілим для багатьох розробників і легко інтегрується з іншими системами. Логи зберігаються в структурованому вигляді в реляційній базі даних, що дозволяє зручно використовувати їх для побудови звітів, аналізу ефективності процесів і пошуку вузьких місць.

Але часті операції запису й оновлення створюють значне навантаження на реляційну СУБД, особливо у випадку високого рівня паралелізму або великої кількості кроків бізнес-процесів. Через постійний доступ до бази даних для запису й оновлення записів уповільнюється виконання бізнес-процесів. Якщо кожен крок оформляється як окрема транзакція, це може призводити до затримок через блокування ресурсів. Часте оновлення й збереження логів у базі даних призводить до накопичення великих обсягів історичних даних, які необхідно періодично очищувати, рис.1.

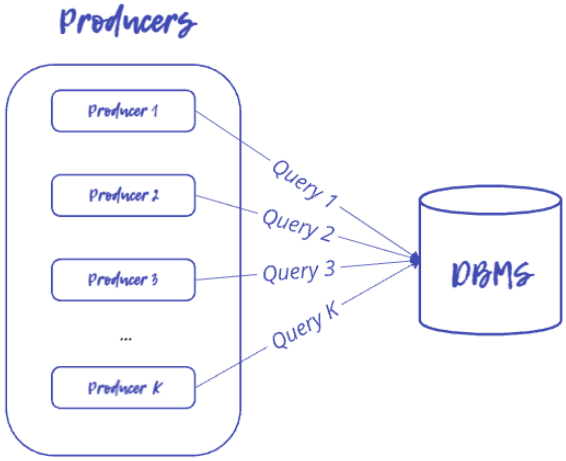


Рис. 1. Загальна схема першого підходу

Щоб зменшити вплив на СУБД, зменшити кількість даних, які надсилаються через мережу, зменшити кількість операцій із дисковою підсистемою, а також пришвидшити виконання кроків бізнес-процесів, запропоновано та реалізовано вдосконалення. Впроваджено “розумний” буфер між підсистемою виконання бізнес-процесів та підсистемою логування, який асинхронно отримує дані логування у вигляді повідомлень і перед записом виконує операцію об’єднання (агрегації) цих повідомлень з метою мінімізації кількості запитів до СУБД.

За допомогою механізму фіче-тоглінгу [17] стало можливо впровадити нову реалізацію з опцією переключатися на попередню. З новим підходом (рис.2) кожний крок екземпляра бізнес-процесу генерує дані логування у вигляді повідомлень, які накопичуються у деякій черзі. Певна кількість обробників (consumers) перетворює ці повідомлення на запити до реляційної СУБД. Підсистема логування є моделлю “виробник-черга-споживач” (producer-consumer) [18, 19].

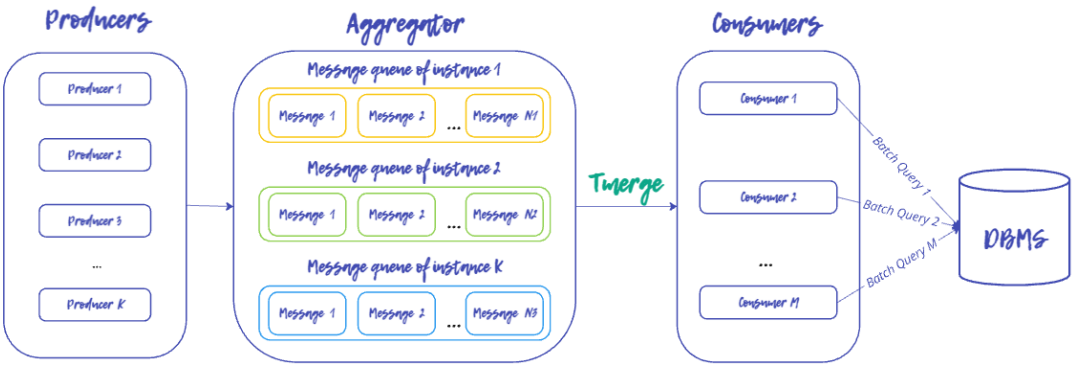


Рис.2. Загальна схема запропонованого підходу на основі концепції producer–consumer

Математична модель шаблону producer-consumer описує взаємодію між двома типами процесів: виробниками (producers), які генерують дані або продукти, та споживачами (consumers), які споживають ці дані. Зазвичай вони використовують спільний буфер обмеженого розміру для обміну даними. Модель допомагає зрозуміти оптимальні умови для узгодження продуктивності обох сторін і уникнення надмірного очікування чи блокувань [20, 21].

Змінні моделі: N — розмір спільного буфера (це максимальна кількість елементів, які можуть зберігатися одночасно); $P(t)$ — кількість елементів, згенерованих виробниками до моменту часу t; $C(t)$ — кількість елементів, спожитих споживачами до моменту часу t; $B(t)$ — кількість елементів у буфері на момент часу t (це змінна стану буфера, яка оновлюється в часі відповідно до дій виробників і споживачів; r_p та r_c — середні швидкості генерації (виробниками) та споживання (споживачами) елементів, відповідно).

Стан буфера в будь-який момент часу можна описати як:

$$B(t) = P(t) - C(t), \tag{1}$$

де $B(t)$ обмежено між 0 та N та $0 <= B(t) <= N$

Зміну кількості елементів у буфері в часі представимо через похідну:

$$dB(t)/dt = dP(t)/dt - dC(t)/dt = r\_p - r\_c. \tag{2}$$

Оскільки швидкість генерації елементів $dP(t)/dt$ та швидкість споживання елементів $dC(t)/dt$ дорівнюють r_p та r_c відповідно, це рівняння можна спростити до:

$$dB(t)/dt = r\_p - r\_c. \tag{3}$$

Умови стабільності:

1. Умова заповнення буфера: якщо $r\_p > r\_c$ то буфер поступово заповнюватиметься, і при досягненні $B(t)=N$ виробникам доведеться зупинитися або очікувати, що призведе до затримок.
2. Умова спустошення буфера: якщо $r\_p < r\_c$, то буфер буде спустошений, і споживачі будуть змушені очікувати, поки з'являться нові дані.
3. Стабільна робота системи: для оптимальної роботи необхідно, щоб $r\_p = r\_c$. У цьому випадку буфер перебуватиме в рівноважному стані, не переповнюючись і не залишаючись порожнім.

Математична модель шаблону producer-consumer допомагає визначити, як підтримувати баланс між швидкостями генерації та споживання. Це забезпечує оптимальне використання буфера, мінімізуючи затримки як для виробників, так і для споживачів.

Але такий підхід не дозволяє зменшити кількість запитів до СУБД. Для зменшення кількості запитів до СУБД запропоновано змінити поведінку буфера: обробка елементів виконується з певною періодичністю (агрегаційний інтервал T_{merge}), тобто споживачі працюють у режимі накопичення даних. Розглядається модель, за якої дані споживаються батчем після закінчення періоду T_{merge}, що суттєво впливає на загальне навантаження на СУБД.

Ведемо додаткові змінні моделі: T_{merge} — агрегаційний інтервал (період, протягом якого дані накопичуються перед їх обробкою); $M(t)$ — кількість елементів, накопичених у буфері за час T_{merge} перед споживанням; $C(t)$ — кількість споживачів, які одночасно обробляють накопичені дані.

Врахуємо, що стан буфера тепер залежить від таких фаз, а саме від фази накопичення - протягом часу T_{merge}, елементи додаються в буфер, але не споживаються. Крім того, врахуємо фазу споживання - по завершенню T_{merge}, накопичені елементи споживаються всіма $C(t)$ споживачами за один запит.

Рівняння для стану буфера, $B(t)$ в цьому випадку виглядає із врахуванням (1) наступним чином:

$$c(t) = \begin{cases} 0, \ t \bmod T_{merge} \neq 0 \\ \min\big(B(t), C(t) \cdot M(t)\big), \ t \bmod T_{merge} = 0 \end{cases} \tag{4}$$

Фаза накопичення ($t \bmod T_{merge} \neq 0$):
- споживання не відбувається $C(t) = 0$;
- буфер зростає із швидкістю, визначеною виробниками r_p.

Фаза споживання ($t \bmod T_{merge} = 0$):
- усі $C(t)$ споживачі одночасно споживають дані.
- максимальна кількість даних, яку може бути спожито: $C(t) \cdot M(t)$, де

$$M(t) = \int_{t-T_{merge}}^{t} p(t)dt \ . \tag{5}$$

де $M(t)$ - кількість накопичених елементів за час T_{merge}.

Якщо $B(t) < C(t) \cdot M(t)$, то споживається весь вміст буфера, і після цього $B(t) = 0$.

Для уникнення переповнення буфера необхідно, щоб середня швидкість споживання за T_{merge} відповідала швидкості генерації:

$$C(t) \cdot M(t) \geq r\_p \cdot T_{merge} \ . \quad (6)$$

Аналіз результатів застосування запропонованого підходу

Дослідження проводилося на таких обчислювальних ресурсах: веб-сервер (4 ядра, 16 ГБ ОЗП) та сервер баз даних SQL (4 ядра, 32 ГБ ОЗП). Для оцінки впливу на сервер баз даних Microsoft SQL Server запропоновано аналіз таких метрик: Execution Count (кількість виконань запитів), Total Duration (ms) (загальна тривалість виконання запитів у мілісекундах) та Total Logical Writes (KB) (загальний обсяг логічних записів у кілобайтах). Вибір метрик обґрунтований необхідністю забезпечення релевантності аналізу, мінімізації впливу зовнішніх факторів та фокусування на аспектах, які мають найбільше значення для ефективності роботи MS SQL Server у досліджуваних умовах.

а) **б)**

Рис.3. Використання ресурсів центрального процесора веб-сервера

В результаті аналізу графіків використання ресурсів центрального процесора веб-сервера до (рис. 3, а) та після (рис. 3, б) впровадження підсистеми логування, можна зробити висновок, що загальна картина навантаження на процесор залишається подібною. Зокрема, обидва графіки демонструють наявність пікових значень використання CPU, що повторюються з приблизно однаковою частотою та амплітудою. Хоча після впровадження змін (рис. 3, б) можна помітити дещо менш виражені стрибки навантаження, загальна тенденція збережена. Це свідчить про те, що впровадження підсистеми логування не впливає на рівень завантаженості центрального процесора є досить гарним показником.

На графіку до впровадження змін (рис. 4, а) спостерігається відносно стабільне використання оперативної пам’яті з періодичними піковими навантаженнями. Після впровадження підсистеми логування (рис. 4, б) загальна структура навантаження зберігається, однак простежується певне збільшення базового рівня використання пам’яті.

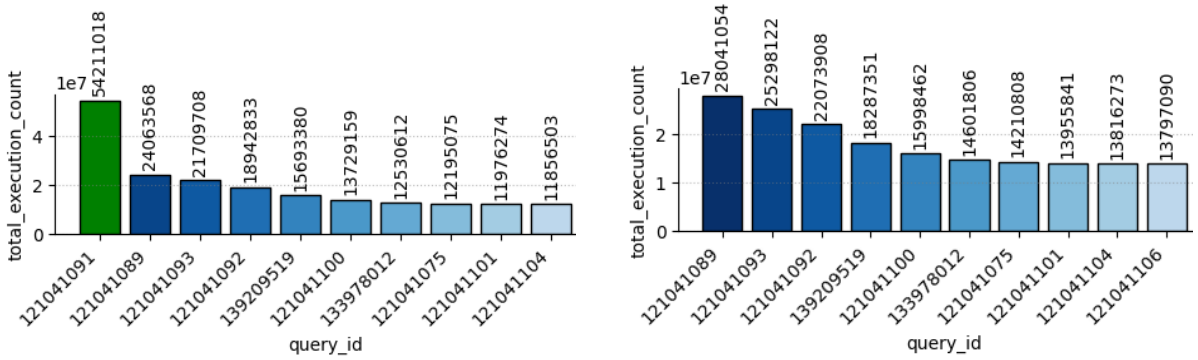
а) **б)**

Рис.4. Використання ресурсів оперативної пам’яті веб-сервера

Це може свідчити про підвищене споживання ресурсів оперативної пам’яті внаслідок нововведень. Водночас, частота та інтенсивність пікових навантажень залишаються подібними до тих, що спостерігалися до впровадження змін, що може вказувати на те, що основні закономірності роботи веб-сервера залишилися незмінними.

Найбільш значущою зміною є відсутність у QueryStore запиту з ідентифікатором 121041091 (рис. 5а), який до впровадження логування мав найбільшу частку викликів (відображений зеленим кольором), який оновлював дані журналу.

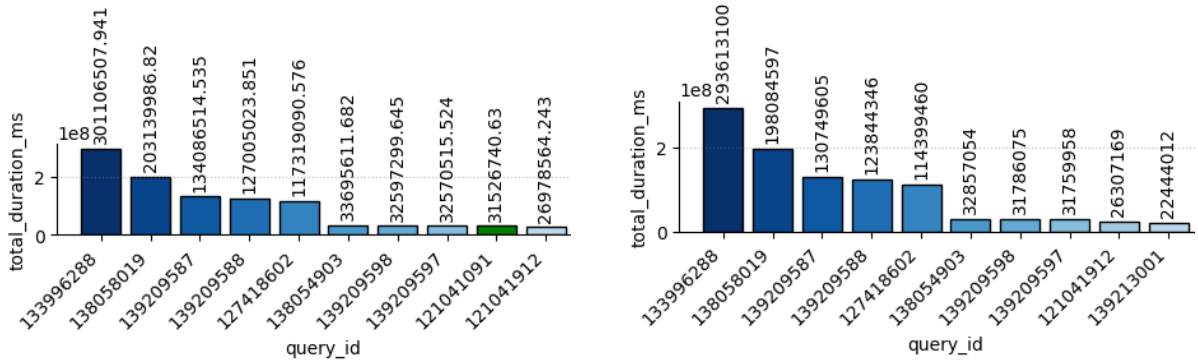
Це свідчить про те, що його виконання стало значно рідшим, що зменшило вплив на сервер БД. Аналізуючи графік після впровадження змін (рис. 5б), можна помітити, що інші запити зазнали перерозподілу частоти викликів, зокрема, найбільш частотні запити з ідентифікаторами 121041054, 121041093 та 121041092 демонструють зростання кількості викликів. Що є наслідком системи до нового підходу збереження логів.



а) **б)**

Рис.5. Загальна кількість викликів запитів

До впровадження запропонованого підходу функціонування підсистеми логування (рис. 6а) запит з ідентифікатором 121041091, виділений зеленим кольором, мав певний вплив на загальну тривалість виконання запитів.

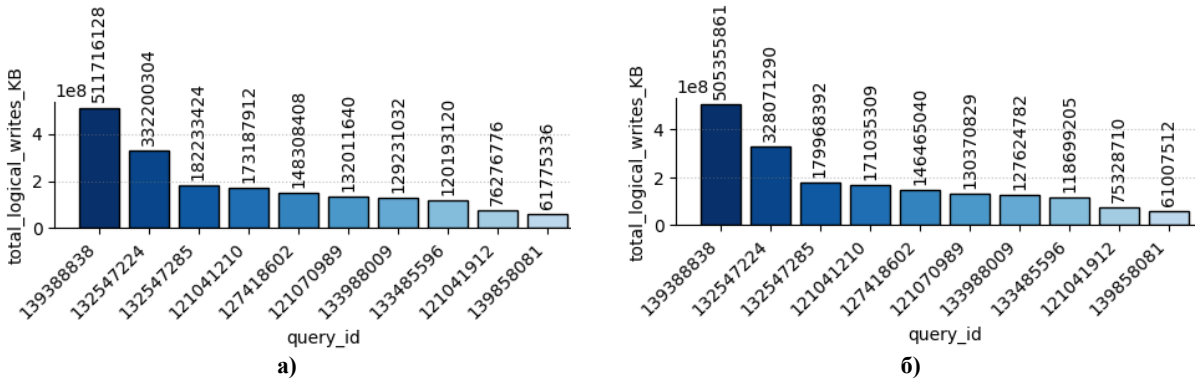


а) **б)**

Рис.6. Загальна тривалість запитів в мс

Після впровадження змін (рис. 6б) цей запит більше не фіксується в QueryStore, що свідчить про його усунення або значне зниження частоти виконання. Це дозволило суттєво зменшити навантаження на сервер бази даних. Деякі запити, які мали значну сумарну тривалість виконання до змін (наприклад, 121041912), у новій конфігурації більше не фіксуються або демонструють суттєве зниження часу виконання. Це свідчить про оптимізацію або зменшення частоти їхнього використання.

Порівнюючи обидва графіки (рис. 7а та рис. 7б), спостерігається загальне зменшення кількості логічних записів після впровадження підсистеми логування.



а) **б)**

Рис.7. Загальна кількість логічних записів

На графіку "б" максимальні значення загального обсягу логічних записів значно менші, ніж на графіку "а", що свідчить про підвищення ефективності обробки запитів. Зменшення загальної кількості логічних записів без суттєвого збільшення кількості малих запитів свідчить про покращення алгоритмів запису та, ймовірно, зменшення дублюючих або зайвих операцій логування. Таким чином, накопичувальний

(batch) режим дозволив зменшити кількість звернень до БД і забезпечити стабільніший розподіл навантаження.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У цій роботі було розглянуто й проаналізовано підхід до логування бізнес-сервісів, який базується на моделі producer-consumer із впровадженням буфера, що працює в режимі накопичення та періодичної (батчової) обробки даних. Дослідження підтвердило, що математична модель producer-consumer з урахуванням агрегаційного інтервалу T_{merge} дає змогу зменшити кількість запитів до СУБД, оскільки замість безперервних одиночних операцій запису логів відбувається групова обробка. Емпіричні результати свідчать про скорочення загальної кількості логічних записів, зниження навантаження на процесори сервера бази даних та часткове збільшення споживання оперативної пам'яті веб-сервера (за рахунок зберігання батчу).

Аналіз останніх наукових досліджень у сфері буферизації та логування (2021–2023 рр.) показує, що світова практика також переходить до батч-обробки повідомлень і використання адаптивних періодичних інтервалів. Це підтверджує перспективність даного підходу й узгоджується з трендом оптимізації логування для зменшення накладних витрат та підвищення стійкості систем.

Впроваджена реалізація дає змогу значно зменшити кількість транзакцій на запис до бази даних, водночас забезпечуючи достатній рівень деталізації логів для подальшого аналізу.

Запропонований алгоритм може бути адаптований до різноманітних умов: змінної швидкості генерування логів, різної кількості споживачів та індивідуальних вимог до продуктивності. За умови правильного налаштування періоду T_{merge} і розрахунку необхідної ємності буфера, підхід забезпечує стабільну роботу системи навіть за інтенсивної генерації логів. Подальші дослідження можуть бути зосереджені на вдосконаленні механізмів адаптації агрегаційного інтервалу в режимі реального часу (на основі попереднього аналізу навантаження та прогнозування), а також на розширенні моделі для розподілених середовищ, де логування виконується одночасно на кількох вузлах.

Література

1. Smart, T. (2023). Logging and Testing. In: Serverless Beyond the Buzzword. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8761-3_8.

2. Wang, K.C. (2023). Process Management in Embedded Systems. In: Embedded and Real-Time Operating Systems. Springer, Cham. https://doi.org/10.1007/978-3-031-28701-5_5.

3. Sax, M. J. (2020). Performance optimizations and operator semantics for streaming data flow programs. Humboldt-Universität zu Berlin. Access mode: https://edoc.hu-berlin.de/bitstreams/6a57c23e-f7ff-41ec-b202-63d2d01fb1da/download. (last access: 07.05.2025).

4. Sharma, M. K. (2020). A Design Space for Distributed Producer-Consumer Data Structures Using RDMA (Master's thesis). University of Waterloo. Access mode: https://uwspace.uwaterloo.ca/bitstream/10012/16163/4/Sharma_Manoj.pdf. (last access: 07.05.2025).

5. Foyer, C. M. (2021). Abstractions for Portable Data Management in Heterogeneous Memory Systems (PhD dissertation). University of Bristol. Access mode: https://research-information.bris.ac.uk/ws/portalfiles/portal/281444302. (last access: 07.05.2025).

6. Marcu, O. C., Costan, A., Antoniu, G., & Pérez-Hernández, M. S. (2018). Storage and ingestion systems in support of stream processing: A survey. Inria. Access mode: https://inria.hal.science/hal-01939280/file/RT-0501v2.pdf (last access: 07.05.2025).

7. Shah, M. A., & Hellerstein, J. M. (2003). Flux: An adaptive partitioning operator for continuous query systems. Proceedings of the 19th International Conference on Data Engineering. California 94720. Report No. UCB/CSD-2-1205. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2002/CSD-02-1205.pdf.

8. John, V., & Liu, X. (2017). A survey of distributed message broker queues. arXiv preprint arXiv:1704.00411. https://arxiv.org/abs/1704.00411.

9. Lohrmann, B., Janacik, P., & Kao, O. (2015). Elastic stream processing with latency guarantees. In: 2015 IEEE 35th International Conference on Distributed Computing Systems (ICDCS). Columbus, OH, USA, 2015, pp. 399-410, doi: 10.1109/ICDCS.2015.48.

10. Marcu, O. C., Costan, A., Antoniu, G., & Pérez-Hernández, M. S. (2017). Towards a unified storage and ingestion architecture for stream processing. IEEE International Conference on Big Data. Boston, MA, USA 2017, pp. 2402-2407, doi: 10.1109/BigData.2017.8258196.

11. Doan, Q. T. (2021). A Framework for Integrating IoT Streaming Data from Multiple Sources (PhD thesis). La Trobe University. Access mode: https://opal.latrobe.edu.au/articles/thesis/A_Framework_for_Integrating_IoT_Streaming_Data_from_Multiple_Sources/17211713. (last access: 07.05.2025).

12. Zhang, H., Dong, H., Li, G., & Zhang, C. (2024). Elastic buffer-based logging for high-throughput cloud services. Proceedings of the VLDB Endowment, 17(12), 3869–3882. https://www.vldb.org/pvldb/vol17/p3869-chen.pdf.

13. Lee, S., Jeong, Y., Park, K. et al. zStream: towards a low latency micro-batch streaming system. Cluster Comput 26, 2773–2787 (2023). https://doi.org/10.1007/s10586-022-03758-1.

14. Bharti, U., Bajaj, D., Goel, A., & Gupta, S. C. (2019). Identifying requirements for Big Data analytics and mapping

to Hadoop tools. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 5(1), 4384-4392. DOI: 10.35940/ijrte.C5524.098319.

15. Parrinello, A. (2021). Benchmarking materialized views of SQL-based stream processing systems (Master's thesis). Università di Bologna. Access mode: https://amslaurea.unibo.it/id/eprint/30885/1/thesis.pdf. (last access: 07.05.2025).

16. Repository. Wikipedia. [Electronic resource]. – 2024. - Access mode: https://uk.wikipedia.org/wiki/Repository (last access: 07.05.2025).

17. Feature toggle. Wikipedia. [Electronic resource]. – 2024. Access mode: https://uk.wikipedia.org/wiki/Feature_toggle. (last access: 07.05.2025).

18.Producer–consumer problem. Wikipedia. [Electronic resource]. – 2024. Access mode: https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem. (last access: 07.05.2025).

19. Monitoring performance by using the Query Store. Microsoft Docs. [Electronic resource]. – 2024. https://learn.microsoft.com/en-us/sql/relational-databases/performance/monitoring-performance-by-using-the-query-store?view=sql-server-ver16. (last access: 07.05.2025).

20. Семенко, А. І., Бойко, Ю. М., Шпур, О. М., Стрелковська, І. В., Корчинський, В. В., & Яровий, Р. О. (2024). Сучасні технології інфокомунікаційних та комп'ютерних мереж: Монографія (А. І. Семенко, Ред.). Європейський університет, ФО-П Білецький Р. Г., 557 с. https://elar.khmnu.edu.ua/handle/123456789/17381.

21. Пархомей, І., Бойко, Ю., & Лемешко, В. (2025). Математична модель та адаптивне управління алгоритмом обслуговування даних журналу в умовах змінного серверного навантаження. Herald of Khmelnytskyi National University. Technical Sciences, 347(1), 168-174. https://doi.org/10.31891/2307-5732-2025-347-23.

References

1. Smart, T. (2023). Logging and Testing. In: Serverless Beyond the Buzzword. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8761-3_8.

2. Wang, K.C. (2023). Process Management in Embedded Systems. In: Embedded and Real-Time Operating Systems. Springer, Cham. https://doi.org/10.1007/978-3-031-28701-5_5.

3. Sax, M. J. (2020). Performance optimizations and operator semantics for streaming data flow programs. Humboldt-Universität zu Berlin. Access mode: https://edoc.hu-berlin.de/bitstreams/6a57c23e-f7ff-41ec-b202-63d2d01fb1da/download. (last access: 07.05.2025).

4. Sharma, M. K. (2020). A Design Space for Distributed Producer-Consumer Data Structures Using RDMA (Master's thesis). University of Waterloo. Access mode: https://uwspace.uwaterloo.ca/bitstream/10012/16163/4/Sharma_Manoj.pdf. (last access: 07.05.2025).

5. Foyer, C. M. (2021). Abstractions for Portable Data Management in Heterogeneous Memory Systems (PhD dissertation). University of Bristol. Access mode: https://research-information.bris.ac.uk/ws/portalfiles/portal/281444302. (last access: 07.05.2025).

6. Marcu, O. C., Costan, A., Antoniu, G., & Pérez-Hernández, M. S. (2018). Storage and ingestion systems in support of stream processing: A survey. Inria. Access mode: https://inria.hal.science/hal-01939280/file/RT-0501v2.pdf (last access: 07.05.2025).

7. Shah, M. A., & Hellerstein, J. M. (2003). Flux: An adaptive partitioning operator for continuous query systems. Proceedings of the 19th International Conference on Data Engineering. California 94720. Report No. UCB/CSD-2-1205. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2002/CSD-02-1205.pdf.

8. John, V., & Liu, X. (2017). A survey of distributed message broker queues. arXiv preprint arXiv:1704.00411. https://arxiv.org/abs/1704.00411.

9. Lohrmann, B., Janacik, P., & Kao, O. (2015). Elastic stream processing with latency guarantees. In: 2015 IEEE 35th International Conference on Distributed Computing Systems (ICDCS). Columbus, OH, USA, 2015, pp. 399-410, doi: 10.1109/ICDCS.2015.48.

10. Marcu, O. C., Costan, A., Antoniu, G., & Pérez-Hernández, M. S. (2017). Towards a unified storage and ingestion architecture for stream processing. IEEE International Conference on Big Data. Boston, MA, USA 2017, pp. 2402-2407, doi: 10.1109/BigData.2017.8258196.

11. Doan, Q. T. (2021). A Framework for Integrating IoT Streaming Data from Multiple Sources (PhD thesis). La Trobe University. Access mode: https://opal.latrobe.edu.au/articles/thesis/A_Framework_for_Integrating_IoT_Streaming_Data_from_Multiple_Sources/17211713. (last access: 07.05.2025).

12. Zhang, H., Dong, H., Li, G., & Zhang, C. (2024). Elastic buffer-based logging for high-throughput cloud services. Proceedings of the VLDB Endowment, 17(12), 3869–3882. https://www.vldb.org/pvldb/vol17/p3869-chen.pdf.

13. Lee, S., Jeong, Y., Park, K. et al. zStream: towards a low latency micro-batch streaming system. Cluster Comput 26, 2773–2787 (2023). https://doi.org/10.1007/s10586-022-03758-1.

14. Bharti, U., Bajaj, D., Goel, A., & Gupta, S. C. (2019). Identifying requirements for Big Data analytics and mapping to Hadoop tools. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 5(1), 4384-4392. DOI: 10.35940/ijrte.C5524.098319.

15. Parrinello, A. (2021). Benchmarking materialized views of SQL-based stream processing systems (Master's thesis). Università di Bologna. Access mode: https://amslaurea.unibo.it/id/eprint/30885/1/thesis.pdf. (last access: 07.05.2025).

16. Repository. Wikipedia. [Electronic resource]. – 2024. - Access mode: https://uk.wikipedia.org/wiki/Repository (last access: 07.05.2025).

17. Feature toggle. Wikipedia. [Electronic resource]. – 2024. Access mode: https://uk.wikipedia.org/wiki/Feature_toggle. (last access: 07.05.2025).

18.Producer–consumer problem. Wikipedia. [Electronic resource]. – 2024. Access mode: https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem. (last access: 07.05.2025).

19. Monitoring performance by using the Query Store. Microsoft Docs. [Electronic resource]. – 2024. https://learn.microsoft.com/en-us/sql/relational-databases/performance/monitoring-performance-by-using-the-query-store?view=sql-server-ver16. (last access: 07.05.2025).

20. Semenko, A. I., Boiko, J. M., *et al.* (2024). Suchasni tekhnolohii infokomunikatsiinykh ta kompiuternykh merezh: Monohrafiia (A. I. Semenko, Red.). European University, FO-P Biletskyi R. H., 557 s. https://elar.khmnu.edu.ua/handle/123456789/17381.

21. Parkhomey, I., Boiko, J., & Lemeshko, V. (2025). Mathematical model and adaptive control of a data log management algorithm under variable server load conditions. Herald of Khmelnytskyi National University. Technical Sciences, 347(1), 168-174. https://doi.org/10.31891/2307-5732-2025-347-23