УДК 004.6
DOI https://doi.org/10.32782/2663-5941/2025.5.2/42

Шутенко В.М.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

АНАЛІЗ ЕФЕКТИВНОСТІ БАГАТОРІВНЕВОГО КЕШУВАННЯ В РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

Стаття присвячена проблематиці підвищення ефективності управління даними у високонавантажених інформаційних системах шляхом застосування багаторівневого кешування в поєднанні з індексаційними структурами. У статті розкрито передумови зростання актуальності теми, що пов'язані зі стрімким збільшенням обсягів даних та ускладненням архітектури розподілених середовищ, а також з обмеженістю традиційних методів оптимізації доступу до інформації.

Мета дослідження полягає у визначенні оптимальних підходів до використання багаторівневого кешування для забезпечення продуктивності, масштабованості та стійкості систем до пікових навантажень. Методологічною основою дослідження визначено системний аналіз архітектурних рішень, порівняльний аналіз стратегій кешування та індексації, а також моделювання типових сценаріїв обробки даних, що дозволило виявити закономірності їх взаємодії.

З'ясовано, що ефективність функціонування систем значною мірою залежить від правильного поєднання кешування та індексації. Доведено, що застосування write-through і refresh-ahead забезпечує баланс між консистентністю та швидкодією у критично важливих сервісах, тоді як write-back є доцільним у середовищах аналітичної обробки. Встановлено наявність проблем інвалідності кешу та дисбалансу навантаження, що потребують використання гібридних стратегій.

Наукова новизна полягає у формулюванні моделі взаємодії багаторівневого кешування та індексації, яка мінімізує промахи кешу та оптимізує час відповіді системи. Уперше обґрунтовано інтеграцію прогнозних алгоритмів у механізми кешування, що відкриває можливості самонавчальної оптимізації архітектур.

У висновках доведено, що інтегрований підхід, заснований на поєднанні багаторівневого кешування, адаптивних стратегій витіснення, методів індексації та інструментів моніторингу, забезпечує зниження латентності, підвищення відмовостійкості та стабільність роботи систем. Перспективи подальших досліджень пов'язано з розробкою гібридних архітектур кешування та створенням самонавчальних систем управління, здатних у реальному часі адаптуватися до змін навантаження.

Ключові слова: *продуктивність систем, узгодженість даних, адаптивні алгоритми, розподілені обчислення, оптимізація доступу до інформації.*

Постановка проблеми. У сучасних умовах стрімкого зростання обсягів даних та ускладнення архітектури інформаційних систем постає проблема забезпечення їхньої стабільної роботи в режимі високих навантажень. Традиційні підходи до управління даними виявляються недостатньо ефективними, оскільки зростає кількість одночасних запитів, потреба в оперативному доступі до критично важливої інформації та вимоги до мінімізації затримок. У цьому контексті особливої актуальності набуває багаторівневе кешування, яке дозволяє оптимізувати використання обчислювальних ресурсів і скоротити час відповіді системи. Проте сама наявність кешу не гарантує очікуваного ефекту, адже ефективність його роботи визначається алгоритмами управління, структурою індексів та здатністю системи адаптуватися до динамічних змін у потоках даних.

Наукова проблема полягає в пошуку та обґрунтуванні методів, що забезпечують баланс між продуктивністю, узгодженістю даних і масштабованістю розподілених систем. Вона має безпосередній зв'язок із важливими прикладними завданнями, серед яких створення високонадійних сервісів електронної комерції, фінансових платформ, систем реального часу та аналітичних інструментів для роботи з великими масивами інформації. Вирішення цієї проблеми сприятиме розвитку технологій обробки даних у напрямах, що поєднують ефективні моделі кешування з інноваційними методами індексації, що, у свою чергу, забезпечить підвищення продуктивності інформаційних сис-

тем та їхню здатність функціонувати в умовах постійно зростаючих навантажень.

Аналіз останніх досліджень і публікацій. Аналіз сучасних досліджень ефективності багаторівневого кешування в розподілених інформаційних системах свідчить про формування чотирьох взаємопов'язаних напрямів, які охоплюють архітектурні рішення для високонавантажених сервісів, міжрівневе розміщення та доменно-специфічні розширення, системно-мережеві та теоретичні засади, а також edge-сценарії й довірчі механізми перевірки даних.

Перший напрям стосується архітектурної еволюції кешування для високонавантажених мікросервісів і розподілених СУБД. Б. Волох обґрунтовує доцільність інтелектуалізації політик заміщення та інвалідизації порівняно з класичними LRU/LFU/TTL, показуючи, що продуктивність і узгодженість досягаються через контекстно-залежне керування термінами придатності та подієвою інвалідацією [1]. Г. Г. Киричек, М. Ю. Тягунова, В. В. Братчиков демонструють, що у розгалуженій мікросервісній архітектурі ефективність забезпечує ієрархія кешів (процесний, боковий, на межовому шлюзі, на рівні CDN) у зв'язці з подієвими шинами, що мінімізує міжсервісну латентність без зсувів узгодженості [2]. Ф. Лу (F. Lu), З. Ши (Z. Shi), Л. Гу (L. Gu) та співавтори пропонують адаптивну багаторівневу стратегію для розподілених СУБД з динамічним переміщенням «гарячих» ключів між рівнями пам'яті та вузлами, що стабілізує частку влучень і скорочує хвости затримок під час зміни шаблонів навантаження [3]. Перспективним є подальше дослідження SLO-орієнтованого профілювання кешів для окремих сервісів із автоматизованим налаштуванням TTL/рефрешу на основі телеметрії запитів і прогнозування сплесків.

Другий напрям охоплює розширення парадигми багаторівневого кешування в суміжні домени та скоординоване міжрівневе розміщення. Дж. В. Хео (J. W. Heo), Г. С. Рамачандран (G. S. Ramachandran), А. Доррі (A. Dorri) та співавтори показують, що багаторівневе розподілене кешування навколо блокчейна істотно знижує тиск на ланцюг і вартість зберігання, зберігаючи можливість перевірки й відтворення стану [4]. А. С. Мірафтаб (A. S. Miraftab), А. Монтазеролгхаем (A. Montazerolghaem), Б. Махбообі (B. Mahboobi) для content-centric мереж обґрунтовують децентралізоване кодоване кешування з урахуванням багаторівневої популярності та неоднорідного доступу, що балансує пропускну здатність і затримку [5]. М. Т. Іслам (M. T. Islam), Р. Боровіца-Гайїч (R. Borovica-Gajic), С. Карунасекера (S. Karunasekera) описують архітектуру багаторівневого кеша для станної потокової обробки, у якій локальні й розподілені стани кешуються вздовж графа обчислень, зменшуючи витрати на відновлення і міжвузлові комунікації [6]. З. Зенг (Z. Zeng), І. Тан (Y. Tan), З. Ма (Z. Ma) та співавтори пропонують адаптивне cross-level розміщення, що усуває взаємні інвалідизації між рівнями ієрархії та підвищує ефективність у цілому [7]. Перспективним є розвиток ML-керованих оркестраторів розміщення з урахуванням вартості пам'яті/мережі та нестаціонарності попиту, а також формування методик стабільності таких політик.

Третій напрям пов'язаний із системно-мережевою оптикою та теоретичними й файловими засадами. М. Пасєка (M. Pasieka), В. Шекета (V. Sheketa), Н. Пасєка (N. Pasieka) та співавтори проводять системний аналіз кеш-запитів на мережевих обчислювальних вузлах і показують вплив маршрутизації й агрегації на «гарячі точки» та пропускну здатність [8]. Л. Лв (L. Lv), Х. Ду (H. Du) узагальнюють інженерні шаблони розподіленої багаторівневої архітектури та практики її впровадження у виробничих системах [9]. Дж. В. Хео (J. W. Heo), А. Доррі (A. Dorri), Р. Джурдак (R. Jurdak) у контексті блокчейн-екосистем конкретизують «off-chain-first» підхід, який пізніше деталізовано у прикладних сценаріях оптимізації зберігання [10]. А. Налаяла (A. Nalajala), Т. Рагунатан (T. Ragunathan), Р. Гопісетті (R. Gopisetty) та співавтори демонструють, що ранжований префетчинг у зв'язці з багаторівневим кешем підвищує ефективність читання в розподілених файлових системах [11]. Т. Ло (T. Luo), В. Аггарвал (V. Aggarwal), Б. Пелеато (B. Peleato) теоретично показують переваги кодованого кешування з розподіленим зберіганням, яке знижує трафік доставки та збалансовує навантаження на вузли [12]. Перспективним є поєднання кодованих схем із мережево-обізнаними планувальниками запитів на рівні кластера та створення відтворюваних бенчмарків, які одночасно фіксують latency/throughput/tail, частку влучень і частоту порушень узгодженості.

Четвертий напрям охоплює edge/vehicular-сценарії та довірчі механізми перевірки даних. Т. Солеймані (T. Soleymani), Н. Яздані (N. Yazdani), С. П. Шаріатпанахі (S. P. Shariatpanahi) застосовують багаторівневі кеш-стратегії на основі глибинного підкріплювального навчання у тран-

спортних мережах із високою мобільністю [13]. М. С. Аль-Абіад (M. S. Al-Abiad), М. З. Хассан (M. Z. Hassan), М. Дж. Хоссейн (M. J. Hossain) показують, що комбінація мережевого кодування з багаторівневими кешами в гетерогенних мережах максимізує пропускну здатність і зменшує інтерференцію завдяки ефективнішому мультикасту [14]. К. Чжан (Q. Zhang), С. Лі (C. Li), Т. Ду (T. Du) та співавтори інтегрують багаторівневе кешування з криптографічною верифікацією даних в екосистемі Ethereum, демонструючи можливість зниження затримки читання без компромісів щодо цілісності [15]. Перспективним є розроблення безпечних протоколів інвалідизації та перевірюваного кешування для середовищ із відкладеною фіналізацією транзакцій і агресивними TTL, а також спільних методик оцінювання на периферії.

Попри значний прогрес у вивченні механізмів кешування, досі залишаються невирішеними низка аспектів, які стримують розвиток теорії та практики у сфері високонавантажених інформаційних систем. Зокрема, недостатньо обґрунтовано місце багаторівневого кешування в архітектурі розподілених середовищ, бракує комплексних досліджень впливу різних стратегій кешування на баланс між швидкодією та консистентністю даних. Відкритими залишаються питання взаємодії кешування з методами індексації, особливо у контексті роботи з великими обсягами інформації, а також проблеми масштабування, що супроводжуються ефектом кеш-інвалідності, дисбалансом навантаження та зниженням відмовостійкості. Обмеженою є також емпірична база, яка здебільшого представлена прикладними дослідженнями окремих платформ без системного узагальнення.

Запропоноване дослідження спрямоване на подолання цих прогалин шляхом формування рекомендацій щодо взаємодії кешування та індексації, розробки адаптивних стратегій кешування для різних класів даних, а також аналізу проблем масштабування з позицій їхнього впливу на узгодженість і продуктивність. Очікується, що результати дозволять не лише уточнити теоретичні основи застосування багаторівневого кешування у розподілених системах, але й сформувати практичні рекомендації щодо його оптимізації.

Мета статті полягає у визначенні ефективних напрямів застосування багаторівневого кешування в розподілених інформаційних системах для забезпечення їх продуктивності та стійкості до високих навантажень.

Завдання статті:

1. Обґрунтувати місце багаторівневого кешування в архітектурі розподілених систем та його взаємозв'язок з індексаційними механізмами.

2. Дослідити вплив стратегій кешування на продуктивність і консистентність даних та виявити ключові обмеження масштабування.

3. Сформувати науково обґрунтовані рекомендації з оптимізації кешування для підвищення ефективності управління даними.

Виклад основного матеріалу. Багаторівневе кешування поступово перетворюється на один із ключових елементів архітектури розподілених інформаційних систем, оскільки воно дозволяє не лише зменшити затримки при доступі до даних, але й суттєво оптимізувати використання апаратних ресурсів. Його місце визначається функціональною роллю у зниженні навантаження на бази даних та мережеву інфраструктуру, що особливо важливо для високонавантажених систем, де одночасно обробляються мільйони запитів. Завдяки багаторівневій структурі кешування вдається побудувати ієрархію проміжних сховищ, які максимально наближені до користувача або до обчислювальних вузлів, що забезпечує гнучке балансування між швидкодією, узгодженістю та вартістю зберігання інформації (табл. 1).

Застосування багаторівневого кешування в архітектурі розподілених інформаційних систем

Таблиця 1

Рівні багаторівневого кешування в архітектурі розподілених систем

Рівень кешу	Розташування	Призначення	Приклади реалізацій
L1 – локальний	У межах процесу/додатка	Швидкий доступ до даних без мережевих звернень	In-memory cache (Ehcache, Caffeine)
L2 – міжсервісний	На рівні серверного вузла	Спільне використання кешованих даних кількома сервісами	Redis, Memcached
L3 – мережевий	На рівні мережевої інфраструктури	Зменшення затримок при віддаленому доступі, розвантаження ЦОД	CDN, проксі-сервери (Varnish, Nginx cache)

Джерело: сформовано автором на основі [1; 2; 3; 6; 7; 9].

доцільно розглядати як один із базових механізмів підтримання продуктивності, масштабованості та стійкості до збоїв. Його наукова цінність полягає в тому, що кешування функціонує як додатковий рівень абстракції між користувацькими запитами та первинними сховищами даних, забезпечуючи баланс між швидкодією та узгодженістю. На практиці це означає, що обчислювальні ресурси можуть бути оптимізовані завдяки зменшенню частоти звернень до основної бази даних і мережевої інфраструктури, що у свою чергу мінімізує латентність (затримку відповіді системи) та підвищує надійність обслуговування великої кількості паралельних користувачів.

На локальному рівні (L1) кешування здійснюється безпосередньо у межах додатка за допомогою структур оперативної пам'яті. Це дозволяє повторно використовувати результати дорогих обчислювальних операцій, наприклад, при формуванні динамічних сторінок у системах електронної комерції. Такі рішення реалізуються через високоефективні бібліотеки на кшталт Ehcache чи Caffeine, що здатні адаптувати політики витіснення до поведінки користувачів.

На міжсервісному рівні (L2) використовується централізоване сховище проміжних даних, яке обслуговує кілька додатків або сервісів одночасно. Тут поширеним рішенням є Redis (Remote Dictionary Server) та Memcached, які інтегруються у мікросервісні архітектури для зниження навантаження на реляційні бази даних. Практичним прикладом є фінансові транзакційні системи, де Redis-кластер виконує роль швидкісного буфера для результатів обробки запитів, що дозволяє скорочувати кількість звернень до основних таблиць обліку та забезпечувати стабільний час відповіді навіть у періоди пікових навантажень.

На мережевому рівні (L3) кешування реалізується за допомогою інфраструктурних рішень, зокрема Content Delivery Network (CDN) – мереж доставки контенту. Використання CDN, таких як Cloudflare чи Akamai, дає змогу зберігати копії статичного та мультимедійного контенту у географічно розподілених дата-центрах, забезпечуючи мінімізацію затримок і рівномірний розподіл навантаження. Прикладом є діяльність компанії Netflix, яка через власну платформу Open Connect Appliance (OCA) поєднує CDN із локальним кешуванням, гарантуючи стабільну якість потокового відео для мільйонів користувачів у реальному часі [16].

Так, багаторівневе кешування виступає не лише технічним інструментом оптимізації запитів, але й концептуальною моделлю керування потоками даних у масштабованих системах. У поєднанні з алгоритмами індексації воно створює підґрунтя для формування самонавчальних і адаптивних архітектур, здатних ефективно функціонувати в умовах стрімкого зростання обсягів інформації та вимог до швидкості її обробки.

Дослідження впливу різних стратегій кешування на швидкодію та узгодженість даних має ключове значення для побудови високонадійних розподілених інформаційних систем. Кожна стратегія визначає спосіб взаємодії кешу з основною базою даних, формуючи специфічні характеристики системи – від латентності доступу до рівня гарантій консистентності. Саме правильний вибір і комбінування цих стратегій дозволяють забезпечити масштабованість та стабільність роботи у середовищах із мільйонними обсягами транзакцій (табл. 2).

У розподілених інформаційних системах стратегії кешування формують різні парадигми управ-

Таблиця 2

Характеристика стратегій кешування у розподілених інформаційних системах

Стратегія кешування	Механізм взаємодії з базою даних	Характерні особливості	Типові сфери застосування
Write-through	Запис одночасно у кеш і базу даних	Гарантує узгодженість; вища затримка при записі	Фінансові системи, банківські додатки
Write-back (write-behind)	Запис у кеш, база оновлюється асинхронно	Мінімізація затримки запису; ризик втрати даних при збої	Аналітика подій, системи логування
Write-around	Запис напряму у базу, кеш оновлюється при читанні	Зменшення «засмічення» кешу рідко використовуваними даними	Сховища «холодних» даних
Read-through	Автоматичне завантаження у кеш при запиті	Простота реалізації; залежність від політики оновлення	Веб-додатки з високим навантаженням
Refresh-ahead	Прогнозне оновлення кешу до звернення	Зменшення латентності читання; складність у прогнозуванні	Стрімінгові сервіси, системи рекомендацій

Джерело: сформовано автором на основі [1; 2; 5; 11; 12].

ління даними, які по-різному впливають на часові характеристики доступу та гарантії узгодженості. Стратегія write-through забезпечує транзакційну надійність завдяки синхронному оновленню кешу й бази даних, що унеможливлює розходження між ними. Такий підхід застосовується у фінансових сервісах, де навіть мілісекундні розбіжності у даних рахунку можуть створити системні ризики. Write-back, навпаки, орієнтований на асинхронність: дані спершу зберігаються у кеші, а потім – у базі, що знижує затримку операцій, але створює потребу у додаткових механізмах контролю на випадок збоїв. Це актуально для платформ потокової аналітики, де головним критерієм є пропускна здатність, а не абсолютна синхронність.

У випадку write-around кешування використовується вибірково: рідко потрібні дані одразу йдуть у базу, а кеш завантажується лише при запиті. Такий підхід характерний для систем з великими масивами «холодної» інформації, як-от архіви медичних зображень чи судових документів, де кешування усіх даних було б економічно недоцільним. Стратегія read-through оптимізує обробку часто повторюваних звернень: дані автоматично завантажуються у кеш під час першого запиту, що істотно знижує кількість звернень до сховища. Прикладом є системи електронної комерції, де мільйони користувачів одночасно переглядають однакові каталоги товарів. Refresh-ahead реалізує прогностичну логіку: система завчасно оновлює кеш, виходячи з передбачуваних звернень. Цей підхід активно застосовують мультимедійні платформи, наприклад, у стрімінгових сервісах кешування метаданих і рекомендацій здійснюється ще до фактичного запиту, що гарантує безперервність користувацького досвіду.

Загалом, вплив стратегій кешування визначається не лише технічними характеристиками, а й галузевим контекстом. Там, де ключовим є забезпечення консистентності, переважають синхронні моделі, тоді як у сферах із високою інтенсивністю обробки даних – асинхронні та прогнозні підходи. Це створює підґрунтя для гібридних рішень, де для різних класів даних поєднуються різні стратегії, формуючи адаптивні архітектури з динамічним балансом між швидкодією та узгодженістю.

У високонавантажених інформаційних системах взаємозв'язок кешування та індексації визначає не лише швидкодію, а й архітектурну стійкість обробки даних. Кешування дає змогу повторно використовувати результати запитів без прямого звернення до основної бази, тоді як індексація структурує дані у спосіб, що скорочує час пошуку та знижує ймовірність «промахів» кешу. Ці два механізми функціонують у тісній взаємодії: правильний вибір індексаційної структури створює оптимізовані набори даних для збереження у кеші, тоді як ефективне кешування зменшує навантаження на індекси, подовжуючи їхній життєвий цикл у системі (табл. 3).

Поєднання кешування з індексаційними структурами формує основу сучасних високонавантажених сервісів. У банківських транзакційних системах, де важливі діапазонні запити до рахунків, B-tree індекси дозволяють швидко локалізувати потрібні записи, а результати операцій кешуються для повторного використання клієнтами, що

Таблиця 3

Взаємодія стратегій кешування та методів індексації

Метод індексації	Характеристика індексу	Взаємодія з кешуванням	Типові сценарії застосування
B-tree (Balanced tree)	Балансоване дерево для діапазонних запитів	Формує ефективні кешовані діапазони даних	Банківські транзакційні системи, OLTP (Online Transaction Processing)
LSM-tree (Log-Structured Merge-tree)	Індекс для швидкого запису та злиття даних у рівнях	Зменшує промахи кешу при масових потоках запису	NoSQL бази даних (Apache Cassandra, HBase), журнали подій
Inverted index	Індекс, що зіставляє ключові терміни з документами	Забезпечує швидке формування кешованих результатів пошуку	Пошукові системи (Elasticsearch, Apache Solr), аналітика текстів
Bitmap index	Використання бітових масивів для категоріальних атрибутів	Створює компактні кешовані зрізи для запитів	Сховища даних, BI (Business Intelligence) системи
Hash index	Пряме відображення ключа у позицію зберігання	Прискорює кешування у форматі ключ–значення	Розподілені кеші (Redis, Memcached), системи реального часу

Джерело: сформовано автором на основі [3; 6; 7; 11; 12].

істотно знижує час відповіді. У NoSQL (Not only SQL) середовищах на кшталт Apache Cassandra застосування LSM-tree дозволяє оптимізувати масові потоки запису: дані тимчасово зберігаються у пам'яті та кешуються, а фонові процеси виконують їх консолідацію, знижуючи навантаження на диск. Пошукові системи, зокрема Elasticsearch, використовують інверсні індекси для формування списків релевантних документів, які одразу кешуються, забезпечуючи мілісекундний відгук при повторних запитах мільйонів користувачів. У сховищах даних, де необхідно миттєво будувати зрізи для аналітики, bitmap-індекси у поєднанні з кешуванням створюють компактні й швидкодоступні агрегати даних. Нарешті, у розподілених кешах на кшталт Redis (Remote Dictionary Server) чи Memcached hash-індекси поєднуються з механізмом кешування ключ–значення, що дозволяє стабільно обслуговувати потоки у сотні тисяч запитів за секунду.

Так, індексація виступає як «інтелектуальний провідник» до даних, тоді як кешування виконує роль «швидкісного буфера», і лише їхня скоординована взаємодія забезпечує масштабованість і надійність обробки великих інформаційних потоків. Сучасні системи тяжіють до гібридних рішень, де різні типи індексів інтегруються з багаторівневим кешуванням, створюючи адаптивні архітектури, здатні ефективно функціонувати у динамічних умовах глобальних обчислювальних мереж.

Масштабування багаторівневого кешування у високонавантажених інформаційних системах супроводжується низкою фундаментальних проблем і технічних обмежень, які суттєво впливають на стабільність і передбачуваність роботи таких архітектур. Однією з ключових проблем є узгодженість даних: при наявності декількох рівнів кешу виникають ризики застарілих копій, особливо у розподілених середовищах з географічно віддаленими дата-центрами, де затримки реплікації унеможливлюють одночасне оновлення всіх кешованих об'єктів [1; 2]. Це породжує ефект кеш-інвалідності, коли користувач отримує недостовірні дані, а механізми оновлення споживають додаткові ресурси.

Другою проблемою є управління латентністю при збільшенні кількості рівнів кешу. Замість очікуваного прискорення може виникати ситуація надмірних накладних витрат на перевірку кожного рівня, що збільшує час відповіді системи у порівнянні з прямим доступом до бази [7]. Тісно з цим пов'язане питання вибору політик витіснення: традиційні алгоритми, як-от LRU (Least Recently Used), не завжди ефективні у високонавантажених системах з нерівномірним доступом до даних, оскільки вони можуть витісняти критично важливі елементи через короткочасні піки запитів [5].

Проблемою також є балансування навантаження у розподілених кешах. При масштабуванні кластера спостерігається явище «гарячих ключів», коли певні дані запитуються набагато частіше за інші, створюючи локальні перевантаження та знижуючи загальну пропускну здатність [8]. Використання хеш-функцій для розподілу ключів частково розв'язує цю проблему, проте не гарантує рівномірності при зміні структури кластера. Суттєвим обмеженням виступає і керування пам'яттю. Зі збільшенням обсягів кешу у пам'яті сервера зростають витрати на його обслуговування, включно з фоновими процесами очищення та інвалідизації. Це може призвести до конкуренції між обробкою користувацьких запитів та системними процесами підтримки кешу, що особливо відчутно у системах реального часу [1]. Не менш критичним є питання відмовостійкості. У випадку збою вузлів кешу можливі значні втрати продуктивності через лавиноподібне перенаправлення навантаження на базу даних. Відновлення стану кешу після таких збоїв потребує додаткового часу та ресурсів, що унеможливлює безперервність сервісу [4; 10] Ще однією проблемою є ускладнення механізмів моніторингу й діагностики. У багаторівневих архітектурах важко точно визначити джерело затримки чи помилки: чи вона виникла через промах кешу, надмірну кількість реплікацій, неефективну політику витіснення чи перевантаження окремого вузла [2]. Відсутність прозорості у ланцюзі запитів ускладнює оптимізацію та підвищує витрати на експлуатацію.

Додатковим обмеженням стає і мережевий трафік. При великій кількості рівнів кешу та географічно розподілених вузлах постійна синхронізація споживає значні мережеві ресурси, що може призводити до затримок у критично важливих операціях. Для мультимедійних сервісів або фінансових платформ це означає втрату якості обслуговування або навіть ризики транзакційних помилок.

Оптимізація використання кешування у високонавантажених інформаційних системах потребує поєднання архітектурних рішень, алгоритмічних підходів та механізмів адаптації до змінних умов навантаження. Ефективність управління даними значною мірою залежить від того, наскільки

гнучко система здатна поєднувати кешування з іншими методами оптимізації, зокрема індексацією, балансуванням навантаження та механізмами узгодженості. Одним із ключових напрямів є впровадження адаптивних стратегій кешування, які динамічно коригують політику зберігання залежно від характеру потоків даних і профілю користувацьких запитів. Такий підхід дозволяє уникнути надмірного використання пам'яті для рідко використовуваних елементів і водночас забезпечує високу доступність даних, що активно циркулюють у системі.

Перспективним рішенням є інтеграція інтелектуальних алгоритмів прогнозування на основі машинного навчання, які здатні визначати «гарячі» ділянки даних та завчасно формувати відповідні кеші. Це особливо актуально для систем реального часу, таких як платформи потокового відео чи фінансові аналітичні сервіси, де ймовірність запитів до певних даних можна передбачати з високою точністю. Крім того, значну роль відіграє узгодження механізмів кешування із стратегіями індексації. Створення багаторівневої моделі, у якій результати індексованих пошуків одразу потрапляють до кешу, дозволяє не лише підвищити швидкодію, але й знизити кількість промахів у випадках масових звернень.

Суттєвим чинником оптимізації є підтримка відмовостійкості й масштабованості. Для цього доцільно застосовувати реплікацію кешів у розподілених кластерах, використання механізмів sharding (розбиття даних на сегменти) та впровадження протоколів узгодженості, здатних зберігати баланс між продуктивністю та коректністю даних. Також актуальним напрямом є інтеграція кешування з інфраструктурою Content Delivery Network, що дає можливість не лише пришвидшувати доступ до статичного контенту, але й зменшувати навантаження на центральні бази даних у глобально розподілених системах.

Важливим завданням є й розвиток системного моніторингу, які дозволяють відстежувати ефективність політик кешування, аналізувати промахи та визначати вузькі місця. У поєднанні з інтелектуальними механізмами керування такі системи здатні автоматично змінювати стратегії залежно від поточного стану середовища. У результаті формуються адаптивні архітектури управління даними, які забезпечують стійку продуктивність у різних сценаріях навантаження та дозволяють мінімізувати ризики, пов'язані з узгодженістю й відмовостійкістю.

Таким чином, оптимізація кешування не зводиться лише до вибору алгоритмів витіснення чи збільшення обсягів оперативної пам'яті. Вона передбачає системний підхід, у якому кешування стає частиною інтегрованої стратегії управління даними, поєднаної з прогнозними моделями, індексацією, балансуванням та відмовостійкими механізмами

Висновки. У ході дослідження встановлено, що багаторівневе кешування є базовим механізмом підвищення продуктивності розподілених інформаційних систем, оскільки воно забезпечує зменшення латентності, оптимізацію використання ресурсів і балансування навантаження між вузлами. Показано, що ефективність кешування визначається його узгодженістю з методами індексації, коректним вибором стратегій управління даними та здатністю архітектури адаптуватися до динамічних потоків транзакцій. Серед виявлених проблем ключовими є труднощі забезпечення консистентності в багаторівневих структурах, виникнення ефекту кеш-інвалідності, дисбаланс навантаження у випадку «гарячих ключів», а також значні накладні витрати на обслуговування великих обсягів проміжної пам'яті. Ускладнює ситуацію й необхідність підтримки відмовостійкості, оскільки збої окремих вузлів можуть спричиняти лавиноподібне зростання навантаження на бази даних. Рекомендовано застосовувати адаптивні стратегії кешування, інтегрувати прогнозні алгоритми машинного навчання для визначення критично важливих даних, поєднувати кешування з індексаційними структурами та впроваджувати механізми шардінгу й реплікації для підвищення надійності. Важливою умовою оптимізації є також розвиток інтелектуальних систем моніторингу, здатних автоматично змінювати політики кешування залежно від реального стану середовища. Перспективи подальших досліджень пов'язані з формуванням гібридних архітектур, що комбінують різні стратегії кешування та методи індексації для окремих класів даних, а також зі створенням самонавчальних систем управління потоками інформації. Це дозволить забезпечити масштабованість, стійкість і високу якість обслуговування у середовищах із глобальними обсягами даних.

Список літератури:

1. Волох Б. Інтелектуальне кешування у високонавантажених системах як відповідь на обмеження класичних методів керування кешем. Шляхи підвищення ефективності будівництва. 2023. Вип. 3, № 52. С. 227–236. DOI: https://doi.org/10.32347/2707-501x.2023.52(3).227-236.

2. Киричек Г. Г., Тягунова М. Ю., Братчиков В. В. Система кешування даних в розгалуженій мікросервісній архітектурі. Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. 2024. Т. 35, Вип. 74, № 1. С. 141–146. DOI: https://doi.org/10.32782/2663-5941/2024.1.1/21.

3. Lu F., Shi Z., Gu L., Jin H., Yang L. T. An adaptive multi-level caching strategy for distributed database system. Future Generation Computer Systems. 2019. Vol. 97. P. 61–68. DOI: https://doi.org/10.1016/j.future.2018.11.050.

4. Heo J. W., Ramachandran G. S., Dorri A., Jurdak R. Blockchain storage optimisation with multi-level distributed caching. IEEE Transactions on Network and Service Management. 2022. Vol. 19, № 4. P. 3724–3736. DOI: https://doi.org/10.1109/TNSM.2022.3224735.

5. Miraftab A. S., Montazerolghaem A., Mahboobi B. Improving performance of content-centric networks via decentralized coded caching for multi-level popularity and access. Cluster Computing. 2025. Vol. 28. Article 458. DOI: https://doi.org/10.1007/s10586-025-05256-6.

6. Islam M. T., Borovica-Gajic R., Karunasekera S. A multi-level caching architecture for stateful stream computation. Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. 2022. P. 67–78. DOI: https://doi.org/10.1145/3524860.3539803.

7. Zeng Z., Tan Y., Ma Z., Li J., Zhao S., Liu D., Ren A. CMCache: An adaptive cross-level data placement method for multi-level cache. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2025. Vol. 44, № 8. P. 2911–2924. DOI: https://doi.org/10.1109/TCAD.2025.3534116.

8. Pasieka M., Sheketa V., Pasieka N., Chupakhina S., Dronyuk I. System analysis of caching requests on network computing nodes. 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT). Lviv, Ukraine, 2019. P. 216–222. DOI: https://doi.org/10.1109/AIACT.2019.8847909.

9. Lv L., Du H. Research and application on distributed multi-level cache architecture. 2022 11th International Conference of Information and Communication Technology (ICTech). Wuhan, China, 2022. P. 138–143. DOI: https://doi.org/10.1109/ICTech55460.2022.00035.

10. Heo J. W., Dorri A., Jurdak R. Multi-level distributed caching on the blockchain for storage optimisation. 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). 2022. P. 1–5. DOI: https://doi.org/10.1109/ICBC54727.2022.9805518.

11. Nalajala A., Ragunathan T., Gopisetty R., Garrapally V. Rank-based prefetching and multi-level caching algorithms to improve the efficiency of read operations in distributed file systems. In: Srirama S. N., Lin J. C. W., Bhatnagar R., Agarwal S., Reddy P. K. (eds). Big Data Analytics. BDA 2021. Lecture Notes in Computer Science. Vol. 13147. Springer, Cham. 2021. P. 231–243. DOI: https://doi.org/10.1007/978-3-030-93620-4_17.

12. Luo T., Aggarwal V., Peleato B. Coded caching with distributed storage. IEEE Transactions on Information Theory. 2019. Vol. 65, № 12. P. 7742–7755. DOI: https://doi.org/10.1109/TIT.2019.2940979.

13. Soleymani T., Yazdani N., Shariatpanahi S. P. Multi-level deep reinforcement learning-based edge caching strategies in vehicular networks. 2024 11th International Symposium on Telecommunications (IST). Tehran, Iran, 2024. P. 690–696. DOI: https://doi.org/10.1109/IST64061.2024.10843640.

14. Al-Abiad M. S., Hassan M. Z., Hossain M. J. Throughput maximization of network-coded and multi-level cache-enabled heterogeneous network. IEEE Transactions on Vehicular Technology. 2021. Vol. 70, № 10. P. 11039–11043. DOI: https://doi.org/10.1109/TVT.2021.3106148.

15. Zhang Q., Li C., Du T., Luo Y. Multi-level caching and data verification based on ethereum blockchain. Wireless Networks. 2023. Vol. 29. P. 713–727. DOI: https://doi.org/10.1007/s11276-022-03151-1

16. Netflix Open Connect. Netflix: website. URL: https://openconnect.netflix.com (date of access: 02.09.2025).

Shutenko V.M. ANALYSIS OF THE EFFICIENCY OF MULTI-LEVEL CACHING IN DISTRIBUTED INFORMATION SYSTEMS

The article is devoted to the problem of improving data management efficiency in high-load information systems through the use of multi-level caching combined with indexing structures. The article reveals the prerequisites for the growing relevance of the topic, which are associated with the rapid increase in data volumes and the complication of distributed system architectures, as well as the limitations of traditional methods of optimizing access to information.

The purpose of the study is to determine the optimal approaches to the use of multi-level caching to ensure system performance, scalability, and resilience to peak loads. The methodological basis of the research is defined as system analysis of architectural solutions, comparative analysis of caching and indexing strategies, as well as modeling of typical data processing scenarios, which made it possible to identify patterns of their interaction.

It has been found that the efficiency of system functioning largely depends on the proper combination of caching and indexing. It has been proved that the use of write-through and refresh-ahead strategies ensures a balance between consistency and speed in mission-critical services, while write-back is appropriate for

analytical environments. The existence of cache invalidation issues and load imbalance has been established, which requires the use of hybrid strategies.

The scientific novelty lies in the formulation of a model of interaction between multi-level caching and indexing that minimizes cache misses and optimizes system response time. For the first time, the integration of predictive algorithms into caching mechanisms has been substantiated, which opens up opportunities for self-learning optimization of system architectures.

The conclusions prove that an integrated approach based on the combination of multi-level caching, adaptive eviction strategies, indexing methods, and monitoring tools ensures reduced latency, improved fault tolerance, and stable system operation. Prospects for further research are related to the development of hybrid caching architectures and the creation of self-learning management systems capable of real-time adaptation to workload changes.

Key words: *system performance, data consistency, adaptive algorithms, distributed computing, information access optimization.*

Дата надходження статті: 03.09.2025
Дата прийняття статті: 19.09.2025
Опубліковано: 16.12.2025