УДК 004.05-048.34:004.254
DOI: 10.31673/2412-9070.2025.027500
О. А. ЗОЛОТУХІНА[1], канд. техн. наук, доцент;
ORCID: 0000-0002-3314-417X
Б. О. ХУДІК[2], PhD;
ORCID: 0009-0005-1914-6418
А. С. ГАВОР[2], асистент,
ORCID: 0009-0002-9705-1666
[1] Київський національний університет імені Тараса Шевченко
[2] Державний університет інформаційно-комунікаційних технологій, Київ

ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ КЕШУВАННЯ ЗА ДОПОМОГОЮ ГІБРИДНОГО МЕТОДА КЕШУВАННЯ ДАНИХ

Кешування даних відіграє ключову роль у покращенні продуктивності та швидкості доступу до часто запитуваних ресурсів. Використання різних типів та методів кешування є вирішальним фактором для оптимальної продуктивності та надійності систем.

Ця технологія застосовується для зберігання ресурсів, які часто використовуються, у оперативні пам'яті, на диску або в гібридних системах, що поєднують обидва підходи. Використання оперативної пам'яті забезпечує високу швидкість доступу до даних, тоді як кешування на диску дозволяє зберігати більші обсяги даних. Гібридні системи об'єднують переваги обох методів, досягаючи балансу між швидкістю та обсягом зберігання.

Особливу увагу приділено алгоритмам кешування, що забезпечують ефективне управління даними в кеші. Розглянуто популярні алгоритми, такі як Least Recently Used (LRU), Least Frequently Used (LFU), First In, First Out (FIFO), Adaptive Replacement Cache (ARC) та Most Recently Used (MRU). Ці алгоритми аналізуються в контексті їх застосування для оптимізації продуктивності систем кешування.

Досліджено переваги та недоліки використання зазначених алгоритмів у різних сценаріях. Увага приділяється вирішенню проблем оптимізації розміру кешу, зниженню затримок при доступі до даних та підвищенню ефективності використання ресурсів. Запропоновано математичні моделі та методи аналізу продуктивності кешування, що дозволяють оцінити ефективність різних алгоритмів та оптимізувати налаштування систем кешування для досягнення максимальної продуктивності.

Запропоновано та реалізовано гібридний метод кешування, який поєднує алгоритми LRU та MRU шляхом динамічного перемикання між ними на основі аналізу дисперсії частоти доступу до даних. Цей підхід передбачає обчислення статистичних характеристик доступу до даних, що дозволяє системі адаптивно обирати найбільш підходящий алгоритм кешування в реальному часі. Використання цього методу дозволило підвищити продуктивність та ефективність кешування, зменшити кількість кеш-промахів та покращити загальну пропускну здатність системи.

Ключові слова: кешування даних, продуктивність, інформаційні системи, алгоритми кешування, гібридні системи, розподілене кешування, локальне кешування, оптимізація, аналіз продуктивності.

Вступ

У сучасному світі технологій кешування даних відіграє ключову роль у забезпеченні високої продуктивності та ефективності доступу до даних у різноманітних системах. Ефективне кешування визначає здатність систем оперативно обробляти великі обсяги даних, зменшуючи затримки та покращуючи загальну швидкість роботи. Проте, ефективність традиційних методів кешування може знижуватися в умовах зростання обсягу даних та збільшення вимог до продуктивності.

У цьому контексті розвиток та впровадження нових алгоритмів і методів кешування стає перспективним напрямком для підвищення продуктивності сучасних інформаційних систем. Сучасні підходи до кешування базуються на різних принципах управління даними, таких як частота використання, частого доступу та адаптивне керування ресурсами. У порівнянні з традиційними методами, нові алгоритми кешування дозволяють більш ефективно управляти пам'яттю, адаптуючись до змін у патернах доступу та забезпечуючи високу надійність систем.

Проведене дослідження дозволяє виділити нові напрямки розвитку алгоритмів та методів кешування, що включають оптимізацію параметрів систем кешування, впровадження гібридних підходів та розробку нових моделей аналізу продуктивності. Це сприяє підвищенню ефективності та стабільності систем в умовах динамічних змін та зростання обсягу даних.

Аналіз останніх досліджень і публікацій. Розглянемо основні типи кешування даних, такі як кешування в пам'яті, на диску та гібридні системи. Кешування в оперативно запам'ятовуючий пристрій (RAM) забезпечує швидкий доступ до даних завдяки низьким затримкам, але має обмежений обсяг пам'яті та вразливість до знеструмлення. Кешування на диску дозволяє зберігати більші обсяги даних, однак швидкість доступу значно нижча порівняно з RAM. Гібридні системи поєднують переваги обох підходів, що робить їх привабливими для застосування у високопродуктивних системах [1].

Останні дослідження в галузі кешування даних акцентують увагу на алгоритмах та їх модернізації, таких як Least Recently Used (LRU), Least Frequently Used (LFU), First In, First Out (FIFO), Adaptive Replacement Cache (ARC) та Most Recently Used (MRU). Розглянуто еластичні системи кешування на базі Memcached, що дозволяють динамічно адаптуватися до змін у навантаженні [2]. Ці дослідження сприяли розвитку нових архітектур та алгоритмів, що дозволяє динамічно масштабувати кеш відповідно до змін у навантаженні, забезпечуючи стабільну продуктивність та ефективне використання ресурсів.

Jelenković та Radovanović [3] дослідили поведінку алгоритму LRU при залежних запитах. Вони показали, що залежність між запитами може суттєво впливати на ефективність LRU, що важливо враховувати при його застосуванні в реальних системах, де запити часто не є незалежними. Запропонували спектр політик кешування, які об'єднують властивості LRU та Least Frequently Used (LFU). У досліджені обговорена можливість створення гібридних алгоритмів, які адаптивно перемикаються між LRU та LFU залежно від характеру навантаження, що дозволяє досягти кращої продуктивності в різних умовах [4].

Вперше застосували алгоритм ARC для реалізації гібридного кешування у великих системах, який перевершує традиційний LRU. ARC автоматично налаштовується під поточні патерни доступу до даних, комбінуючи переваги як недавнього використання, так і частоти доступу. Це дозволяє ефективніше використовувати кеш при змінних навантаженнях [5]. У дослідженні проведено аналіз сотень кластерів кешування на основі ключів та метаданих у Twitter, що дозволило краще зрозуміти особливості використання та оптимізації кешування в масштабних системах [6]. Дослідження надає інформацію про реальні патерни доступу та ефективність різних стратегій кешування у масштабних системах також було виявлено ключові фактори, що впливають на продуктивність, такі як розмір кешу, політики заміщення та характер робочих навантажень [6].

Гібридний алгоритм оптимізації кешу покращує традиційний LRU за допомогою генетичних алгоритмів та алгоритму мурашиної колонії. Цей метод дозволяє адаптивно налаштовувати стратегію кешування, враховуючи історію доступу та прогнозуючи майбутні запити, що підвищує ефективність управління кешем [7].

Адаптивна багаторівнева стратегія кешування для розподілених баз даних – підхід, який дозволяє ефективно управляти кешем на різних рівнях системи, зменшуючи затримки доступу та покращуючи загальну продуктивність. Для цього було запропоновано використовувати механізми прогнозування та адаптації для оптимального розподілу ресурсів кешу [8].

Спільне управління кешем LRU та MRU [9]. У цій статті дослідили теоретичний потенціал колаборативного кешування. З двочастинним кешем ці техніки можуть бути розширені для досягнення оптимальної продуктивності кешу.

Мета і задачі дослідження – аналіз сучасних методів кешування даних та їх застосування для підвищення продуктивності інформаційних систем. Основна увага приділяється ефективності та адаптивності різних алгоритмів кешування в різноманітних умовах. Для досягнення мети поставлено наступні завдання:

1. Проаналізувати сучасні алгоритми кешування даних та їх застосування в інформаційних системах.
2. Визначити переваги і недоліки застосування різних методів кешування у порівнянні з традиційними методами управління даними.
3. Розробити модель гібридного методу кешування, яка динамічно перемикається між алгоритмами LRU та MRU на основі аналізу патернів доступу.

Основна частина

Алгоритми кешування даних є одним із напрямків оптимізації управління пам'яттю в інформаційних системах [1]. У даному дослідженні розглянуто основні алгоритми кешування, такі як LRU, LFU, FIFO, ARC та MRU, і їх застосування в різних умовах. За основу кожен алгоритм має спільну архітектуру яка складається с ключа та значення, які мають часову або іншу мітку в метаданих, ознайомитися детальніше з якими можна у таблиці. Методи кешування базуються на різних підходах до управління даними. Алгоритм LRU видаляє нещодавно використані елементи, що дозволяє зберігати актуальні дані у кеші [7]. LFU краще підходить для систем зі стабільними частотами доступу до даних, де певні елементи постійно запитуються [4;7]. Однак він повільно реагує на зміни в патернах доступу, що може призвести до накопичення застарілих даних у кеші. FIFO забезпечує простоту реалізації, видаляючи найдавніше додані елементи [10]. Проте цей алгоритм не враховує ні частоту, ні недавність використання даних, що може знижувати ефективність кешування. ARC поєднує переваги LRU та LFU, автоматично адаптуючись до змін у патернах доступу, що забезпечує високу ефективність у різних умовах [5;11]. Складність реалізації та більші вимоги до ресурсів можуть бути недоліками цього алгоритму. MRU видаляє найбільш недавно використані елементи, що може бути корисним у специфічних випадках, наприклад, при обробці стекових структур даних [9]. Проте у більшості загальних випадків цей алгоритм може бути неефективним.

На основі представлених характеристик та аналізу алгоритмів кешування було визначено, що кожен з них оптимально підходить для певних типів робочих навантажень та патернів доступу до даних. LRU та LFU демонструють високу ефективність у системах з різними патернами доступу, тоді як FIFO є простим, але менш гнучким варіантом. ARC забезпечує адаптивність, яка дозволяє йому працювати ефективно в різних умовах, але його складність може обмежити застосування в деяких випадках. MRU залишається специфічним алгоритмом для певних сценаріїв, але не підходить для загального використання через свою обмежену ефективність. Зведені дані представлені у таблиці.

Порівняння характеристик алгоритмів

Алгоритм кешування	Методи використання	Складність алгоритму
LRU	Використовується в кешуванні, де ключі метаданих зберігають інформацію про час останнього доступу до кожного елемента.	O(1) для доступу і O(n) для оновлення у найгіршому випадку, O(1) при використанні хеш-таблиці з двозвязним списком

LFU	Використовується в кешуванні, де ключі метаданих зберігають інформацію про частоту доступу до кожного елемента.	O(1) для доступу і O(log_2n) для оновлення з використанням пріоритетної черги або хеш-таблиці
FIFO	Використовується в кешуванні, де ключі метаданих зберігають інформацію про час додавання кожного елемента до кешу.	O(1) для доступу і O(1) для оновлення
ARC	Використовується в кешуванні, де ключі метаданих зберігають інформацію про доступність елементів як за часом, так і за частотою.	O(1) для доступу і O(1) для оновлення у середньому
MRU	Використовується в кешуванні, де ключі метаданих зберігають інформацію про час останнього доступу до кожного елемента.	O(1) для доступу і O(1) для оновлення

Математичні моделі кешування дозволяють формалізувати процеси управління кешем та оцінити їх ефективність. Це важливо для точного налаштування систем кешування та забезпечення їх максимальної продуктивності. Нижче розглянуто кілька основних моделей.

Ймовірність Р-доступу використовується для визначення ймовірності того, що певний елемент буде запитано:

$$P(i) = \frac{n_i}{n_{total}}, \quad (1)$$

де $P(i)$ – ймовірність того, що конкретний елемент i буде запитано користувачем або системою, n_i – кількість звернень до елемента i, n_{total} – загальна кількість доступів.

Модель черги W – середнього часу очікування обробки запитів в системах кешування:

$$W = \frac{\lambda}{\mu(\mu-\lambda)}, \quad (2)$$

де λ – інтенсивність запитів, μ – інтенсивність обслуговування.

Оцінка пропускної здатності системи визначення максимального числа запитів:

$$T = \frac{N}{T_{total}}, \quad (3)$$

де N – кількість оброблених запитів, T_{total} – загальний час обробки.

Ефективність кешу часто вимірюється за допомогою коефіцієнта, який визначає частку запитів, що обслуговуються кешем:

$$H = \frac{n}{n_r}, \quad (4)$$

де H – коефіцієнт кешування, n – кількість звернень до кешу, n_r – загальна кількість запитів.

Розмір вартості та ефективності кешування збільшується пропорційно до логарифма розміру кешу, можна використати модель:

$$E(s) = a \cdot log_2(s) + b, \quad (5)$$

де E – функція ефективності, s – розмір кешу, a і b – константи.

Визначення часу відповіді системи. Для оцінки середнього часу відповіді системи можна використовувати моделі черг:

$$R = W + \frac{1}{\mu}, \quad (6)$$

де R – середній час відповіді.

Математичні моделі і методи дозволяють точно оцінювати продуктивність систем кешування, визначати оптимальні параметри і прогнозувати навантаження, що значно підвищує ефективність інформаційних систем.

Ці моделі дозволяють точно оцінювати ефективність різних алгоритмів та оптимізувати налаштування систем кешування для досягнення максимальної продуктивності [1].

Запропонована гібридна модель, яка поєднує стратегії LRU та MRU за допомогою адаптивного механізму вибору стратегії кешування залежно від історії доступів та частоти використання даних.

$$C = L_{lru} + L_{mru},$$

де C – загальний обсяг кешу, L_{lru} – частка кешу, виділена під алгоритм LRU, L_{mru} – частка кешу, виділена під алгоритм MRU.

Видалення даних з кешу виконується за принципом вибору між LRU і MRU залежно від значення функцій частоти доступу.

$$P(t) = a \times (\frac{f_{lry}(t)}{L_{lru}}) + b \times (\frac{f_{mry}(t)}{L_{mru}}),$$

де $f_{lry}(t)$ – частота доступу до даних у кеші LRU за час t , $f_{mry}(t)$ – частота доступу до даних у кеші LRU за час t, a та b – вагові коефіцієнти.

Алгоритм вибору стратегії видалення даних:

$$f(x) = \begin{cases} LRU, \ f_{lry}(t) > f_{mry}(t) \\ MRU, \ f_{mry}(t) \leq f_{lry}(t \ ' \end{cases}$$

де S – частота доступу в алгоритмі.

Представлений гібридний метод кешування поєднує алгоритми LRU (Least Recently Used) і MRU (Most Recently Used) з інтеграцією механізму архівування для підвищення ефективності зберігання та доступу до даних. Архітектура гібридного кешування (рис. 1) забезпечує адаптивне перемикання між алгоритмами LRU та MRU залежно від патернів доступу, що дозволяє динамічно підлаштовуватися до змінних умов роботи системи.

Процес обробки запиту розпочинається з надходження запиту від користувача, який обробляється модулем кешування. Якщо необхідні дані вже знаходяться в кеші, вони повертаються користувачу без затримки. У випадку відсутності даних у кеші, запит перенаправляється до архіву, де вони зберігаються для подальшого доступу. Ключовим компонентом архітектури є адаптивний блок перемикання, який обирає відповідний алгоритм кешування – LRU або MRU – на основі ана-лізу поточних запитів. Це забезпечує гнучке управління кешем, що дозволяє оптимально використовувати ресурси системи.

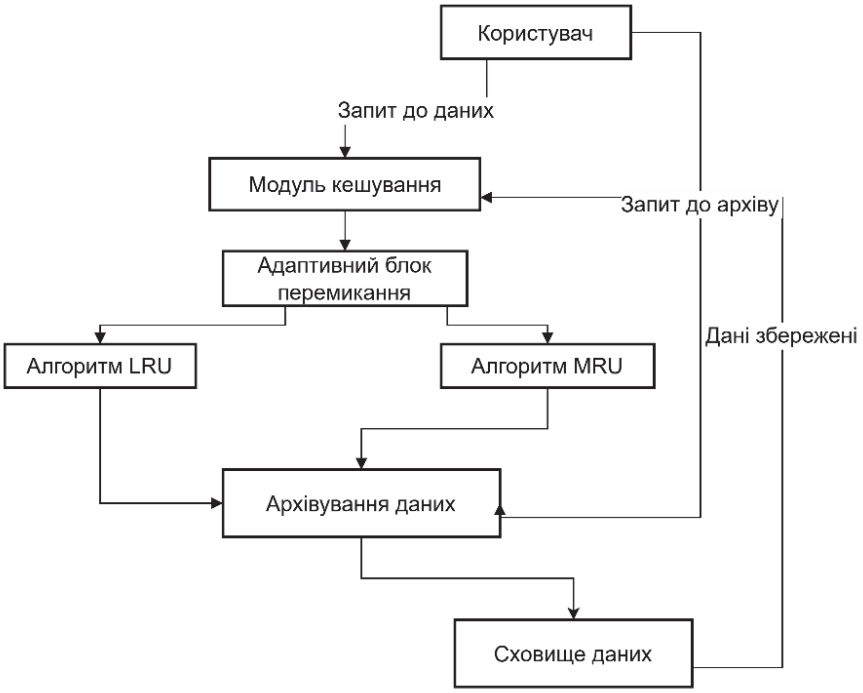

Рис. 1. Архітектура гібридного кешування

Адаптивність у виборі стратегії кешування

Адаптивність використовується для забезпечення оптимальної продуктивності системи в умовах змінних патернів доступу. Алгоритми кешування можуть адаптуватися до нових умов, змінюючи свої стратегії на основі поточних характеристик навантаження.

Частота доступу до даних в кеші є метрикою, що дозволяє зрозуміти, які дані користувачі запитують найчастіше. Для ефективної стратегії адаптування треба дізнатися, частоту доступу до даних та які дані мають більшу ймовірність бути викликаними знову. Визначимо частоту доступу до даних в кеші:

$$f_{lry}(t) = \frac{R_{lru}(t)}{T},$$
$$f_{mry}(t) = \frac{R_{mru}(t)}{T},$$

де $R_{lru}(t)$ – запити до кешу, оброблених алгоритмом LRU до моменту t, $R_{mru}(t)$ – запити до кешу, оброблених алгоритмом MRU до моменту t. T — загальний час спостереження.

Визначення загальної частоти можна обчислити для всіх об'єктів, а також загальну частоту доступу до кешу:

$$f_{cache}(t) = \frac{\sum_{i=1}^{N} N(t)}{T},$$

де $f_{cache}(t)$ – загальна частота доступу до кешу в момент часу t, N – загальна кількість кешованих об'єктів.

Оцінювання продуктивності кешування здійснюється на основі параметрів, які враховують затримки та пропускну здатність системи. Оцінка загальної затримка при обробці запитів можливо розраховувати у вигляді:

$$Delay = D_{lru} + D_{mru},$$

де $Delay$ – загальна затримка, D_{lru} – затримка для запитів, оброблених алгоритмом LRU, D_{mru}– затримка для запитів, оброблених алгоритмом MRU.

Оцінка ефективності кешування. Ефективність кешування можна виміряти за допомогою коефіцієнта, який визначає частку запитів, що обслуговуються кешем, вимірявши для кожного алгоритму свій коефіцієнт:

$$H = \frac{H_{ltu} + H_{mtu}}{n_r},$$

де H_{ltu} – кількість успішних запитів у кеші для LRU, H_{mtu} – кількість успішних запитів у кеші для MRU.

На основі отриманих даних системи адаптуємо розміри часток кешу для LRU та MRU. Встановлюємо нові значення:

$$L_{lru} = L_{lru} + \gamma \times (H - H_{threshold}),$$
$$L_{mru} = L_{mru} + \gamma \times (H_{threshold} - H),$$

де L_{lru} – частка кешу, виділена під LRU, L_{mru} – частка кешу, виділена під MRU, γ – швидкість адаптації, $H_{threshold}$ – порогове значення для коефіцієнта потрапляння.

Алгоритм адаптивного кешування

Для покращення розуміння стабільності патернів доступу до даних пропонується обчислювати дисперсію частоти доступу. Дисперсія вимірює варіацію частоти запитів до даних і дозволяє оцінити стабільність та змінність патернів доступу. Чим вища дисперсія, тим менша стабільність алгоритму, що допомагає визначити, наскільки оптимально використовувати той чи інший алгоритм кешування.

Дисперсія для алгоритмів LRU та MRU обчислюється за формулами:

$$\sigma^2_{LRU} = \frac{1}{T}\sum_{i=1}^{n}(f_{lry}(t_i) - \bar{f})^2 \Delta t_i,$$
$$\sigma^2_{MRU} = \frac{1}{T}\sum_{i=1}^{n}(f_{mry}(t_i) - \bar{f})^2 \Delta t_i,$$
$$\bar{f} = \frac{\sum_{i=1}^{n}(f(i) \times \Delta t_i)}{\sum_{i=1}^{n} \Delta t_i},$$

$$T = \sum_{i=1}^{n} \Delta t_i ,$$

де σ^2_{LRU} – дисперсія частоти доступу для алгоритму LRU; σ^2_{MRU} – дисперсія частоти доступу для алгоритму MRU; T – загальний час; $\bar{f}$ – середнє значення частоти доступу для кожного алгоритму за період часу T; $f_{lry}(t_i)$ та $f_{mry}(t_i)$ – частота доступа до певного ключа у кеші, на момент часу t_i; n - кількість окремих моментів часу t_i , на яких здійснюється вимірювання показників.

З огляду на те, що простий підрахунок частоти доступу до даних не завжди відображає реальну потребу у видаленні даних, пропонується використовувати дисперсію для прийняття рішень. Висока дисперсія свідчить про нестабільність доступу до даних, що може вказувати на те, що нещодавно використані дані можуть більше не знадобитися. У такому випадку адаптивна модель кешування обирає алгоритм, який найкраще відповідає поточному патерну запитів.

Критерій вибору алгоритму кешування можна записати наступним чином:

$$f(x) = \begin{cases} LRU, \sigma^2_{LRU} < \sigma^2_{MRU} \\ MRU, \sigma^2_{MRU} \le \sigma^2_{LRU} \end{cases}.$$

Для динамічної адаптації змін у патернах доступу використовується метод експоненціального згладжування для обчислення середньої частоти доступу:

$$F_i(t) = a \times X_i(t) + (1 - a) \times F_i(t-1),$$

де $F_i(t)$ – згладжена частота доступу до елемента і в момент часу t; $X_i(t)$ – фактична кількість звернень до елемента i за останній період часу; a – коефіцієнт згладжування.

При отриманні нового запиту система перевіряє наявність даних у кеші. Якщо дані є, вони негайно повертаються користувачу. Тоді як даних немає, проводиться аналіз патернів доступу шляхом обчислення дисперсій частоти доступу для обох алгоритмів. Блок-схема алгоритму адаптивного кешування (рис. 2), яка ілюструє процес прийняття рішень при обробці запитів до даних із можливістю перемикання між алгоритмами кешування LRU (Least Recently Used) і MRU (Most Recently Used) на основі аналізу патернів доступу.

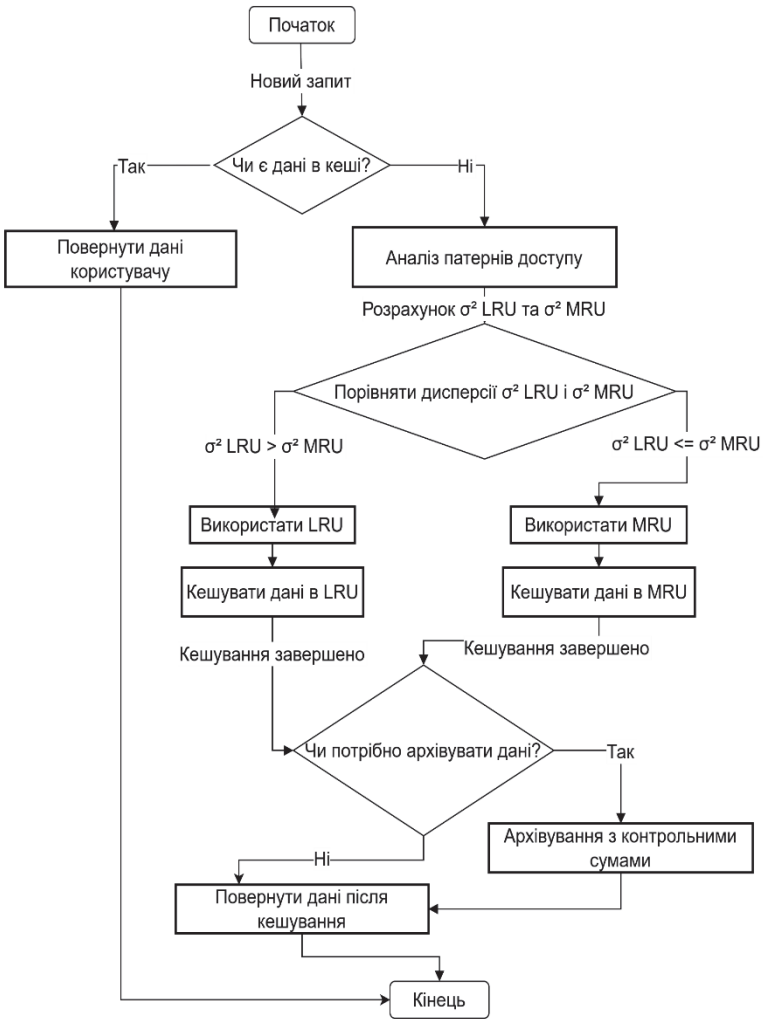

Рис. 2. Алгоритм адаптивного кешування

Додатково система перевіряє необхідність архівування даних. Якщо архівування потрібне, дані зберігаються з обчисленням контрольної суми для забезпечення цілісності. Якщо політика кешування не передбачає архівування, то дані залишаються в кеші для подальшого використання.

Такий адаптивний підхід дозволяє системі ефективно реагувати на зміни в патернах доступу, оптимізуючи використання кешу та підвищуючи загальну продуктивність. Він забезпечує гнучкість управління кешуванням, поєднуючи переваги алгоритмів LRU та MRU відповідно до поточних потреб системи.

Висновки

Вибір алгоритму кешування, що відповідає конкретній задачі, має вирішальне значення для досягнення оптимальних результатів. Дослідження алгоритмів та методів кешування даних, які застосовуються для зменшення затримок на відповідь даних зі сховищ, показало, що кожен з розглянутих алгоритмів має свої унікальні особливості та переваги, що робить їх ефективними у відповідних сценаріях. Гібридні алгоритми кешування можуть ефективно комбінувати переваги LRU та MRU, забезпечуючи гнучкість та адаптивність управління кешем у динамічних умовах роботи системи. Це дозволяє оптимізувати використання ресурсів кешу, знижувати затримки доступу та підвищувати загальну продуктивність інформаційних систем.

Запропонований гібридний метод кешування, який поєднує алгоритми LRU та MRU, може продемонструвати підвищення в продуктивності та ефективності кешування. Використання адаптивної стратегії, яка динамічно підлаштовується під дані, дозволить значно зменшити кількість кеш-місів і підвищити загальну пропускну здатність системи. Дослідження підтвердило, що гібридні метод кешування є обіцяючими для підвищення ефективності систем.

Список літератури

1. Киричек Г.Г., Тягунова М.Ю., Братчиков В.В. Система кешування даних в розгалуженій мікросервісній архітектурі. Вчені записки Таврійського національного університету ім. В.І. Вернадського. Серія: Технічні науки. 2024. 141–146.

2. Hafeez U. U., Wajahat M., Gandhi A. ElMem: Towards an Elastic Memcached System. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, 2–6 July 2018. 2018.

3. Jelenković P. R., Radovanović A. Least-recently-used caching with dependent requests. Theoretical Computer Science. 2004. Vol. 326, no. 1-3. P. 293–327.

4. On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies / D. Lee et al. ACM SIGMETRICS Performance Evaluation Review. 1999. Vol. 27, no. 1. P. 134–143.

5. Megiddo N., Modha D. S. Outperforming LRU with an adaptive replacement cache algorithm. Computer. 2004. Vol. 37, no. 4. P. 58–65.

6. Yang J., Yue Y., Rashmi K. V. A Large-scale Analysis of Hundreds of In-memory Key-value Cache Clusters at Twitter. ACM Transactions on Storage. 2021. Vol. 17, no. 3. P. 1–35.

7. LRU-GENACO: A Hybrid Cached Data Optimization Based on the Least Used Method Improved Using Ant Colony and Genetic Algorithms / M. I. Zulfa ma ін. Electronics. 2022. Т. 11, № 19. С. 2978. URL: https://doi.org/10.3390/electronics11192978.

8. An adaptive multi-level caching strategy for Distributed Database System / F. Lu ma ін. Future Generation Computer Systems. 2019. Т. 97. С. 61–68.

9. Gu X., Ding C. On the theory and potential of LRU-MRU collaborative cache management. ACM SIGPLAN Notices. 2011. Т. 46, № 11. С. 43–54.

10. Operating Systems: Internals and Design Principles / G. Paul et al. Pearson Education, Limited, 2013. 788 p.

11. Nimrod M., S M. D. ARC: A Self-Tuning, Low Overhead Replacement Cache. 2nd USENIX Conference on File and Storage Technologies, San Francisco, CA, 31 March 2003. P. 115–130.

O. Zolotukhina, B. Khudik, A. Havor

OPTIMIZE CACHING PERFORMANCE WITH A HYBRID DATA CACHING METHOD

Data caching plays a key role in improving performance and speed of access to frequently requested resources. The use of different types and methods of caching is crucial for optimal system performance and reliability.

Data caching is used to store frequently used resources in RAM, on disk, or in hybrid systems that combine both approaches. The use of RAM provides high speed data access, while disk caching allows you to store larger amounts of data. Hybrid systems combine the advantages of both methods, achieving a balance between speed and storage capacity.

Particular attention is paid to caching algorithms that provide efficient data management in the cache. Popular algorithms such as Least Recently Used (LRU), Least Frequently Used (LFU), First In, First Out (FIFO), Adaptive Replacement Cache (ARC), and Most Recently Used (MRU) are considered. These algorithms are analyzed in the context of their application to optimize the performance of caching systems.

The advantages and disadvantages of using these algorithms in different scenarios are investigated. Attention is paid to solving the problems of optimizing the cache size, reducing delays in accessing data, and increasing the efficiency of resource use. Mathematical models and methods for analyzing caching performance are proposed to evaluate the effectiveness of various algorithms and optimize the settings of caching systems to achieve maximum performance.

A hybrid caching method is proposed and implemented that combines the LRU and MRU algorithms by dynamically switching between them based on the analysis of the variance of data access frequency. This approach involves calculating the statistical characteristics of data access, which allows the system to adaptively select the most appropriate caching algorithm in real time. Using this method has increased caching performance and efficiency, reduced the number of cache misses, and improved overall system throughput.

Keywords: data caching, performance, information systems, caching algorithms, hybrid systems, distributed caching, local caching, optimization, performance analysis.