Специфікація програмного забезпечення

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«Алгоритми управління кешованими даними в Redis для підвищення продуктивності застосунків»

Автор:

ст. гр. ІПЗм-23-1

Кучапін Матвій Юрійович

Харків, 2025

**ВСТУП**

* 1. Огляд програмного забезпечення

У сучасних інформаційних системах обсяг даних та запитів до кешу постійно зростає. Redis, як високопродуктивне in-memory сховище, широко використовується для кешування даних та зниження затримок обробки запитів [1]. Однак класичні підходи до керування кешем з використанням фіксованого TTL, LRU чи LFU мають суттєві обмеження при роботі зі складними структурами даних (хеші, списки, множини). Внаслідок цього часто відбувається надмірне видалення актуальних даних або зберігання застарілих елементів, що призводить до додаткового навантаження на основну базу даних та зниження продуктивності системи.

Для вирішення цих проблем необхідне програмне забезпечення, яке забезпечить:

* адаптивне керування часом життя даних із урахуванням їхньої важливості та частоти звернень;
* гнучкий механізм евікції, здатний балансувати між віком даних, їх частотою доступу й обсягом пам’яті;
* можливість інтеграції з існуючими .NET-застосунками, що використовують StackExchange.Redis [2].

Розроблене рішення повинно дозволити знизити кількість непотрібних кеш-промахів, оптимізувати використання пам’яті та підтримувати стабільну високу продуктивність навіть при змінних навантаженнях.

* 1. Мета

Метою даної роботи є проведення дослідження існуючих методів управління кешованими даними та розробка модифікацій і впровадження покращень для їх адаптації в Redis для підвищення ефективності використання пам’яті та продуктивності систем при роботі зі складними структурами даних. Для досягнення цієї мети необхідно:

* провести аналіз та оцінку обмежень традиційних методів TTL, LRU та LFU;
* розробити механізм динамічного визначення TTL на рівні окремих елементів складних структур із урахуванням частоти доступу та давності звернень;
* створити адаптивний алгоритм евікції, що поєднує метрики віку даних, їх частоти використання та розміру об’єктів;
* реалізувати готовий до інтеграції модуль для платформи .NET із використанням StackExchange.Redis;
* провести експериментальну оцінку ефективності розроблених підходів за ключовими метриками: Cache Hit Rate, Cache Miss Rate, середній TTL, кількість евікшенів, затримки операцій читання/запису, використання пам’яті та навантаження на CPU.

Дотримання поставленої мети забезпечить створення рішення, яке не тільки адаптуватиметься до різних характерів навантаження, а й надасть практичні рекомендації щодо інтеграції та налаштування адаптивного кеш-менеджера у реальних проєктних середовищах.

* 1. Межі

У рамках цього дослідження охоплюються наступні аспекти:

* управління TTL лише для кешованих об’єктів у Redis;
* механізми евікції в межах одного екземпляру Redis та без розгляду розподілених або багатоквартетних конфігурацій;
* інтеграція з .NET через клієнтську бібліотеку StackExchange.Redis;
* аналіз продуктивності та використання ресурсів для одиночного вузла Redis.

При цьому дослідження не охоплює:

* налаштування та оптимізацію конфігурації шардування чи реплікації Redis;
* питання безпеки, резервного копіювання та відновлення даних у Redis;
* реалізацію клієнтів для інших мов програмування;
* дослідження мережевих аспектів розгортання Redis у клауд-середовищах.

Виділені межі забезпечують чіткість та фокус дослідження на розробці та оцінці адаптивних алгоритмів кешування для одиночного вузла Redis.

* 1. Посилання

Документ Специфікація програмного забезпечення посилається на стандарт ДСТУ ISO / IEC 15910-2002 «Процес створення документації користувача для програмного застосунку».

У тексті використані такі посилання:

1. Redis Documentation. URL: https://redis.io/docs/ (дата звернення 20.05.2025).
2. StackExchange.Redis Documentation. URL: https://stackexchange.github.io/StackExchange.Redis/ (дата звернення 20.05.2025).

# **2 ЗАГАЛЬНИЙ ОПИС**

2.1. Перспективи програмного забезпечення

Сучасне програмне забезпечення дедалі більше покладається на in-memory кеші для забезпечення високої продуктивності та низьких затримок при обробці даних. Розробка адаптивного кеш-менеджера для Redis відкриває широкі можливості підвищити гнучкість систем, дозволяючи автоматично налаштовувати поведінку кешу під специфічні шаблони доступу та змінне навантаження без потреби в ручних коригуваннях. За рахунок оптимізованого керування TTL та адаптивної евікції зменшується надмірне споживання пам’яті та навантаження на основну базу даних, що допомагає економити обчислювальні ресурси. Легка інтеграція рішення, завдяки модульній архітектурі та сумісності зі StackExchange.Redis, дає змогу швидко підключати кеш-менеджер до існуючих .NET-застосунків, а подальше розширення функціональних можливостей, наприклад, підтримка кластерних конфігурацій Redis або клієнтів інших мов може забезпечить застосовність у різних середовищах. Накопичені метрики й статистика доступів також можуть бути використані для побудови аналітичних дашбордів та прогнозування майбутніх навантажень, що надає додаткову цінність впровадженню такого рішення.

Таким чином, адаптивний кеш-менеджер створює потужну платформу для розвитку інтелектуальних механізмів кешування у сучасних інформаційних системах.

2.2. Функції програмного забезпечення

На даному етапі проекту передбачається, що розроблене програмне забезпечення забезпечуватиме наступні ключові функції:

* сервіс оброблятиме операції читання та запису кешованих об’єктів без блокування потоків, дозволяючи масштабувати обробку численних одночасних запитів і знижувати затримки в системі;
* компонент моніторингу збирає дані про час останнього доступу, частоту операцій та використовуваний обсяг пам’яті для кожного ключа, зберігаючи історію звернень для подальшого аналізу та звітності;
* механізм динамічно перераховуватиме час життя елементів кешу, зменшуючи вплив старих звернень та підтримуючи актуальність даних у кеші;
* алгоритм розраховуватиме спеціальну метрику видалення з урахуванням віку, популярності та розміру об’єкта і автоматично видалятиме менш пріоритетні записи;
* фонова служба очищення прострочених елементів періодично скануватиме кеш із метою видалення елементів із завершеним TTL та очищення пов’язаних метаданих без втручання користувача.
* налаштування вагових коефіцієнтів та порогів алгоритмів.

Ці функції разом забезпечать автоматизоване та інтелектуальне управління кешем у Redis, зменшуючи ручні налаштування та покращуючи загальну продуктивність системи.

2.3. Характеристики користувачів

Характеристики зацікавлених користувачів наведені у таблиці 2.1.

Таблиця 2.1 – Характеристика зацікавлених користувачів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Зацікавлена сторона** | **Основна цінність** | **Обов’язки** | **Головна ціль** | **Обмеження** |
| 1 | 2 | 3 | 4 | 5 |
| **Розробники .NET-додатків** | Простота інтеграції, зворотна сумісність із StackExchange.Redis | Інтегрувати адаптивний кеш-клієнт у свої сервіси; налаштовувати й оновлювати параметри | Знизити кількість кеш-промахів, підвищити швидкодію | Обмежений час на вивчення нового API; необхідність мінімальних змін у коді |

Кінець таблиці 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| **DevOps / Системні адміністратори** | Моніторинг та стабільність роботи інстансу Redis | Розгортати, масштабувати й контролювати інстанси Redis; налаштувати параметри кеш-менеджера | Підтримувати високу доступність і передбачуваність | Обмежені права доступу; політики безпеки та відмовостійкості |
| **Архітектори систем** | Гнучкість конфігурації, масштабованість | Визначати стратегію кешування на рівні архітектури; вибирати оптимальні налаштування TTL і евікції | Забезпечити баланс між складністю рішення та функціональністю | Необхідність узгодження з іншими компонентами архітектури |
| **Тестувальники (QA)** | Репродуктивність сценаріїв, наявність метрик для перевірки | Створювати тестові кейси для навантаження й функціональності кешу; перевіряти коректність роботи сервісу | Виявити потенційні дефекти кеш-логіки | Обмежений доступ до продакшн-даних; час на автоматизацію тестів |
| **Менеджмент / Бізнес-власники** | Зниження операційних витрат, підвищення користувацького досвіду | Бюджетування проєкту; прийняття рішення про впровадження; оцінка результатів експериментів | Максимізувати продуктивність системи при мінімальних витратах | Обмежений бюджет і жорсткі терміни запуску |

2.4. Загальні обмеження

У межах даного проєкту передбачено низку загальних обмежень. Підтримуються лише інстанси Redis версії 6.x та новіші без гарантії зворотної сумісності з попередніми релізами. Рішення розраховане на платформу .NET 8 та вище, тоді як робота на старіших фреймворках не забезпечується. Архітектура проекту зосереджена на одиночному вузлі Redis. Кластерна конфігурація, реплікація або шардування в межах цього рішення не розглядаються. Прототип оптимізовано для навантаження до 50 000 запитів на секунду за стандартних апаратних ресурсів, а для більших обсягів потрібні додаткові заходи з оптимізації. Відповідальність за забезпечення безпеки, включаючи налаштування TLS та ACL, покладається на зовнішні інфраструктурні компоненти, оскільки вбудовані механізми шифрування та автентифікації в межах даного проєкту не реалізуються. Крім того, клієнтська сумісність обмежується бібліотекою StackExchange.Redis, інші драйвери та клієнти Redis не підтримуються.

Ці обмеження спрямовані на забезпечення стабільного середовища для розробки та тестування адаптивних алгоритмів кешування у Redis.

2.5. Припущення і залежності

У процесі розробки та впровадження рішення робляться такі ключові припущення:

* інфраструктура забезпечує стабільне та надійне мережеве з’єднання між .NET-додатком та сервером Redis без затримок, що б ускладнили аналіз метрик та адаптивне встановлення TTL;
* основний екземпляр Redis працює в режимі «single instance» та не піддається непередбаченим перезапускам або аварійним відмовам протягом періоду коректної роботи алгоритмів;
* апаратура має достатні ресурси CPU та пам’яті для обробки додаткових обчислень, пов’язаних із розрахунком метрик та метаданих кешу;
* усі C#-клієнти використовують єдину версію бібліотеки StackExchange.Redis без неконкурентних модифікацій.

Залежності проєкту включають:

* сервер Redis (версії 6.x та вище) із налаштованим обмеженням maxmemory для активації алгоритмів евікції;
* .NET 8+ SDK для збірки та запуску клієнтської бібліотеки;
* конфігураційний сервіс для передачі вагових коефіцієнтів і порогових значень алгоритмів;
* система логування для збирання та збереження звітних даних.

# **3 КОНКРЕТНІ ВИМОГИ**

3.1. Вимоги до зовнішніх інтерфейсів

3.1.1. Апаратний інтерфейс

Для коректної роботи адаптивного розширення кешу та евікшен-менеджера необхідна серверна платформа з такими мінімальними характеристиками:

* процесор: x64-сумісний CPU з не менше ніж двома ядрами (2.0 ГГц або вище);
* оперативна пам’ять, мінімум 4 ГБ, рекомендовано 8 ГБ для .NET-додатків із середнім навантаженням;
* пропускна здатність мережевого інтерфейсу не менше 10 МБ/с та затримка між додатком та Redis-сервером < 10 мс.
* відкритий TCP-порт, за замовчанням 6379 або налаштований для TLS/SSL, для з’єднання зі службою Redis;
* SSD-накопичувач.

Ці вимоги гарантують стабільну роботу бібліотеки під навантаженням та своєчасне виконання фонового очищення кешу.

3.1.2. Програмний інтерфейс

Програмний інтерфейс реалізується у вигляді .NET-бібліотеки, призначеної для інтеграції у програмні рішення, що використовують систему кешування Redis, та забезпечує розширений функціонал управлення кешованими даними шляхом використання адаптивного управління TTL та гібридного алгоритму заміщення, який поєднує LRU та LFU стратегії для ефективнішого використання пам'яті. Бібліотека розробляється з використанням технології .NET 8, бібліотеки StackExchange.Redis для підключення до Redis, а також інтегрованого середовища розробки Visual Studio 2022, що забезпечує комфортну розробку, тестування та відлагодження коду. Для написання юніт-тестів використовується фреймворк xUnit, для зручної автоматичної серіалізації даних використовується бібліотека System.Text.Json, а для моніторингу стану бібліотеки та логування – стандартний механізм ILogger платформи .NET. Доступ до бібліотеки надається через програмний код та не передбачає наявність окремого конфігураційного файлу або зовнішнього інтерфейсу користувача. Всі налаштування роботи бібліотеки, такі як параметри адаптивного TTL або коефіцієнти гібридного алгоритму, здійснюються безпосередньо у програмному коді інтегрованого рішення, що забезпечує максимальну гнучкість і контроль. Для використання бібліотеки необхідно мати доступ до Redis-сервера, який може бути розгорнутий локально, на віддаленому сервері, або за допомогою хмарних сервісів. Використовується протокол TCP/IP для обміну даними між бібліотекою та Redis.

3.1.3. Комунікаційний протокол

Комунікація з Redis-сервером відбувається через Redis Serialization Protocol (RESP) поверх TCP/IP на налаштований порт за замовчуванням 6379, за необхідності з підтримкою TLS-шифрування, причому для обміну даними використовується набір стандартних Redis-команд (HSET, HGET, ZADD тощо), тоді як внутрішня взаємодія між компонентами бібліотеки реалізується виключно через асинхронні виклики .NET без залучення додаткових HTTP-або REST-протоколів.

3.1.4. Обмеження пам’яті

Потрібно мати мінімум 4ГБ оперативної пам’яті.

3.1.5. Операції

Система виконує такі ключові операції: асинхронне збереження складних об’єктів у кеш із динамічною установкою TTL для окремих полів, читання даних із кешу з автоматичним оновленням метрик доступу, видалення конкретних ключів за запитом, періодичний прохід фонового процесу для очищення прострочених елементів, розрахунок пріоритетів евікшену на основі вагових коефіцієнтів давності, частоти та обсягу пам’яті, примусове видалення найменш пріоритетних записів при досягненні порогу використання пам’яті.

3.2. Властивості програмного забезпечення

Програмна забезпечення розроблено за використанням актуальних технологій. Розроблена за допомогою мови програмування C# та технології .NET.

3.3. Атрибути програмного забезпечення

Програмне забезпечення, що розробляється, характеризується такими атрибутами:

* надійність. Гарантується стійкість до відмов шляхом автоматичного контролю використання пам’яті, мінімізації ризику втрати даних за допомогою адаптивних алгоритмів витіснення та динамічного TTL, а також забезпечується регулярне логування та моніторинг ключових метрик;
* ефективність. Оптимізоване використання ресурсів пам’яті та процесора завдяки застосуванню адаптивних методик управління життєвим циклом кешованих об’єктів, що дозволяє зменшити кількість евікшенів та покращити час відповіді при частих зверненнях до даних;
* супроводжуваність. Забезпечується шляхом використання сучасних підходів до розробки програмного забезпечення, зокрема чіткої структури коду, модульності, широкого застосування автоматизованих юніт-тестів, а також використання загальноприйнятих стандартів документування та ведення коду в системі контролю версій;
* гнучкість. Програмне рішення надає можливість налаштування вагових коефіцієнтів алгоритмів евікшену та TTL безпосередньо в коді, що дозволяє адаптувати систему до конкретних бізнес-вимог та навантажень;
* масштабованість. Рішення здатне підтримувати ефективну роботу за умов зростання навантаження завдяки використанню асинхронних операцій та підтримці роботи в багатопотоковому середовищі;
* безпека. Передбачено використання надійних комунікаційних протоколів (TCP/IP, HTTPS) та механізмів шифрування даних для запобігання несанкціонованому доступу до чутливої інформації, яка зберігається у кеші.