ІНФОРМАТИКА, ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА АВТОМАТИЗАЦІЯ

УДК 004.043
DOI https://doi.org/10.32838/2663-5941/2022.1/09

Абгарян Ю.С.
SoftServe

REMOTE DICTIONARY SERVER ЯК ОДНА ІЗ ГОЛОВНИХ СТРУКТУР МЕРЕЖЕВОГО СХОВИЩА ДАНИХ

У статті проведено дослідження Remote Dictionary Server як однієї з головних структур мережевого сховища даних. Зазначено, що Redis – це швидке сховище даних типу «ключ-значення» у пам'яті з відкритим вихідним кодом. Наголошено на тому, що Redis забезпечує час відгуку на рівні часток мілісекунди та дозволяє програмам, які працюють у режимі реального часу, виконувати мільйони запитів на секунду. Такі програми потрібні у сферах ігор, рекламних технологій, фінансових сервісів, охорони здоров'я та IoT. Сьогодні Redis – один із найпопулярніших ядер із відкритим вихідним кодом, який протягом п'яти років поспіль називається «найулюбленішою» базою даних від Stack Overflow. Всі дані Redis зберігаються у пам'яті, що забезпечує низьку затримку та високу пропускну здатність доступу до даних. На відміну від традиційних баз даних, сховища даних пам'яті не вимагають переміщення на диск, а це скорочує затримку ядра до мікросекунд. Завдяки цьому сховища даних у пам'яті можуть багаторазово збільшувати кількість операцій, які виконуються, і скорочувати час відгуку. Як результат, забезпечується надзвичайно висока продуктивність. Операції читання та запису у середньому займають менше мілісекунди, швидкість роботи досягає мільйонів операцій на секунду. У роботі наголошується на перевагах використання Remote Dictionary Server, до яких варто віднести: гнучкі структури даних (на відміну від інших сховищ на основі пар «ключ – значення», які підтримують обмежений набір структур даних, Redis підтримує величезну різноманітність структур даних, що дозволяє задовольнити потреби різноманітних програм); простоту та зручність (Redis дозволяє писати найскладніший код із меншою кількістю простих рядків); реплікацію та постійне зберігання (у Redis застосовується архітектура вузлів «провідний підлеглий» і підтримується асинхронна реплікація, за якої дані можуть копіюватися на кілька підлеглих серверів); висока доступність і масштабованість (Redis пропонує архітектуру «провідний підлеглий» із одним провідним вузлом або із кластерною топологією); всі інструменти з відкритим вихідним кодом.

Ключові слова: *сховище даних, мережа, сервер, база даних, Remote Dictionary Server, структура, інформація.*

Вступ і постановка завдання. Останнім часом обробка даних у пам'яті стає дедалі більш цінною, оскільки важливо досліджувати величезну кількість інформації за короткий час. Робота з великим обсягом даних завжди є проблемою розробки та щоденного обслуговування. Якщо для зберігання такої величезної кількості даних використовується SQL, неможливо обробити непередбачувану та неструктуровану інформацію, але мережа потребує бази даних, орієнтованої на випадки, яка буде дуже гнучкою та працюватиме на схемі з меншою моделлю даних, тому SQL не може бути пристосований для цієї роботи. NoSQL має здатність підтримувати великий обсяг операцій читання-запису та зберігати загальні об'єкти, такі як JSON, сприяє узгодженості даних у розподіленій системі. Це показує, що база даних NoSQL є чудовим вибором для обробки великих і неструктурованих даних. Враховуючи кількість даних, схему доступу до даних і характеристики сучасних мереж, NoSQL має свої неодмінні переваги. Насправді NoSQL вже широко використовується у сучасних веб-додатках і корпоративних рішеннях для розробки програмного забезпечення. Останнім часом обробка даних у пам'яті стає дедалі більш цінною, оскільки важливо досліджувати величезну кількість інформації за короткий проміжок часу. NoSQL має багато переваг, включаючи

простоту дизайну, масштабованість моделі даних, контроль паралельності, узгодженість у сховищі тощо. REmote DIctionary Server (Redis) – це сховище структури даних у пам'яті, віддалена база даних, яка пропонує високу продуктивність, володіє гнучким і відкритим вихідним кодом.

Аналіз останніх досліджень і публікацій. Формуючи уявлення про REmote DIctionary Server, варто відзначити наукові праці таких сучасних вчених, як: Г.Г. Жолткевич [1], котра у своїй дисертаційній роботі вирішила актуальну науково-практичну задачу розробки загального методу прийняття рішень, на який можливо спиратися у побудові мережі для розподіленого сховища даних (РСД) з метою запобігання певних ризиків некоректної поведінки систем при розподілених обчисленнях, обробці запитів у розподіленому середовищі. Це дозволяє уникати у процесі проєктування постійного врахування ліміту запитів, доступної кількості кластерів і їх розташування, а натомість мати конкретні критерії для формування сховища за умов різноманітних потреб конкретних розподілених систем із урахуванням обмежень, встановлених CAP-теоремою.

М. Колонко [2] створив методи автоматизації розробки логічних моделей баз даних на основі концептуальних субмоделей із багатоваріантною персистентністю. Автором удосконалено мову концептуального моделювання AGILA MOD+ за рахунок за рахунок удосконаленої структури концептуальної моделі даних і методів створення моделей даних на концептуальному рівні, що дозволило враховувати багатоваріантну персистентність і скоротити час проектування баз даних інформаційних систем.

У роботі [3] А.В. Ісаков, В.Ю. Ліміна, О.В. Романюк розглянули основні типи нереляційних баз даних і конкретні приклади їх використання провідними компаніями. Проведене дослідження довело, що NoSQL (нереляційні) бази даних є потужним інструментом обробки великої кількості слабоструктурованих або неструктурованих даних, для яких на першому місці стоїть швидкість обробки, а не повнота даних. Також викликає інтерес робота [4], де науковцями здійснено порівняльний аналіз реляційних і NoSQL баз даних. С.С. Бучик та А.В. Швед [5] розробили архітектуру системи управління відносинами із клієнтами на прикладі CRM-системи Creatio.

Проте, враховуючи описані наукові набутки, питання обґрунтування механізмів реалізації REmote DIctionary Server як однієї з головних структур мережевого сховища даних залишається відкритим і потребує детального опрацювання.

Постановка завдання. Здійснити дослідження REmote DIctionary Server як однієї з головних структур мережевого сховища даних.

Викладення основного матеріалу дослідження. Кешування даних у пам'яті може бути однією із найефективніших стратегій для покращення загальної продуктивності програми та зниження витрат на базу даних.

Кешування – це техніка, яка використовується для прискорення часу відповіді програми та допомоги програмам у масштабуванні, розміщуючи часто необхідні дані дуже близько до програми. У сучасних обчислювальних середовищах процеси запускаються на різних рівнях, від низькорівневих процесів, які вбудовані в апаратне забезпечення, до високорівневих абстракцій, таких як кластери [6]. Redis – сервер із відкритим вихідним кодом зі структурою даних у пам'яті, котрий часто використовується як розподілений спільний кеш (на додаток до використання як посередник повідомлень або бази даних), оскільки він забезпечує справжню відсутність стану для процесів додатків, одночасно зменшуючи дублювання даних або запити до зовнішніх джерел даних.

Redis широко використовується у масштабованих програмах, які працюють із великим обсягом даних. У сучасній обробці великих даних перерахування у пам'яті стало популярним, оскільки використовується для збільшення ємності та високої пропускної здатності основної пам'яті. При створенні розподілених програм, які вимагають низької затримки та масштабованості, дискові бази даних можуть викликати низку проблем. Швидкість отримання даних із диска плюс доданий час обробки запиту зазвичай виражає швидкість відповіді запиту у двозначних мілісекундах. Це передбачає, що користувач має постійне навантаження і база даних працює оптимально [7].

Незалежно від того, чи розподіляються дані у базі даних NoSQL на основі диска, чи вертикально збільшуються у реляційній базі даних, масштабування для надзвичайно високих показників читання може бути дорогим.

Також може знадобитися кілька реплік читання бази даних, щоб відповідати тому, що може забезпечити один вузол кешу в пам'яті з погляду запитів за секунду. Хоча реляційні бази даних забезпечують чудовий засіб для моделювання відносин, вони не є оптимальними для доступу до даних [8].

Трапляються випадки, коли програми можуть захотіти отримати доступ до даних у певній структурі або представленні, щоб спростити отримання даних і підвищити продуктивність

програми. Перш ніж запровадити кешування бази даних, багато архітекторів та інженерів витрачають багато зусиль, намагаючись отримати якомога більше продуктивності зі своїх баз даних, однак існує обмеження продуктивності, якої можна досягти за допомогою дискової бази даних і намагатися вирішити проблему за допомогою неправильних інструментів непродуктивно. Наприклад, велика частина затримки запиту до бази даних продиктована фізикою отримання даних із диска. Тільки Redis не може продуктивно працювати у реальному інженерному проекті, але комбінація Redis і SQL, наприклад, MySQL, є перспективним напрямком. Redis є швидким сервером введення-виведення за рахунок своєї функції «кешування». MySQL – популярна реляційна SQL, яка має високу продуктивність і програмну підтримку. Очевидною слабкістю Redis є обмежений обсяг сховища. Пам'ять не є ідеальним рішенням для зберігання великої кількості даних. Найпоширенішим рішенням є те, що MySQL або інша база даних SQL буде функціонувати як основне сховище даних, тим часом Redis може бути призначений для сфер, які потребують більш гнучкої та ефективної продуктивності [9].

Що стосується реального проекту розвитку, то зазвичай лише невелика кількість даних є неструктурованими, оскільки більшість інформації знаходиться у певній структурі. Крім того, хмарні обчислення швидко розвиваються, і локальне сховище буде менш важливим у майбутньому. Можливо, на локальному сервері працюють лише основна програма та дані. Якщо так, то Redis відіграватиме набагато більшу роль у розвитку.

І реляційні, і NoSQL бази даних є базами даних у пам'яті, які надають різні механізми зберігання та пошуку даних.

Відповідно до моделі даних NoSQL, сховища даних згруповані у чотири категорії:

- сховища даних ключ-значення;
- сховища документів;
- сховища сімейства стовпців;
- бази даних графіків [10].

Сховище ключ-значення: дані зберігаються як пари ключ-значення. Ця структура даних також відома як «хеш-таблиця», де дані витягуються за допомогою ключів. Найбільш відомими прикладами сховищ ключів і значень є Redis.

Сховище документів: дані зберігаються у колекціях, які містять пари ключ-значення, котрі інкапсулюють пари значень ключів у JSON (об'єктна нотація Javascript) або документи, подібні до JSON.

Сімейство стовпців: дані зберігаються як набір рядків і стовпців, де стовпці згруповані відповідно до зв'язку даних.

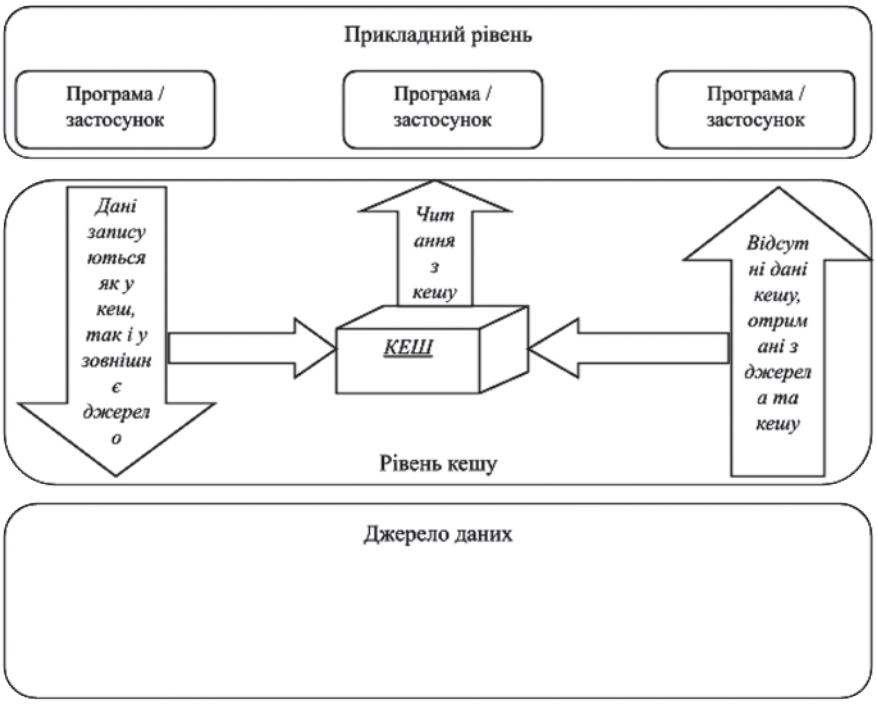
Графічні бази даних є категорією бази даних NoSQL, яка зберігає дані у вигляді графіка. У графових базах даних граф складається із двох речей: вузли діють як сутності чи об'єкти, а ребра діють як зв'язок між сутностями чи об'єктами.

Redis зазвичай відомий як база даних ключ-значення. Redis також має п'ять основних структур даних: Set, Zset, List, Hash, String, які надають йому гнучкість. Порівняно із SQL, навіть з іншими NoSQL, ця група структур даних виділяє Redis. Хоча ця функція може бути особливістю Redis, вона також може бути складною для розробників.

На відміну від MongoDB, Redis добре працює зі швидкістю, його дані зберігаються у пам'яті, тому продуктивність гарантується, проте вартість зберігання є високою та здається нереальним зберігати велику кількість даних в пам'яті. Redis дає нам своє рішення оптимізації, ключ EXPIRE. Після закінчення часу очікування ключ буде автоматично видалено. У термінології Redis часто кажуть, що ключ із асоційованим тайм-аутом є нестабільним. Як описано в офіційному документі Redis [10], Redis може встановити тайм-аут для ключа, і ключ буде видалено через певний час. Робота у пам'яті є однією з найбільших відмінностей порівняно з іншими NoSQL. Відносно менше даних, що зберігаються, і неймовірно висока швидкість гарантують ефективність передачі даних. Користувач може мати досить хороший досвід, використовуючи Redis як серверну частину. Соціальні медіа мають високі вимоги до швидкості доступу, і висока синхронність стає однією із найскладніших проблем для будь-якого веб-сайту сучасних мереж.

Інші NoSQL, такі як MongoDB, мають власні відповідні робочі області, які вдосконалюють функцію реляційної бази даних, як і Redis. Redis чудово кешує та працює з невеликою кількістю даних, однак Redis – це набагато більше, ніж кеш.

Як тільки процес програми використовує зовнішнє джерело даних, його продуктивність може бути вузьким місцем через пропускну здатність і затримку зазначеного джерела даних. Коли використовується повільніше зовнішнє джерело даних, дані, до яких часто звертаються, періодично тимчасово переміщуються у швидше сховище, розташоване ближче до програми, щоб підвищити продуктивність програми. Це швидше проміжне сховище даних називається кешом



Дискове сховище
СУБД
API

Рис. 1. Схема високорівневої архітектури кешу програми

програми, термін походить від французького дієслова «cacher», що означає «ховати». На рис. 1 показано високорівневу архітектуру кешу програми.

Основна мета кешу – скоротити час, необхідний для доступу до даних, що зберігаються за межами основної пам'яті програми. Крім того, кешування також є надзвичайно потужним інструментом для масштабування зовнішніх джерел даних і пом'якшення впливу на них сплесків використання. Кеш на стороні програми ефективно зменшує всі потреби у ресурсах, необхідні для обслуговування даних із зовнішніх джерел, таким чином звільняючи ці ресурси для інших цілей. Без використання кешу програма взаємодіє із джерелом даних для кожного запиту, тоді як із використанням кешу потрібен лише один запит до зовнішнього джерела даних із наступним доступом до кешу.

Крім того, кеш також сприяє доступності програми. У зовнішніх джерелах даних можуть виникати збої, які призводять до погіршення або припинення обслуговування. Під час таких збоїв кеш-пам'ять все ще може передавати дані додатку і таким чином зберігати свою доступність. Кеш програми призначений для зберігання даних, що потребують додаткової обробки під час виконання. Хоча кожна програма є унікальною і потенційно може зберігати будь-які дані у своєму кеші, програма зазвичай використовує кеш для:

– налаштувань конфігурації. Інформація, потрібна додатку для прийняття рішень щодо завантаження та виконання, часто зберігається у відносно повільному сховищі (наприклад, текстові файли на диску або спільному сховищі конфігурації). Зберігаючи кешовані копії цих налаштувань, програма може отримати доступ до даних із мінімальною затримкою;

– даних локалізації та інтернаціоналізації. Додатки для користувачів часто надають локалізовані варіанти свого інтерфейсу для пристосування міжнародної аудиторії. Дані інтернаціоналізації зазвичай зберігаються поза програмою, тому ними можна керувати окремо. Оскільки ці дані необхідні для обслуговування більшості запитів, їх кешування покращує час відповіді програми;

– шаблонів і частково відтворених відповідей. Багато програм складають свої відповіді, додаючи дані до шаблонів і попередньо підготовленого вмісту (наприклад, фрагменти HTML і фрагменти JSON);

– результатів багаторазового використання обчислювальних функцій. Іноді робочий процес програми вимагає створення ресурсомістких результатів. Як тільки ці результати отримані,

є випадки, коли результати пізніше можуть бути використані повторно, наприклад, для виконання часткових агрегатів. Кеш є ідеальним проміжним середовищем для збереження таких результатів між запитами;

– даних сеансу. Кешування даних сеансу користувача є невід'ємною частиною створення адаптивних додатків, які можуть масштабуватися. Оскільки кожна взаємодія користувача вимагає доступу до даних сеансу, збереження їх у кеші забезпечує найшвидший час відповіді користувачеві програми. Кешування даних сеансу на рівні програми перевершує альтернативні методи. Наприклад, збереження сеансів на рівні балансувальника навантаження практично змушує всі запити у сеансі оброблятися одним сервером додатків, тоді як кешування дозволяє обробляти запити будь-яким сервером додатків без втрати станів користувачів;

– даних СУБД. Більшість традиційних баз даних розроблені для забезпечення надійної функціональності, а не швидкості у масштабі. Кеш програми часто використовується для зберігання копій таблиць пошуку та відповідей на дорогі запити від СУБД як для підвищення продуктивності програми, так і для зменшення навантаження на джерело даних;

– відповідей API. Сучасні програми створюються з використанням слабо пов'язаних компонентів, які взаємодіють через API. Компонент програми використовує API, щоб робити запити на обслуговування від інших компонентів як всередині (наприклад, в архітектурі мікросервісів), так і за межами (наприклад, у разі використання SaaS) самої програми. Коли відповідь API може бути збережена у кеші, навіть якщо лише на відносно короткий термін, продуктивність програми покращується за рахунок уникнення взаємодії між процесами;

– об'єктів програми. Будь-який об'єкт, який генерує програма і який може бути використаний пізніше, може бути збережений у кеші. Метод кешування об'єктів залежить від програми, але більшість кешів дозволяють зберігати як серіалізовані, так і вихідні дані об'єктів. Типовим прикладом часто кешованого об'єкта програми є профіль користувача, що складається з даних із кількох джерел.

Висновки. У роботі проведено дослідження REmote DIctionary Server як однієї з головних структур мережевого сховища даних, вона використовує систему зберігання даних у пам'яті «ключ-значення» Redis, яка працює з великими даними. Порівняно з реляційною базою даних Redis володіє надзвичайно швидким процесом читання. Основна проблема полягає у тому, що попередній фреймворк Client Server надає погане виконання процесу читання та запису щодо пропускної здатності та затримки, оскільки всі сервери використовують свою власну пам'ять для роботи із загальною процедурою, що займає багато часу.

NoSQL відіграватиме важливу роль у майбутній розробці та приверне більше уваги компаній-гігантів і стартапів. І наукові кола, і промисловість високо оцінюють майбутнє NoSQL. Безсумнівно, NoSQL не спрямований на повну заміну SQL, але допомагає реляційній базі даних працювати краще. Крім цього, NoSQL може використовувати абсолютно нову сферу.

Список літератури:

1. Жолткевич Г.Г. Моделювання процесів реплікації даних у розподілених сховищах : дис. … канд. тех. наук : 01.05.02 / Харківський Національний Університет імені В.Н. Каразіна ; Інститут проблем машинобудування ім. А.М. Підгорного НАН України, Харків, 2021. 150 с.

2. Müllenbach S., Kern-Bausch L., Kolonko M. Conceptual Modeling Language Agila Mod. *Herald of Advanced Information Technology.* 2019. Vol. 2. № 4. P. 246–258.

3. Ісаков А.В., Ліміна В.Ю., Романюк О.В. Сфери застосування нереляційних баз даних. *Матеріали XLIX науково-технічної конференції підрозділів ВНТУ (Вінниця, 27–28 квітня 2020 р.)*. Вінниця, 2020. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9416.

4. Ковтун Б.В., Манич А.М., Романюк О.В. Порівняльна характеристика реляційних та NoSQL баз даних. *Матеріали XLIX науково-технічної конференції підрозділів ВНТУ, Вінниця, 27–28 квітня 2020 р*. 2020. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9907 (дата звернення: 09.12. 2020)

5. Бучик С.С., Швед А.В. Архітектура системи управління відносинами з клієнтами на прикладі CRM-системи Creatio. *Прикладні системи та технології в інформаційному суспільстві* : зб. тез доповідей і наук. повідомл. учасників IV Міжнародної науково-практичної конференції (Київ, 30 вересня 2020 р.) / за заг. ред. В.Л. Плескач, В.Л. Міронова. Київ : Київський нац. ун-т імені Тараса Шевченка, 2020. С. 30–34.

6. Zhang J., Yao Z., Feng, J. NCRedis: An NVM-Optimized Redis with Memory Caching. *International Conference on Database and Expert Systems Applications* Springer, Cham. 2021. P. 70–76.

7. Chopade R., & Pachghare V. A data recovery technique for Redis using internal dictionary structure. *Forensic Science International: Digital Investigation*. 2021. Vol. 38. P. 301218. doi: 10.1016/j.fsidi.2021.301218.
8. Tkachenko V., & Lukianiuk S. Analysis of the use of the Redis in the distributed order processing system in the restaurant network. *Technology Audit and Production Reserves*. 2021. Vol. *5* (2). P. 61.
9. Ahmad K., Javed M. Hands-On Redis. *NoSQL: Database for Storage and Retrieval of Data in Cloud.* 2017. P. 355–364. Chapman and Hall/CRC.
10. Gutierrez F. Messaging with Redis. In Spring Boot Messaging Apress, Berkeley, CA. 2017. P. 81–92.

Abharian Yu.S. REMOTE DICTIONARY SERVER AS ONE OF THE MAIN STRUCTURES OF THE NETWORK DATA WAREHOUSE

The article examines Remote Dictionary Server as one of the main structures of a network data warehouse. It is noted that Redis is a fast repository of key-value data in open source memory. It is emphasized that Redis provides response time at the fraction of a millisecond level and allows programs running in real time to execute millions of requests per second. Such programs are needed in the areas of games, advertising technology, financial services, healthcare and IoT. It is emphasized that today Redis is one of the most popular open source cores, and for five years in a row it has been called the "favorite" database from Stack Overflow. All Redis data is stored in memory, which provides low latency and high bandwidth access to data. Unlike traditional databases, memory repositories do not require moving to disk, which reduces core latency to microseconds. As a result, in-memory data stores can multiply the number of operations performed and reduce response time. The result is extremely high performance. Read and write operations take less than a millisecond on average, with speeds reaching millions of operations per second. The paper emphasizes the advantages of using Remote Dictionary Server, which include: flexible data structures (unlike other key-value repositories that support a limited set of data structures, Redis supports a huge variety of data structures to meet needs of various programs); simplicity and convenience (Redis allows you to write the most complex code with fewer simple lines); replication and persistent storage (Redis uses a master slave architecture and supports asynchronous replication, in which data can be copied to multiple slave servers); high availability and scalability (Redis offers a "master slave" architecture with one master node or cluster topology); all open source tools.

Key words: *data warehouse, network, server, database, Remote Dictionary Server, structure, information.*