Engineer

Redis

🧠 швидке сховище для того, що живе недовго

Що це таке

Redis — це сервер бази даних, який тримає весь свій набір даних в оперативній пам'яті у вигляді типізованих структур і дає багатьом процесам застосунку читати та змінювати їх через мережу, по одній короткій команді за раз. Сам проєкт називає себе сервером структур даних, англійською data structure server. Безплатна редакція називається Redis Open Source, а компанія Redis Ltd також продає Redis Cloud і Redis Software за комерційними ліцензіями.

Кожен ключ містить одне значення одного типу. Рядки — це просто байти, які водночас служать лічильниками; хеші — записи з полів і значень; списки зберігають порядок, у якому надійшли елементи; множини містять унікальних членів, а впорядковані множини ранжують членів за числовою оцінкою. Далі перелік типів продовжують потоки (streams), документи JSON, геопросторові індекси, часові ряди, векторні множини та імовірнісні структури, як-от фільтри Блума і HyperLogLog.

Команди працюють безпосередньо з цими типами. `INCR` додає одиницю до лічильника й повертає нове значення, `SADD` кладе членів у множину, а `SPOP` виймає одного. Будь-якому ключу можна задати термін життя, після якого Redis сам його видалить, і такий ключ Redis називає volatile. Коли кілька кроків мають виконатися як один, усередині сервера запускається скрипт на Lua, і Redis гарантує, що поки він працює, не виконується нічого іншого.

Дані живуть у пам'яті, а диск необов'язковий. Redis може записувати знімки стану, так звані файли RDB, або дописувати кожну зміну в журнал тільки на додавання (AOF), який він відтворює під час запуску, або робити і те, й інше,

або ні того, ні іншого. Репліки копіюють основний сервер і можуть обслуговувати читання. Стеля пам'яті, `maxmemory`, і політика витіснення визначають, що станеться, коли даних стане більше, ніж відведено місця.

Ліцензія змінилася з Redis 8. Починаючи з цього випуску, Redis Open Source пропонують на вибір за трьома ліцензіями: RSALv2, SSPLv1 і AGPLv3. Перші дві дають доступ до коду (source available), але не є ліцензіями open source, а AGPLv3 є.

Застосуйте поняття до рішення

Застосуйте «Що таке Redis?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Пошук запитів і відсутніх індексів у системі, що рік за роком важчає, та виправлення тих, що справді мають значення.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. 30 днів даних про електроенергію за 180 мс — 180 ms на агрегацію місяця показань лічильників

30 днів даних про електроенергію за 180 мс Зв'язатися з нами →

Чому це важливо

Вебзастосунок рідко працює одним процесом. За балансувальником навантаження стоять кілька інстансів, кожен зі своєю пам'яттю, і все, що лежить у цій пам'яті — кількість спроб входу, сесія, закешована сторінка, — існує лише в тому процесі, якому випало обробити запит. Ліміт, порахований так, легко обійти, потрапивши на інший інстанс. Redis дає всім інстансам одне спільне місце для такого стану, досить швидке, щоб звертатися до нього на кожному запиті.

Атомарні команди важать не менше за швидкість. Два інстанси, які кожен читають лічильник, додають одиницю і записують його назад, можуть обидва записати те саме число, і одне збільшення зникне. `INCR` робить читання і запис усередині сервера одним кроком, тож сотня одночасних запитів залишає лічильник рівно на сто більшим. Скрипт на Lua поширює ту саму гарантію на коротку послідовність: перевірка і зміна, які не можна розірвати, як-от переконатися, що місце ще є, і зайняти його, виконуються як одне ціле.

Третя складова — термін життя ключів. Вікно для ліміту запитів, сесія чи одноразовий код мають зникати, коли їхній час минув, і з терміном життя на ключі прибирання стає справою сховища, а не запланованого завдання.

Ціна — надійність збереження, і її варто знати в числах до того, як обирати. З журналом AOF і його типовою політикою одного fsync на секунду збій може коштувати приблизно секунди записів. Лише зі знімками, які зазвичай роблять раз на кілька хвилин, документація радить бути готовими втратити дані за останні хвилини. Реплікація за замовчуванням асинхронна, тож основний сервер може підтвердити запис і впасти раніше, ніж його отримає хоч одна репліка. Тому Redis добре підходить для стану, який можна відбудувати або втратити без шкоди, і погано — для єдиної копії замовлення, платежу чи балансу. Найважливіше питання проєктування — що робить застосунок, коли Redis не відповідає або забув щось, що йому повідомили.

Як це виглядає на практиці

У публічному каталозі компаній і фахівців Redis — спільний стан для лімітів: 22 обмежувачі частоти тримають свій стан у Redis, тож ліміт діє для всіх інстансів, а не живе в одному процесі, звідки його тривіально обійти. Кожен обмежувач зроблено під шлях, який він захищає, бо спроба входу, пошук і масовий експорт стають зловживанням на дуже різних швидкостях. Вирішальний вибір — що відбувається, коли Redis немає: обмежувачі закриваються при збої (fail closed), і трафік відхиляється, а не пропускається. Сторінка прямо називає ціну. Проблема з Redis стає проблемою, яку бачать користувачі, і це прийнято свідомо, бо альтернатива — проблема з Redis, що стає проблемою з рахунками.

Шлюзом перед базою даних Redis стає в нотатці про продаж обмеженого запасу, де лічильник у Redis, який зменшує Lua-скрипт, пропускає приблизно стільки покупців, скільки є одиниць, а решту відвертає за мікросекунди. Скрипт робить перевірку і списання одним кроком. Нотатка уважна до того, що шлюз може пообіцяти. Redis реплікує асинхронно, тож після перемикання на репліку новий основний сервер може забути частину пропусків і впустити трохи забагато. Кожен продаж усе одно вирішує умовний `UPDATE` у базі, а процес звірки відновлює шлюз за числом із бази. Коли одиниць небагато і вони унікальні, як-от нумеровані місця, їхні ідентифікатори можна тримати в множині Redis, і `SPOP` видає їх по одному, без лічильника, за який треба битися.

Найбуденніша роль — звичайний кеш. Інфраструктура Azure, описана в Bicep, будує кожне середовище з тих самих шаблонів, testing, staging і product, і в кожному є Redis для кешування поряд із PostgreSQL Flexible

Server, Container Apps, у яких працюють api та вебсайт, і Front Door з політикою WAF. Оскільки кеш оголошено в шаблонах разом з усім іншим, змінити його — це pull request, як і будь-яку іншу зміну середовища.

Де виникають проблеми

Більшість проблем із Redis походить від того, що швидке й забудькувате сховище вважають надійним або розраховують, що воно пам'ятає сказане йому про час.

Ключ, який мав зникнути, лежить вічно. `SET` замінює значення і скидає термін життя, що був у ключа, тож код, який оновлює закешоване значення звичайним `SET`, перетворює десятихвилинний запис на постійний:

```
> SET session:42 "v1" EX 600
OK
> SET session:42 "v2"
OK
> TTL session:42
(integer) -1
```

Передавайте термін життя в кожному записі, якому він потрібен, або додавайте `KEEPTTL`, доступний з версії 6.0.0, щоб зберегти попередній. Такий самий витік ховається в лічильнику з `INCR` і окремого `EXPIRE` після нього: якщо друга команда не дійде, ключ не зникне ніколи. Надсилайте обидві в одному блоці `MULTI` та `EXEC` або в одному скрипті на Lua.

Записи починають повертати помилки, а читання досі працюють. Дані досягли `maxmemory`, а політика витіснення — `noeviction` або одна з політик `volatile-` на наборі даних, де жоден ключ не має терміну життя, що діє так само. Обирайте політику під задачу: `allkeys-` для чистого кешу, де може зникнути будь-який ключ, а `volatile-` лише тоді, коли закешовані ключі ділять інстанс із ключами, які мають залишитися. Для такої суміші документація радить розглянути два окремі інстанси.

Записів, які Redis підтвердив, бракує після перезапуску чи перемикання. Збій стався всередині вікна fsync, або репліку підвищили до основної раніше, ніж до неї дійшов запис. Так Redis задуманий, тож розв'язок у тому, що в

ньому зберігати. Тримайте все, що має пережити збій, у базі даних із надійними комітами, ставте Redis перед нею як шлюз чи кеш і звіряйте одне з одним з боку бази, як це робить нотатка про запас.

Усі клієнти завмирають одночасно. Працює довгий скрипт на Lua, а Redis блокує всю іншу роботу сервера на весь час виконання скрипту. Тримайте скрипти короткими й обмеженими і не пускайте в них цикли по великих колекціях.

Обмежувач пропускає трафік під час збою. Клієнтський код трактує помилку від Redis як «ліміт не досягнуто», тож обмежувач відчиняється саме тоді, коли його сховище недоступне. Оберіть поведінку при збої свідомо й перевірте її, зупинивши Redis: обмежувач, що стереже дорогий шлях, має відмовляти, як згадані вище 22 обмежувачі, з тією ціною, що проблема з Redis стає видимою для користувачів.

Що почитати далі

- Develop with Redis — стартова сторінка проєкту: команди, щоб спробувати, клієнтські бібліотеки за мовами і типи даних.
- Redis persistence — знімки, журнал AOF, політики fsync і що кожна з них може втратити.
- Redis replication — як репліки йдуть за основним сервером і чому вікно для втрати даних лишається завжди.
- Key eviction — `maxmemory`, усі політики витіснення і поради, як обрати.
- EXPIRE — терміни життя ключів і те, які команди їх скидають, а які залишають.
- Scripting with Lua — атомарне виконання скрипту на сервері і те, що він блокує на цей час.

Словник

Обирайте за фактичними умовами

Надайте репрезентативний набір даних, запитання до нього та поточне вузьке місце. Узгодимо вимірювання якості, пропускної здатності та відновлення.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Redis?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/redis/