Redis

План

Типи кешування;

Кешування в .NET;

Яке в нас було кешування і проблеми;

Що таке Redis?

Деталі його реалізації та підтримки;

Типи даних та операції над ними;

Приклад використання Redis в проекту типу Twitter.

Типи кешування

Web caching;

Application / Output cache;

Data caching;

Distributed cache.

Web caching

Головна мета: зменшити трафік і надати швидкий доступ користувачу до вже переглянутих матеріалів;

Кешування відбувається на стороні клієнта.

Application / Output Cache

Головна мета: зниження навантаження на сервер;

Зазвичай кешується уже готовий HTML;

Можливість кешувати дані для цілих груп користувачів по різним критеріям.

Data caching

Головна мета: зниження навантаження на базу даних;

Спрощення доступу до необхідних, або таких, на розрахунок яких необхідні значні витрати часу, даних.

Distributed cache

Головна мета: зниження навантаження на сервер;

Винесення даних на окремі сервера/кеші;

Переважно, використовується високонавантаженими системами.

Web caching .NET

OutputCache

- Лише для сторінок;
- Можливість зберігати в файлі або пам’яті;
- Зберігати декілька копій в залежності від вхідних параметрів;
- Можливість вказати де саме зберігати кеш: на клієнті чи на сервері;
- Вказувати тривалість існування;

Data caching .NET

System.Web.HttpContext.Current.Cache

Плюси

- Key-Value колекція;
- Можливість вказувати залежності, які ініціюють оновлення;
- Можливість вказувати пріоритет. Об’єкти з вищим пріоритетом будуть збережені, якщо на сервері буде недостатньо пам’яті;
- Можливість вказувати callback при видаленні елементу

Мінуси

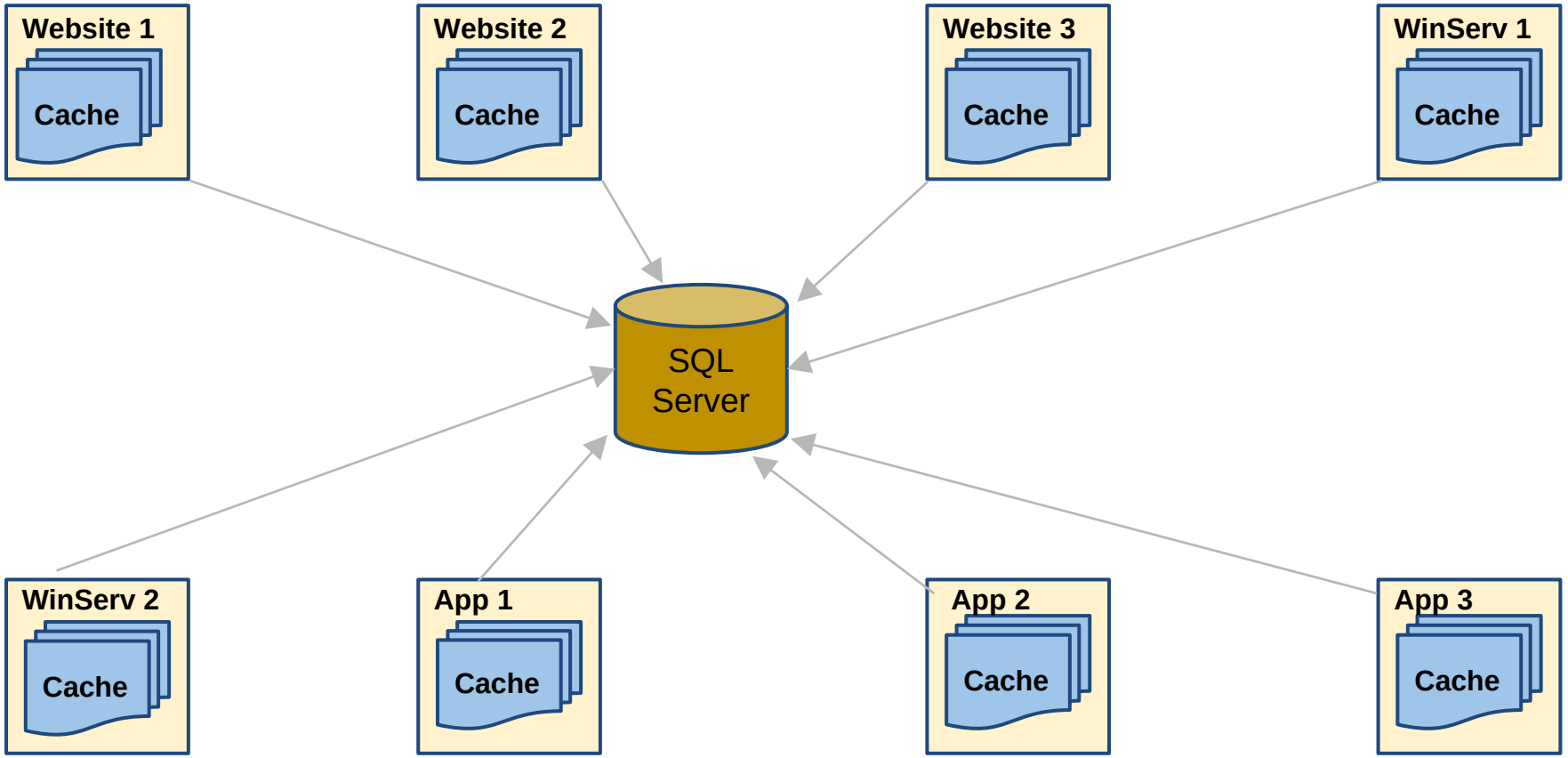
- InProcess

Структури, які ми використовували

AsyncCache: зберегігання даних і їх оновлення кожні n хвилин

LruCache

MruCache


Website 1
Cache
Website 2
Cache
Website 3
Cache
WinServ 1
Cache
SQL
Server
WinServ 2
Cache
App 1
Cache
App 2
Cache
App 3
Cache

Що в нас є?

Зменшене навантаження на базу;

Часті оновлення все одно створюють навантаження;

Всі кеші InProcess;

Можуть бути застарілі дані в кеші;

Неможливо часткове/повне оновлення для всіх частин системи одночасно;

Що нам треба?

Один кеш на всі частини системи;

Відсутність непотрібних оновлень;

Можливість часткових оновлень;

Відсутність застарілих даних;

Кеш, який живе окремо від процесу;

Підтримка Pub/Sub.

Знайомтеся, Redis

REmote DIctionary Server

Key/value сховище. Але підримує багато інших типів:

lists - колекція елементів текстового типу;

sets - колекція *унікальних* несортованих елементів текстового типу;

sorted sets - схоже на sets, але окрім самого елементу має ще один елемент, який вказує на порядок при сортуванні;

hash tables - звичайні хеш таблиці;

bit arrays - представлення рядків як масиву байтів.

OpenSource, але немає *офіційної** підтримки Windows. Linux - рекомендована система.

Головні функції та переваги

Всі дані в пам’яті;

Але дані можуть поступово можуть переміщатися на диск. А можуть і відразу;

Чудово працює з високими навантаженнями (>=150k / sec);

Всі операції атомарні;

Є підтримка транзакцій;

Підтримка Pub/Sub;

O(1) - складність для більшості операцій;

Деталі реалізації

Redis - це TCP сервер, який використовує клієнт/сервер модель;

Використовує власний протокол: RESP (REdis Serialization Protocol);

Всі операції, навіть ті, які виконують декілька дій - атомарні;

Атомарність досягається завдяки тому, що Redis - однопоточний. В певну одиницю часу виконується лише одна операція.

Масштабування

Головна необхідність - пам’ять, а не CPU;

Дозволяє мати набагато більші бази, сумуючи про цьому обчислювальні здатності декількох комп’ютерів;

Варіанти:

На стороні клієнта;

Проксі - додатковий рівень, який сам обирає кластер/машину;

Маршрутизація запитів - відправлення запиту на випадкову машину, яка в свою чергу гарантує перенаправлення запиту на необхідну машину.

Обмеження масштабування

Операції з декількома ключами, такі як перетин множин, зазвичай не підтримуються;

Транзакції з двома ключами теж не підтримуються;

Збільшується складність підтримки та резервних копій.

Копіювання

Редіс використовує копіювання процесів:

- Виконує команду fork;
- Новостворений процес виконує необхідну операцію;
- Дочірній процес заміщає собою головний процес.

Дочірні та батьківські процеси не заважають один одному;

Персистентність даних

Можна взагалі цим не користуватися;

***.RDB файл** - представлення кешу з пам’яті як окремого файлу;

AOF (AppendOnlyFile) - записує кожну операцію запису, яка потім буде відтворена при наступному запуску серверу;

Можна об’єднати два рішення на одному сервері.

RDB файли

+

Оптимізовані для швидких операцій читання та запису;

Чудово підходить для резервних копій;

Не впливає на загальну швидкість роботи;

Не потрібно оновлювати дані в пам’яті після перезапуску серверу;

-

Не підходить, якщо втрата даних недопустима;

Копіювання процесу може використовувати багато часу

AOF

+

Більш гнучке та менш затратне в плані ресурсів рішення;

Запис в файл відбувається в окремому процесі;

Якщо файл стає надто великим, Redis може його створити новий, взявши мінімальну кількість необхідних даних з попереднього файлу;

Зручний формат, можна легко розпарсити.

fsync() кожної секунди не впливає на продуктивність.

-

Зазвичай більше, ніж RDB;

Типи даних

Ключ

string: максимальний розмір 512 мб. Але це не є гарною ідеєю.

Значення

String

List

Hash

Set

Sorted set

Bitmaps

String

Використовується для зберігання будь який рядків, в тому числі json, xml та ін.

set - встановити значення.
заміняє існуюче значення будь-якого типу на нове;
nx, xx дозволяють контролювати поведінку в таких випадках
get - отримати значення
incr - додати 1
decr - відняти 1
incryby, decrby - додати N

> set key newvalue
OK

>get key
newvalue

>set intvalue 100
OK
>incr intvalue
101
>incrby intvalue 50
151

> set mykey newval nx
(nil)
> set mykey newval xx
OK

set - встановити значення;
get - отримати значення;
incr - додати 1;
decr - відняти 1;
incrby, decrby - додати N;
getset - встановлює нове значення, при цьому повертаючи попереднє;
incr, decr, incrby, decrby - повністю атомарні.
exists - повертає 1/0 при існуванні/відстуності значення для заданого ключа;
type - повертає тип значення для заданого ключа;
mset, mget - аналогічно get/set але для декількох елементів

>set justvalue 100
OK
>getset justvalue 199
100
>get justvalue
199

> set mykey hello
OK
> exists mykey
1
> del mykey
1
> exists mykey
0

> mset a 10 b 20 c 30
OK
> mget a b c
1) "10"
2) "20"
3) "30"

Ключі з обмеженим терміном життя

Існує можливість вказувати термін життя для значень незалежно від їхнього типу;
Дані зберігаються як *Дата видалення*, а не *Час до видалення.*
Інформація записується на диск.

expire - вказує час для видалення елементу без параметрів значення дорівнює одній мс;
persist - відміняє дію expire;
ttl - повертає час до видалення в секундах.
pexpire, pttl - аналогічно, але в мс.

> set key some-value
OK
> expire key 5
(integer) 1
> get key (immediately)
"some-value"
> get key (after some time)
(nil)

-
> set key 100 ex 10
OK
> ttl key
(integer) 9

List (LinkedList)

Максимальна кількість елементів - 2^{32};

Додавання елементу займає сталу одиницю часу;

Отримання елементу по індексу завжди O(n);

rpush, lpush - додавання елементу в кінець/початок

lrange (firstIndex, lastIndex) - отримання елементів.

-1 - останній елемент

-2 - передостанній …

ltrim - аналогічно до lrange, але видаляє усі елементи, які не входять в проміжок

```
> rpush mylist A
(integer) 1
> rpush mylist B
(integer) 2
> lpush mylist first
(integer) 3
> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
```

brpop, lrpop - можна запобігати непотрібним операціям *polling*;

0 як параметр означає: чекати наявності елементів вічність.

```
> brpop tasks 5
1) "tasks"
2) "do_something"
(5.00s)
```

Можна використовувати для

стрічка новин в соц. мережах (Twitter так робить);

взаємодії між процесами;

Hash

hash в Редіс - це такий же хеш, як і всюди;

hmset - створює об’єкт з масиву ключів/значень;
кількість ключів/значень обмежена лише об’ємом оперативної пам’яті
hget - отримує значення по ключу для заданого об’єкту;

```
> hmset user:1000 username antirez birthyear 1977 verified 1
OK
> hget user:1000 username
"antirez"
> hget user:1000 birthyear
"1977"
> hgetall user:1000
1) "username"
2) "antirez"
3) "birthyear"
4) "1977"
5) "verified"
6) "1"
```

Sets

Невідсортований набір унікальних рядків;
Невідсортований в даному випадку означає, що кожного разу Redis повертає дані в ***випадковому*** порядку;
Максимальна к-сть - $2^{32}-1$;
Отримання елементу завжди O(1);

sadd - додає елемент до колекції;
sismember - чи є дане значення елементом колекції;

Чудово підходить для збереження відношень між об’єктами та має багато реалізованих методів для роботи з декількома множинами.

```
> sadd myset 1 2 3
(integer) 3
> smembers myset
1. 3
2. 1
3. 2
-------------------------------------------------
-
> sadd news:1000:tags 1 5
(integer) 4
> sadd tag:1:news 1000
(integer) 1
> sadd tag:4:news 1000
(integer) 0
> smembers news:1000:tags
1. 5
2. 1
```

Sorted sets

Схоже на суміш Hash та Set;
Множина унікальних даних;
Сортуються за допомогою додаткового поля score;
A.score != B.score => return A.score > B.score
A.score == B.score => A > B у лексикографічному порядку

zadd додає новий елемент або оновлює score існуючого;

```
> zadd hackers 1940 "Alan Kay"
(integer) 1
> zadd hackers 1957 "Sophie Wilson"
(integer) 1
-------------------------------------------------
-
> zrange hackers 0 -1
1) "Alan Turing"
2) "Sophie Wilson"
-------------------------------------------------
-
> zrevrange hackers 0 -1
1) "Sophie Wilson"
2) "Alan Turing"
```

Сортування

Можна сортувати будь які колекції;
Операції сортування блокують єдиний потік, тому подумайте про те, щоб винести їх в окремий slave.
Існує можливість об’єднання колекцій в одну (join);
Результат сортування можна зберігати як окрему колекцію.

> SORT key
> SORT key BY pattern (eg. sort userIds BY user:*age)
> SORT key BY pattern GET othervalue

> SORT users BY user:*age GET user:*name

Bitmaps

Не є типом. Це лише набір бітових операцій;

setbit - встановлення біта;
getbig - отримання значення;
bitcount - повернення суми бітів, де значення дорівнює 1;
bitop - бітові операції AND, OR, XOR;
bitpos - пошук першого елементу, який дорівнює 1 або 0;

Ідеально підходить для
- аналізу даних в реальному часі;
- збереження великих масивів булевих значень, для яких критичне питання швидкодії.

```
> setbit key 10 1
(integer) 1
> getbit key 10
(integer) 1
> getbit key 11
(integer) 0
```

HyperLogLogs

Структура, яка використовується для знаходження унікальних елементів в колекції;
Потребує константного набору пам’яті: 12к байт максимум.

>pfadd hll a b c d
(integer) 1
> pfcount hll
(integer) 4

Publisher / Subscriber

Клієнти можуть підписуватися на канал і отримувати сповіщення при публікації нового повідомлення;
Операції підписки - O(1), Операція розсилки повідомлення - O(n);

Можна використовувати в чатах, тонких клієнтах, тощо.

> 1: subscribe feed:joe feed:moe
(integer) 1

//---- трохи очікування

> 2: publish feed:joe “Test message”
(integer) 1

> 1: “message”
“Feed:joe”
“Test message”

Транзакції

Реалізація транзакцій дуже легка через те, що в нас лише один потік;
Підтримують всі властивості транзакцій.

> MULTI
OK
> SET foo bar
QUEUED
> INCRBY num 12
QUEUED
> EXEC

> OK

Деякі поради стосовно ключів

Максимальна довжина рядка - 512 мб;

Використання довгих рядків в якості ключів - не дуже хороша ідея;

Але використання коротких рядків - теж не дуже хороша ідея;

Замість u1000flw використайте user:1000:followers

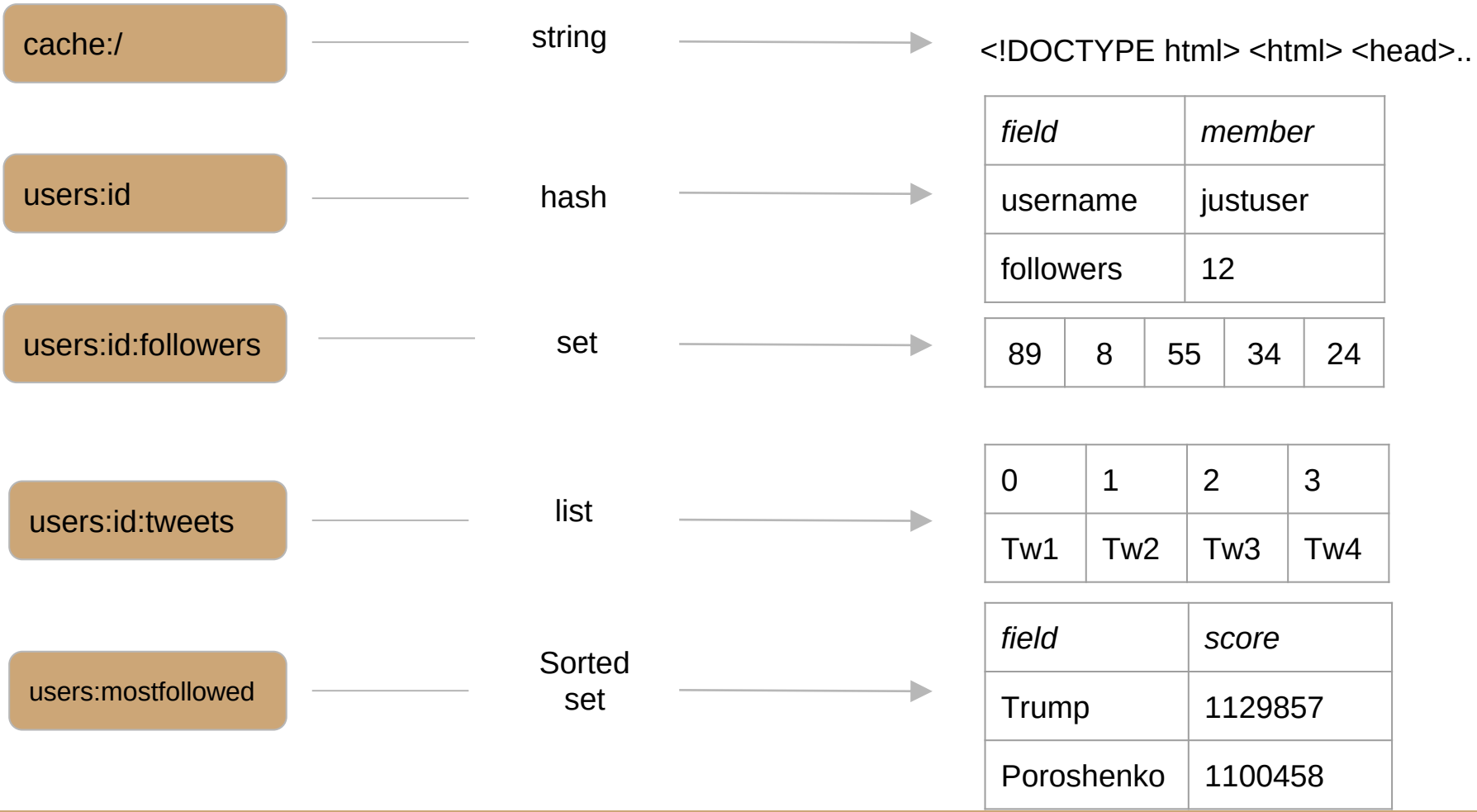
Постарайтеся знайти вірний баланс.

Ключі мають відповідати структурі:

object:id підходить для user:10;

для складнішої стуктури можна використовувати крапки - comment:1234:reply.to

Приклад реалізації соц. мережі


cache:/
string
<!DOCTYPE html> <html> <head>..
users:id
hash
field
member
username
justuser
followers
12
users:id:followers
set
89
8
55
34
24
users:id:tweets
list
0
1
2
3
Tw1
Tw2
Tw3
Tw4
users:mostfollowed
Sorted set
field
score
Trump
1129857
Poroshenko
1100458

> *get* talk:414:description
(nil)
>*set* talk:414:description “Description for event”
OK
>*get* talk:414:description
“Description for event”
>*expire* talk:414:description 5
(integer) 1
//--- after 5 seconds
>*get* talk:414:description
(nil)
>*getrange* talk:414:description 4 10
“Cription f”
>*incr* talk:414:views
(integer) 1

field	*member*
username	justuser
followers	12

> *hmset* users:1 username yuriy followers 10
OK
> *hgetall* users:1
1) “username”
2) “Yuriy”
3) “Followers”
4) 10

> *hincrby* users:1 followers 1
(integer) 11
> *hget* users:1 followers
11
> *hset* users:1 username yuriyk
(integer) 0

> *sadd* users:1:followers 89 8 55 34 24
(integer) 5

> *sadd* users:2:followers 8 24
(integer) 1

> *sinter* users:1:followers users:2:followers
1) 8
2) 24

> *rpush* users:1 tweets Tw1
(integer) 1

> *rpush* users:1:tweets Tw2
(integer) 2

> *lindex* users:1:tweets 1
“Tw1”