**Redis cache**

1. Redis là gì?

Redis là cơ sở dữ liệu ko quan hệ nhanh mà lưu trữ ánh xạ các key đến 5 loại dữ liệu khác nhau. Redis hỗ trợ lưu lâu dài trong mem trên đĩa, nhân bản để mở rộng hiệu năng đọc, và phân mảnh dữ liệu phía client để tăng hiệu năng viết.

Redis là dạng cở sở dữ liệu như NoSQL-không quan hệ. Trong redis, không có bảng, ko có định nghĩa dữ liệu hoặc cách ràng buộc của dữ liệu này liên quan đến dữ liệu khác trong redis.

Giống như memcached (“dạng lưu trữ key-value in-memory”), redis cũng lưu trữ ánh xạ key-value và có thể đạt được mức độ hiệu năng như memcached.

Khác memcached, redis hỗ trợ ghi data vào đĩa tự động bằng 2 cách khác nhau, và có thể lưu trữ data trong 4 cấu trúc ngoài các chuỗi key tring đơn giản như memcached lữu trữ.

Memcached thì đơn giản nhất, chỉ có dạng key-value, tất cả dữ liệu lưu trong RAM. Và lưu ý là Memcached chỉ là tầng cache, ko có tính persistent, tức là khó backup dữ liệu, và dữ liệu có thể mất (cái này là tỉ lệ eviction/reclaim xảy ra khi full bộ nhớ, memcached dùng thuật toán đẩy các key ít dùng hoặc key cũ nhất ra). Ngoài ra memcached còn có khái niệm slab, chunk, chunk grow factor, CAS v.v.

Redis thì phức tạp hơn, có nhiều kiểu dữ liệu string, hashes, sets, sorted sets, transaction, pub/sub v.v.và hình như còn cho phép scripting theo ngôn ngữ LUA (cái này mình chưa nghiên cứu). Kiểu string thì giống hết memcached, tức là key-value. Các kiểu kia có đặc trưng khác nhau. Kiểu string cho phép đặt expire time, nếu ko đặt thì key đó sẽ tồn tại mãi mãi. Redis giống memcached là memory-mapped, tức là tất cả dữ liệu đc đẩy hết vào RAM (có cách để đẩy vào swap), và cho phép theo một khoảng thời gian thì lưu một bản dump của tất cả dữ liệu vào 1 file. Redis hơn memcached ở một điểm nữa là cho phép replication. Nhưng kém memcached một điểm là hay bị dùng rất nhiều RAM. Dữ liệu ở file dump tầm khoảng 1GB, nhưng map vào RAM lên tầm 2GB

* Redis giải quyết nhiều vấn đề và cho phép redis có thể sử dụng cả primary database hoăc auxiliary database với các hệ thống lưu trữ khác nhau.

Redis sử dụng cả lưu trung gian lưu trữ primary và secondary cho dữ liệu, hỗ trợ nhiều trường hợp và nhiều loại kiểu truy vấn.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên | Loại | Dữ liệu lưu trữ | Dạng truy vấn | Tính năng |
| Redis | Cơ sở dữ liệu không quan hệ in-memory | String, list, set, hash, sorted set | Các lệnh cho mỗi loaij dữ liệu với các kiểu truy vấn phổ biến và hỗ trợ chuyển đổi | Publish/subscribe,  Master/slave replication, lưu trừ lâu dài trên đĩa, scripting |
| Memcached | In-memory key-value cache | Ánh xạ key-value | Các lệnh create, read, update, delete,.. | Server đa luồng để thực hiện bổ sung |

Memcached có thể thêm dữ liệu với APPEND và thêm dữ liệu như list. Nhưng không thể remove dữ liệu hoàn toán để tránh đọc/update/write. Nhưng với redis có thể sử dụng cả list hoặc set và sau đó add/remove items trực tiếp.

* Code ngắn hơn, dễ xử lý và nhanh hơn ( bởi vì không cần phải đọc data để update)

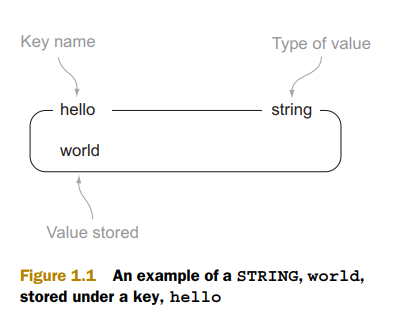
Cách thông thường sử dụng database là lữu trữ dữ liệu báo cáo long-term thành các cụm qua các khoảng thời gian cố định. Để thu thập những cụm này, thì thêm các row vào bảng và sau đó scan tất cả các row này để thu nhập các cụm dữ liệu, sau đó update các row tồn tại trong bảng cụm. Các row được thêm vào bởi vì thêm tính năng row nhanh nhưng cập nhật các row tồn tại trong bảng thì lại lâu. Trong redis, chúng ta phải tính các cụm dữ liệu trực tiếp sử dụng INCR commands (ghi ngẫu nhiên) cho dữ liệu redis luôn luôn nhanh, bởi vì dữ liệu luôn nằm trong mem, và truy vấn đến redis không thông qua parse/optimizer truy vấn.

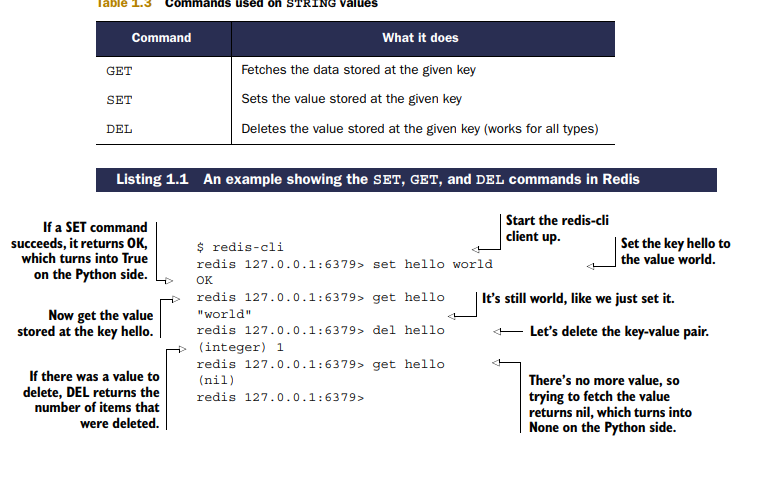
* Sử dụng redis tránh ghi dữ liệu tạm thời không cần thiết, tránh scan không cần thiết và xóa dữ liệu tạm thời, tối ưu hiệu năng.

Redis cho phép lưu trữ key mà ánh xá với bất kì 1 trong 5 dạng cấu trúc dữ liệu khác nhau: Strings, Lists, sets, hashes và zsets. Mỗi cái sẽ chi sẻ một vài commands, hoặc một vài command chỉ được dùng bởi 1 hoặc 2 cấu trúc.

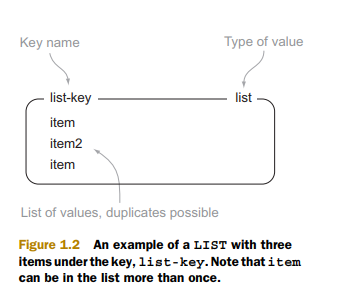
|  |  |  |
| --- | --- | --- |
| Cấu trúc | Bao gồm | Khả năng đọc/ghi |
| String | Strings, integers, float, point value | Hoạt động trên toàn bộ chuỗi. 1 phần, |
| LIST | Linked list của string | Push/pop ietms từ cả ends, trim dựa vào offset, đọc từng item hoặc đa, tìm hoặc remove item bởi value |
| SET | Thu thập không theo thứ tự của string độc nhất | Add, fetch, remove item đơn lẻ, check membership, giao nhau, hợp nhất |
| HASH | Bảng hash không theo thứ tựu của key-value | Add, fetch, remove item lẻ, fetch toàn bộ hash |
| Zsets | Ánh xạ theo thứ tự của string member đến các điểm nối, được sắp xếp theo score | Add, fetch, remove item lẻ, fetch items dự vào hạng score hoặc giá trị member |

1. String

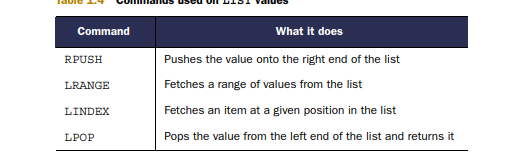




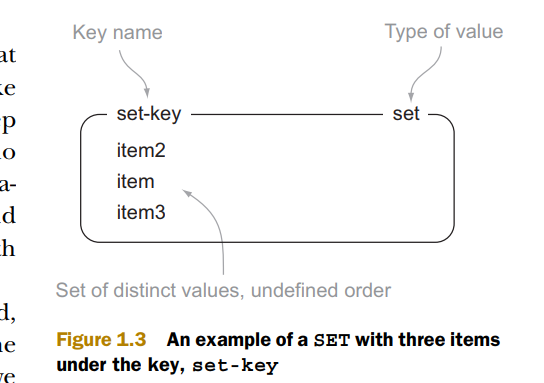
1. Danh sách



Kiểu lưu trữ key-value thì duy nhất redis hỗ trợ danh sách liên kết. List trong redis lưu trữ chuỗi các string theo thứ tự như string.



3.SETs



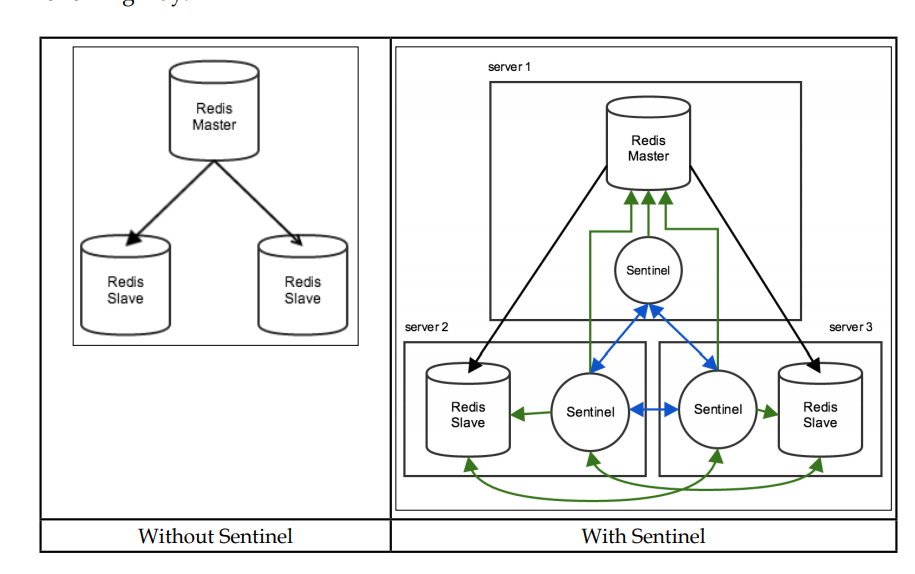
Cả list và set đều lưu trữ chuỗi các item. Nhưng khác list, set sử dụng bảng băm để lưu trữ các chuỗi duy nhất. Set không theo thứ tự nên có thểm hoặc pop từ dưới lên.

Set có nhiều cách sử dụng khác

1. Redis sentinel

Là hệ thống phân phối được thiết kế tự động hỗ trợ redis slave thành master nếu master bị lỗi. Sentinel không phân phối dữ liệu thông qua các node vì master node có tất cả data và slave có bản copy. Sentinal không phân phối dữ liệu lưu trữ

Kiến trúc thông thường bao gồm cài đặt 1 sentinel cho mỗi redis server. Sentinel là tiến trình tách ra từ redis server, và lắng nghe port của nó.



Sự khác biệt khi sử dụng redis sentinel là triển khai một interface khác mà yêu cầu phải cài đặt client redis hỗ trợ sentinel.

Client luôn luôn kết nối đến redis instance, nhưng nó cần truy vaasn đến redis để tìm redis instance sẽ kết nối nối đến.

Redis sentinel cung cấp hê thống có khả năng phục vụ liên tục và chịu nhiệt cao. Sentinel luôn quan sát master server, khi master sập, các sentinel sẽ loan truyền cho nhau 1 tín hiệu sdown báo master đã chết.

Khi đủ 1 lượng n sentinel xác nhận master chết, các slave sẽ gửi tín hiệu odown xác thực master die completely. Lúc này sentinel bầu chọn chọn ra 1 slave để nâng cấp lên làm master mới, đồng thời cập nhật các cấu hình theo hệ thống mới. Khi master kia sống lại, nó sẽ được tham gia với vai trò slave.

Cần ít nhất 3 sentinel

1. Chaỵ mô hình master-slave

Caì redis cho cả 3 node :

Yum install redis

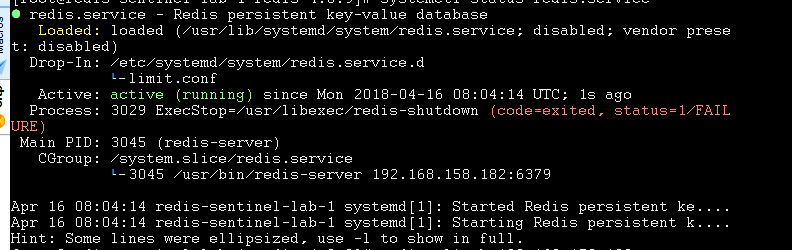
Gán địa chỉ ip của node

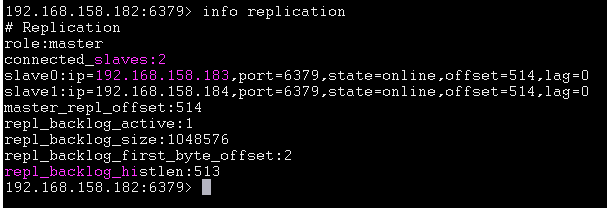
Vim /etc/redis.conf

Systemctl restart redis

Test xem redis-server hoạt động chưa

Redis-cli ping > PONG





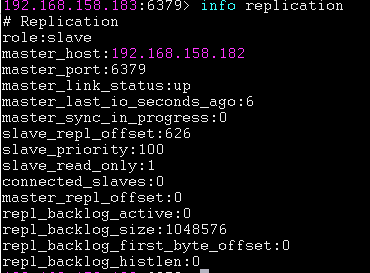
ở node slave:

vim /etc/redis.conf

bind 192.168.158.183/184

slaveof master port

systemctl restart redis



1. Cấu hình sentinel

5.1. Master

Vim ./sentinel.conf

sentinel monitor <master-group-name> <ip> <port> <quorum>

master group name : tên

ip: địa chỉ master

quorum: số lượng sentinel tối thiếu tham gia cho việc bầu chọn.

sentinel down-after-milliseconds <name-master> <milisecond>

second: sau thời gian này nếu master không phản hồi thì master chính thức chết.

1. Persistence

Redis cung cấp nhiều lựa chọn persistence:

* RDB thực thi snapshot từng điểm trong 1 khoảng thời gian
* AOF