Giới thiệu về Redis

# Giới thiệu

Nếu bạn đã từng làm việc với Object của Javascript, .Net hay đơn giản bạn đã từng đọc một file JSON, thì chắc hẳn bạn đã biết đến cấu trúc dạng key-value, thì Redis cũng tương tự như vậy.

* **Redis** là một cơ sở dữ liệu nguồn mở lưu trữ dữ liệu theo dạng key-value.
* **Redis** cũng thường được gọi là server lưu trữ các cấu trúc dữ liệu (data structure server) do các khóa có thể bao gồm các dữ liệu kiểu strings, hashes, lists, sets và sorted sets.
* **Redis** được lưu trữ key-value trên RAM với hiệu năng cao, redis còn hỗ trợ lưu trữ dữ liệu trên đĩa cứng (persistent redis) cho phép phục hồi dữ liệu khi gặp sự cố

# Những điều làm nên chuyện của Redis

Đối với mỗi một ngôn ngữ nào đó, nó đều có một số đặc điểm giúp nó trở nên phổ biến, Redis cũng như vậy.

* Redis hỗ trợ thêm mới, cập nhật và loại bỏ dữ liệu nhanh chóng
* Redis có những đặc điểm giống như Memcached như:
* Lưu trữ dạng key /value.
* Tất cả data được lưu trên Memory(RAM)
* Key có thể hết hạn(expire) hoặc không
* Nhanh(Fast), nhẹ nhàng(light-weight)
* Redis có thêm nhiều đặc điểm, chức năng khác mang lại lợi ích khi sử dụng và triển khai
  + Persistence
  + Hỗ trợ nhiều Databases
  + Truy vấn theo Key
  + Hỗ trợ counters dữ liệu kiểu integer
  + Cấu trúc dữ liệu cấp cao
  + Thao tác dữ liệu chuyên biệt
  + Tự động phân trang danh sách
  + Nhân rộng master-slave
* Redis lấy và nạp dữ liệu trên Memory(RAM), nhưng tại một thời điểm thì dữ liệu có thể được lưu trữ trên disk(Data in memory, but saved on disk).
* Điểm khác biệt dễ nhận thấy của Redis là: Key là một string nhưng value thì không giới hạn ở một string mà có thể là List, Sets, Sorted sets, ....
* Redis hỗ trợ “Multiple database” với nhiều commands để tự động remove key từ một database tới database khác.
* Mặc định thì DB 0 sẽ được lựa chọn cho mỗi lần kết nối(connection), nhưng khi sử dụng lệnh SELECT(SELECT command) thì nó có thể select/create một database khác. Thao tác MOVE(MOVE operation) có thể chuyển một item từ một DB tới DB khác một cách tự động.
* Redis rất nhanh trong các thao tác lấy và nạp dữ liệu do redis hỗ trợ nhiều lệnh mang tính chất chuyên biệt.
* Redis hỗ trợ mở rộng master-slave nếu chúng ta muốn sự an toàn hoặc mở rộng, co giãn trong việc lưu trữ data

# Đặc trưng của Redis

Chúng ta thử tìm hiểu xem Redis khác với những ngôn ngữ khác như MySQL, Postgres, SQLServer như nào nhé.

## Redis không có table mà chỉ có Data model

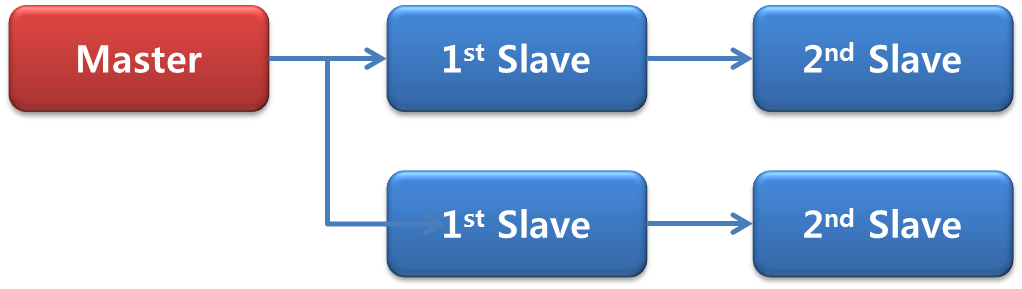
Điểm đặc biệt nhất của Redis đó chính là không có table, vì chúng ta đã nói ở phần trước Redis lưu trữ data dưới dạng key-value. Thực thế thì memcache cũng làm vậy, có thể dễ hiểu hơn Memcached là một hệ thống lưu trữ bản sao các đối tượng (objects) và dữ liệu được truy cập nhiều lần để tăng tốc độc truy xuất. Mục đích chính của nó là để tăng tốc độ ứng dụng web bằng cách truy vấn cơ sở dữ liệu bộ nhớ đệm, nội dung, hoặc kết quả tính toán khác.

Tuy nhiên kiểu dữ liệu của memcache lại bị hạn chế, không đa dạng như Redis, do đó không hỗ trợ được nhiều thao tác từ phía client, bạn có thể tham khảo các kiểu dữ liệu của Redis để lưu value:

* **STRING**: Có thể là string, integer hoặc float. Redis có thể làm việc với cả string, từng phần của string, cũng như tăng/giảm giá trị của integer, float.
* **LIST**: Danh sách liên kết của các strings. Redis hỗ trợ các thao tác push, pop từ cả 2 phía của list, trim dựa theo offset, đọc 1 hoặc nhiều items của list, tìm kiếm và xóa giá trị.
* **SET**: Tập hợp các string (không được sắp xếp). Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, kiểm tra sự xuất hiện của phần tử trong tập hợp. Ngoài ra Redis còn hỗ trợ các phép toán tập hợp, gồm intersect/union/difference.
* **HASH**: Lưu trữ hash table của các cặp key-value, trong đó key được sắp xếp ngẫu nhiên, không theo thứ tự nào cả. Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, cũng như đọc tất cả giá trị.
* **ZSET** **(sorted set):** Là 1 danh sách, trong đó mỗi phần tử là map của 1 string (member) và 1 floating-point number (score), danh sách được sắp xếp theo score này. Redis hỗ trợ thao tác thêm, đọc, xóa từng phần tử, lấy ra các phần tử dựa theo range của score hoặc của string.

## Master/Slave Replication

Redis hỗ trợ tính năng replication(master-slave) cho phép bạn có thể sao chép, đồng bộ giữa 2 CSDL Redis với nhau.



Trên website của Redis cũng đã đăng tải một số nhắc nhở khi thực hiện tính năng này: <https://redis.io/topics/replication>

* Redis sửdụng cơ chế không đồng bộ (asynchronous) để sao chép dữ liệu từ master sang các server redis slave.
* Một Redis master có thể có nhiều hơn một Redis Slave
* Redis Master vẫn tiếp tục xử lí các yêu cầu trong khi các Redis Slave tiến hành các công đoạn của việc đồng bộ dữ liệu mà không bị ảnh hưởng gì. Redis Master là non-blocking.
* Redis Slave cũng là non-blocking, trong qúa trình thực hiện việc đồng bộ dữ liệu, Redis Slave cũng có thể xử lí đồng thời các truy vấn nhưng chỉ với các dữ liệu cũ (dữ liệu trước khi đồng bộ). Hoặc bạn có thể cấu hình trong file redis.conf trả về thông báo lỗi khi client cố gắng truy vấn vào redis slave trong khi nó đang thực hiện đồng bộ với Redis Master (nói cách khác Slave Redis lúc này sẽ chặn mọi kết nối của client tới trong khi đang tiến hành đồng bộ) Sau qúa trình đồng bộ, dữ liệu cũ bị xóa, dữ liệu mới được thêm vào.
* Nhiều Redis slave được cấu hình trạng thái chỉ đọc (Read-only)
* Để gỉam thiểu qúa trình ghi xuống đĩa cứng của Redis Master (dữ liệu lúc này chỉ được lưu trữ trên RAM của Redis Master mà sẽ không được lưu trữ định kì xuống ổ cứng), bằng việc comment dòng cấu hình save trong tập tin cấu hình redis.conf (tắt chức năng Redis persistence). Và các Redis Slave lúc này, khi đồng bộ dữ liệu sẽ thực hiện việc save dữ liệu thay cho Master. Việc này rất nguy hiểm khi Redis Master gặp sự cố - Nguy cơ mất dữ liệu. Bởi vì: Khi chức năng Redis Persistence bị tắt trên Redis Master, có nghĩa dữ liệu sẽ lưu trữ trên memory không được lưu xuống đĩa cứng, khi Redis Master gặp sự cố (bị khởi động lại), mọi dữ liệu đã lưu trong memory bị mất, và cơ chế đồng bộ hóa sẽ thực hiện và đồng bộ dữ liệu rỗng sang các slave. Và hiển nhiên master, slave sẽ mất hết dữ liệu.

## Clustering

Giúp sao lưu Master – Master một cách nhanh chóng.

## In-memory

Redis được lưu trữ dữ liệu trên RAM, cho nên thao tác đọc/ghi sẽ được thực hiện trên RAM. Và tất nhiên thao tác trên RAM nhanh hơn nhiều so với trên ổ cứng. Tuy nhiên cũng sẽ có nhiều rủi ro khi server bị tắt cả dữ liệu trên RAM bị mất.

# Redis Persistence

Như chúng ta đã biết, mặc dù làm việc với data dạng key-value lưu trữ trên RAM, Redis vẫn cần lưu trữ dữ liệu trên ổ cứng. Có 2 lý do cho việc này, 1 là để đảm bảo toàn vẹn dữ liệu khi có sự cố xảy ra (server bị tắt nguồn) cũng như tái tạo lại dataset khi restart server, 2 là để gửi data đến các slave server, phục vụ cho tính năng replication. Redis cung cấp 2 phương thức chính cho việc sao lưu dữ liệu ra ổ cứng, đó là RDB và AOF.

## RDB (Redis DataBase file)

### Cách thức làm việc

* RDB thực hiện tạo và sao lưu snapshot của DB vào ổ cứng sau mỗi khoảng thời gian nhất định.

### Ưu điểm

* RDB cho phép người dùng lưu các version khác nhau của DB, rất thuận tiện khi có sự cố xảy ra.
* Bằng việc lưu trữ data vào 1 file cố định, người dùng có thể dễ dàng chuyển data đến các data centers, máy chủ khác nhau.
* RDB giúp tối ưu hóa hiệu năng của Redis. Tiến trình Redis chính sẽ chỉ làm các công việc trên RAM, bao gồm các thao tác cơ bản được yêu cầu từ phía client như thêm/đọc/xóa, trong khi đó 1 tiến trình con sẽ đảm nhiệm các thao tác disk I/O. Cách tổ chức này giúp tối đa hiệu năng của Redis.
* Khi restart server, dùng RDB làm việc với lượng data lớn sẽ có tốc độ cao hơn là dùng AOF.

### Nhược điểm

* RDB không phải là lựa chọn tốt nếu bạn muốn giảm thiểu tối đa nguy cơ mất mát dữ liệu. Thông thường người dùng sẽ set up để tạo RDB snapshot 5 phút 1 lần (hoặc nhiều hơn). Do vậy, trong trường hợp có sự cố, Redis không thể hoạt động, dữ liệu trong những phút cuối sẽ bị mất.
* RDB cần dùng fork() để tạo tiến trình con phục vụ cho thao tác disk I/O. Trong trường hợp dữ liệu quá lớn, quá trình fork() có thể tốn thời gian và server sẽ không thể đáp ứng được request từ client trong vài milisecond hoặc thậm chí là 1 second tùy thuộc vào lượng data và hiệu năng CPU.

# AOF (Append Only File)

### Cách thức làm việc

* AOF lưu lại tất cả các thao tác write mà server nhận được, các thao tác này sẽ được chạy lại khi restart server hoặc tái thiết lập dataset ban đầu.

### Ưu điểm

* Sử dụng AOF sẽ giúp đảm bảo dataset được bền vững hơn so với dùng RDB. Người dùng có thể config để Redis ghi log theo từng câu query hoặc mỗi giây 1 lần.
* Redis ghi log AOF theo kiểu thêm vào cuối file sẵn có, do đó tiến trình seek trên file có sẵn là không cần thiết. Ngoài ra, kể cả khi chỉ 1 nửa câu lệnh được ghi trong file log (có thể do ổ đĩa bị full), Redis vẫn có cơ chế quản lý và sửa chữa lối đó (redis-check-aof).
* Redis cung cấp tiến trình chạy nền, cho phép ghi lại file AOF khi dung lượng file quá lớn. Trong khi server vẫn thực hiện thao tác trên file cũ, 1 file hoàn toàn mới được tạo ra với số lượng tối thiểu operation phục vụ cho việc tạo dataset hiện tại. Và 1 khi file mới được ghi xong, Redis sẽ chuyển sang thực hiện thao tác ghi log trên file mới.

### Nhược điểm

* File AOF thường lớn hơn file RDB với cùng 1 dataset.
* AOF có thể chậm hơn RDB tùy theo cách thức thiết lập khoảng thời gian cho việc sao lưu vào ổ cứng. Tuy nhiên, nếu thiết lập log 1 giây 1 lần có thể đạt hiệu năng tương đương với RDB.
* Developer của Redis đã từng gặp phải bug với AOF (mặc dù là rất hiếm), đó là lỗi AOF không thể tái tạo lại chính xác dataset khi restart Redis. Lỗi này chưa gặp phải khi làm việc với RDB bao giờ.