Redis Advanced

Kiến thức nâng cao về Redis

- Redis backup - Tạo bản backup trong Redis
- Redis Security - Bảo mật trong Redis database
- Redis Benchmarks - Đo hiệu năng của Redis
- Redis Client Connection - Quản lý kết nối từ Client
- Redis Pipelining - Thực thi nhiều lệnh đồng thời từ Client
- Redis Partitioning

Redis backup - Tạo bản backup trong Redis

Backup Redis data

Sử dụng Redis SAVE command để thực hiện tạo một bản backup cho Redis

Cú pháp:

```
127.0.0.1:6379> SAVE
OK
```

Thực thi command này sẽ tạo 1 file dump.rdb trong thư mục Redis.

Restore Redis Data

Để thực hiện khôi phục Redis data, move file Redis backup (dump.rdb) vào thư mục Redis và start Redis server.

Để lấy đường dẫn thư mục Redis, sử dụng command CONFIG:

```
127.0.0.1:6379> CONFIG get dir
1) "dir"
2) "/user/laptrinhvn/redis-2.8.13/src"
```

Bgsave

Một cách khác để tạo một bản backup của Redis, đó là sử dụng command BGSAVE. Đây là command sẽ start tiến trình backup và được chạy background.

Ví dụ:

```
127.0.0.1:6379> BGSAVE
Background saving started
```

Redis Security - Bảo mật trong Redis database

Redis database có cơ chế bảo mật xác thực khi client kết nối và thực thi lệnh trên Redis. Để bật tính năng bảo mật, bạn cần thiết lập mật khẩu trong config file

Kiểm tra cấu hình xác thực:

```
127.0.0.1:6379> CONFIG get requirepass
1) "requirepass"
2) ""
```

Thiết lập xác thực:

```
127.0.0.1:6379> CONFIG set requirepass "tutorialspoint"
OK
127.0.0.1:6379> CONFIG get requirepass
1) "requirepass"
2) "matkhau"
```

Sau khi thiết lập mật khẩu, client kết nối đến Redis mà không xác thực sẽ nhận được thông báo lỗi: `(error) NOAUTH Authentication required`

Cú pháp xác thực:

```
127.0.0.1:6379> AUTH password
```

Ví dụ:

```
127.0.0.1:6379> AUTH "tutorialspoint"
OK
127.0.0.1:6379> SET mykey "Test value"
OK
127.0.0.1:6379> GET mykey
"Test value"
```

Redis Benchmarks - Đo hiệu năng của Redis

Redis benchmark là công cụ để thực hiện đo tốc độ xử lý đồng thời của Redis database.

Cú pháp:

```
redis-benchmark [option] [option value]
```

Ví dụ: Thực hiện đồng thời 100000 lệnh

```
redis-benchmark -n 100000

PING_INLINE: 141043.72 requests per second
PING_BULK: 142857.14 requests per second
SET: 141442.72 requests per second
GET: 145348.83 requests per second
INCR: 137362.64 requests per second
LPUSH: 145348.83 requests per second
LPOP: 146198.83 requests per second
SADD: 146198.83 requests per second
SPOP: 149253.73 requests per second
LPUSH (needed to benchmark LRANGE): 148588.42 requests per second
LRANGE_100 (first 100 elements): 58411.21 requests per second
LRANGE_300 (first 300 elements): 21195.42 requests per second
LRANGE_500 (first 450 elements): 14539.11 requests per second
LRANGE_600 (first 600 elements): 10504.20 requests per second
MSET (10 keys): 93283.58 requests per second
```

Các tham số:

Sr.No	Option	Description	Default Value
1	-h	Specifies server host name	127.0.0.1
2	-p	Specifies server port	6379
3	-s	Specifies server socket	

4	-c	Specifies the number of parallel connections	50
5	-n	Specifies the total number of requests	10000
6	-d	Specifies data size of SET/GET value in bytes	2
7	-k	1=keep alive, 0=reconnect	1
8	-r	Use random keys for SET/GET/INCR, random values for SADD	
9	-p	Pipeline <numreq> requests	1
10	-h	Specifies server host name	
11	-q	Forces Quiet to Redis. Just shows query/sec values	
12	--csv	Output in CSV format	
13	-l	Generates loop, Run the tests forever	
14	-t	Only runs the comma-separated list of tests	
15	-I	Idle mode. Just opens N idle connections and wait	

Ví dụ:

```
redis-benchmark -h 127.0.0.1 -p 6379 -t set,lpush -n 100000 -q

SET: 146198.83 requests per second
LPUSH: 145560.41 requests per second
```

Redis Client Connection - Quản lý kết nối từ Client

Redis chấp nhận các kết nối từ client theo giao thức TCP socket. Khi kết nối từ client mới được chấp nhận, các thao tác sau được thực hiện:

- Client socket được đặt ở trạng thái non-blocking do Redis sử dụng ghép kênh (multiplexing) và non-blocking I/O
- TCP_NODELAY option được thiết lập để chắc chắn rằng không có delay trong connection
- Một sự kiện được tạo để Redis có thể thu thập truy vấn của client ngay khi có dữ liệu mới trên socket.

Maximum Number of Clients

Để kiểm tra số kết nối tối đa từ client, thực hiện lệnh sau:

```
config get maxclients


1) "maxclients"
2) "10000"
```

Mặc định, có 10000 kết nối (phụ thuộc vào giới hạn trong thiết lập OS), bạn có thể thay đổi thiết lập như sau:

```
redis-server --maxclients 100000
```

Client Commands

STT	Command	Mô tả
1	**CLIENT LIST**	Trả về danh sách client đang kết nối đến Redis server
2	**CLIENT SETNAME**	Thiết lập tên của connection hiện tại
3	**CLIENT GETNAME**	Trả về tên của connection hiện tại

4	**CLIENT PAUSE**	Dừng (suspend) các kết nối theo khoảng thời gian xác định (in milliseconds)
5	**CLIENT KILL**	Đóng (close) một kết nối

Redis Pipelining - Thực thi nhiều lệnh đồng thời từ Client

Redis Pipelining được sử dụng khi bạn cần gửi nhiều câu lệnh tới server trong một yêu cầu.

Ví dụ:

```
$(echo -en "PING\r\n SET tutorial redis\r\nGET tutorial\r\nINCR
visitor\r\nINCR visitor\r\nINCR visitor\r\n"; sleep 10) | nc localhost 6379
+PONG
+OK
redis
:1
:2
:3
```

Trong ví dụ trên, chúng ta thực hiện:

- Kiểm tra Redis connection bằng cách sử dụng PING command
- Set một key với name `tutorial` có giá trị `redis`
- Get giá trị của key `tutorial`
- Tăng giá trị của key `tutorial`

Redis thực hiện và output tất cả kết quả của command.

Lợi ích của Pipelining

Lợi ích của kỹ thuật này là hiệu suất giao thức được cải thiện mạnh mẽ. Việc tăng tốc đạt được bằng pipelining dao động từ 5 kết nối localhost (kết nối nhanh) cho đến ít nhất 100 kết nối internet (chậm hơn).

Redis Partitioning

Partitioning là gì

Partitioning là quá trình phân chia dữ liệu thành nhiều phân vùng riêng biệt (instance) trên Redis, do đó mọi phân vùng sẽ chỉ chứa một tập hợp con các key của bạn.

Lợi ích của việc Partitioning

- Nó cho phép Redis database lưu trữ lớn hơn nhiều bằng cách sử dụng tổng bộ nhớ của nhiều máy tính. Nếu không partitioning, bạn sẽ bị giới hạn số lượng bộ nhớ mà một máy tính có thể hỗ trợ.
- Nó cho phép mở rộng sức mạnh tính toán đa luồng (multiple cores), đa máy tính (multiple computers), và băng thông mạng cho nhiều máy tính và network.

Nhược điểm của Partitioning

- Các hoạt động liên quan đến nhiều khóa (multiple keys) thường không được hỗ trợ. Chẳng hạn, bạn không thể thực hiệnthi trên hai tập (sets) nếu chúng được lưu trữ trong các khóa được ánh xạ tới các Redis instance khác nhau.
- Giao dịch Redis liên quan đến nhiều khóa (multiple keys) không thể được sử dụng.
- Partitioning đặc biệt là khóa, vì vậy không thể loại bỏ một tập dữ liệu với một khóa lớn như một tập hợp được sắp xếp rất lớn.
- Khi Partitioning được sử dụng, xử lý dữ liệu phức tạp hơn. Chẳng hạn, bạn phải xử lý nhiều tệp RDB / AOF và để có bản sao lưu dữ liệu của mình, bạn cần tổng hợp các tệp lưu trữ từ nhiều phiên bản và máy chủ lưu trữ.
- Thêm và loại bỏ capacity có thể phức tạp. Chẳng hạn, Redis Cluster hỗ trợ cân bằng lại dữ liệu trong suốt với khả năng thêm và xóa các node khi chạy. Tuy nhiên, các hệ thống khác như Partitioning phía máy khách và proxy không hỗ trợ tính năng này. Một kỹ thuật được gọi là `Preshending` giúp về vấn đề này.

Các loại Partitioning

Có hai loại Partitioning có sẵn trong Redis. Giả sử chúng ta có bốn phiên bản Redis, R0, R1, R2, R3 và nhiều `key` đại diện cho người dùng như: user:1, user:2....

Range Partitioning

Range Partitioning được thực hiện bằng cách ánh xạ phạm vi của các đối tượng vào các Redis instance cụ thể. Giả sử trong ví dụ của chúng tôi, người dùng có từ ID.0 đến ID.10000 sẽ đi vào R0 instance, trong khi người dùng từ ID.10001 đến ID.20000 sẽ vào R1 instance...

Hash Partitioning

Trong kiểu phân vùng này, một hàm băm - hash (ví dụ: hàm mô đun) được sử dụng để chuyển đổi key thành số và sau đó dữ liệu được lưu trữ trong các Redis instance khác nhau.