

Redis

Redis是一個使用ANSI C編寫的開源、支援網路、基於記憶體、可選永續性的鍵值對儲存資料庫。Redis是最流行的鍵值對儲存資料庫。

支援語言

Python簡單範例

```
# coding:utf-8
import redis

# lredis-server保持开启状态 如果在客户端设置了密码 添加password=密码即可
pool = redis.ConnectionPool(host='127.0.0.1', port=6379, db=0)
r = redis.StrictRedis(connection_pool=pool)

# 字符串
r.set('test', 'aaa')
print r.get('test')

# 列表
# 注意python、lrange两个range的范围
x = 0
for x in range(0, 11):
    r.lpush('list', x)
    x = x + 1
print r.lrange('list', '0', '10')

# 雜湊(哈希)
dict_hash = {'name': 'tang', 'password': 'tang_passwd'}
r.hmset('hash_test', dict_hash)
print r.hgetall('hash_test')

# 集合
r.sadd('set_test', 'aaa', 'bbb')
r.sadd('set_test', 'ccc')
r.sadd('set_test', 'ddd')
print r.smembers('set_test')

# 有序集
r.zadd('zset_test', {'aaa': 1, 'bbb': 1})
r.zadd('zset_test', {'ccc': 1})
r.zadd('zset_test', {'ddd': 1})
print r.zrange('zset_test', 0, 10)
```

資料模型

Redis的外圍由一個鍵、值對映的字典構成。與其他非關係型資料庫主要不同在於：

Redis中值的類型不僅限於字串，還支援如下抽象資料類型：

- 字串列表
- 無序不重複的字串集合
- 有序不重複的字串集合
- 鍵、值都為字串的雜湊表

值的類型決定了值本身支援的操作。

Redis支援不同無序、有序的列表，無序、有序的集合間的交集、並集等進階伺服器端原子操作。

持久化

Redis通常將全部的資料儲存在記憶體中。

2.4版本後可組態為使用虛擬記憶體，一部分資料集儲存在硬碟上，但這個特性廢棄了。

目前通過兩種方式實現持久化：

- 使用快照，一種半持久耐用模式。不時的將資料集以非同步方式從記憶體以RDB格式寫入硬碟。
- 1.1版本開始使用更安全的AOF格式替代，一種只能追加的紀錄檔類型。將資料集修改操作記錄起來。Redis能夠在後台對只可追加的記錄作修改來避免無限增長的紀錄檔。

同步

Redis支援主從同步。資料可以從主伺服器向任意數量的從伺服器上同步，

從伺服器可以是關聯其他從伺服器的主伺服器。

這使得Redis可執行單層樹複製。從盤可以有意無意的對資料進行寫操作。

由於完全實現了發布/訂閱機制，使得從資料庫在任何地方同步樹時，

可訂閱一個頻道並接收主伺服器完整的訊息發布記錄。同步對讀取操作的可延伸性和資料冗餘很有幫助。

效能

當資料依賴不再需要，Redis這種基於記憶體的性質，

與在執行一個事務時將每個變化都寫入硬碟的資料庫系統相比就顯得執行效率非常高。

寫與讀操作速度沒有明顯差別。

安裝 redis (以 Ubuntu 18.04 為例)

```
$ sudo apt-get install redis-server  
$ sudo systemctl status redis (檢測 redis 服務狀態)  
$ redis-cli  
127.0.0.1:6379> PING (檢測 redis 服務是否啟動)  
PONG
```

```
127.0.0.1:6379> CONFIG GET * (使用 * 號獲取所有配置項)
1) "dbfilename"
2) "dump.rdb"
3) "requirepass"
4) ""
5) "masterauth"
6) ""
7) "cluster-announce-ip"
8) ""
9) "unixsocket"
...
127.0.0.1:6379> EXIT
```

Redis 數據類型

- string (字串)

string 是 redis 最基本的類型，你可以理解成與 Memcached 一模一樣的類型，一個 key 對應一個 value。

```
$ redis-cli
127.0.0.1:6379> SET name walter
OK
127.0.0.1:6379> GET name
"walter"
127.0.0.1:6379> DEL name
(integer) 1
127.0.0.1:6379> GET name
(nil)
```

- hash (雜湊(哈希))

Redis hash 是一個鍵值(key=>value)對集合。Redis hash 是一個 string 類型的 field 和 value 的映射表，hash 特別適合用於存儲對象。

```
$ redis-cli
127.0.0.1:6379> HMSET user id 1 name walter
OK
127.0.0.1:6379> HMGET user id
1) "1"
127.0.0.1:6379> HMGET user name
1) "walter"
127.0.0.1:6379> DEL user
(integer) 1
127.0.0.1:6379> HMGET user id
1) (nil)
```

- list (列表)

Redis 列表是簡單的字符串列表，按照插入順序排序。你可以添加一個元素到列表的頭部（左邊）或者尾部（右邊）。

```
$ redis-cli
127.0.0.1:6379> LPUSH id 1
(integer) 1
127.0.0.1:6379> LPUSH id 2 3
(integer) 3
127.0.0.1:6379> LRANGE id 1 3
1) "2"
2) "1"
127.0.0.1:6379> LRANGE id 0 2
1) "3"
2) "2"
3) "1"
127.0.0.1:6379> RPUSH id 4
(integer) 4
127.0.0.1:6379> LRANGE id 0 3
1) "3"
2) "2"
3) "1"
4) "4"
127.0.0.1:6379> DEL id
(integer) 1
127.0.0.1:6379> LRANGE id 0 3
(empty list or set)
```

- set (無序不重複的字串集合)

Redis的Set是string類型的無序集合。

```
$ redis-cli
127.0.0.1:6379> SADD id 1 2 3
(integer) 3
127.0.0.1:6379> SMEMBERS id
1) "1"
2) "2"
3) "3"
127.0.0.1:6379> DEL id
(integer) 1
127.0.0.1:6379> SMEMBERS id
(empty list or set)
```

- zset (有序不重複的字串集合)

Redis zset 和 set 一樣也是string類型元素的集合，且不允許重複的成員。不同的是每個元素都會關聯一個double類型的 score。

```
$ redis-cli
127.0.0.1:6379> ZADD score 10 walter
(integer) 1
127.0.0.1:6379> ZADD score 20 jack
```

```
(integer) 1
127.0.0.1:6379> ZADD score 25 tom
(integer) 1
127.0.0.1:6379>ZRANGEBYSCORE score -inf +inf
1) "walter"
2) "jack"
3) "tom"
127.0.0.1:6379>ZRANGEBYSCORE score -inf +inf withscores
1) "walter"
2) "10"
3) "jack"
4) "20"
5) "tom"
6) "25"
127.0.0.1:6379>ZADD score 20 peter
(integer) 1
127.0.0.1:6379>ZRANGEBYSCORE score -inf +inf withscores
1) "walter"
2) "10"
3) "jack"
4) "20"
5) "peter"
6) "20"
7) "tom"
8) "25"
```

Redis 命令

語法

Redis 客戶端的基本語法為：

```
$ redis-cli
127.0.0.1:6379> PING
PONG
127.0.0.1:6379>
```

在以上實例中我們連接到本地的 redis 服務並執行 PING 命令，該命令用於檢測 redis 服務是否啟動。

在遠程服務上執行命令

如果需要在遠程 redis 服務上執行命令，同樣我們使用的也是 redis-cli 命令。

```
$ redis-cli -h 127.0.0.1 -p 6379
127.0.0.1:6379> PING
PONG
127.0.0.1:6379> INFO
# Server
redis_version:4.0.9
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:9435c3c2879311f3
redis_mode:standalone
```

```
os:Linux 5.0.0-27-generic x86_64
arch_bits:64
multiplexing_api:epoll
... 以下略過
```

Redis 安全

我們可以通過 redis 的配置文件設置密碼參數，這樣客戶端連接到 redis 服務就需要密碼驗證，這樣可以讓你的 redis 服務更安全。

默認情況下 requirepass 參數是空的，這就意味著你無需通過密碼驗證就可以連接到 redis 服務。

你可以通過以下命令來修改該參數：

```
$ redis-cli
127.0.0.1:6379> CONFIG get requirepass
1) "requirepass"
2) ""
```

設置密碼後，客戶端連接 redis 服務就需要密碼驗證，否則無法執行命令。

```
$
127.0.0.1:6379> CONFIG set requirepass "mypassword"
OK
127.0.0.1:6379> CONFIG get requirepass
(error) NOAUTH Authentication required.
127.0.0.1:6379> AUTH "mypassword"
OK
127.0.0.1:6379> CONFIG get requirepass
1) "requirepass"
2) "mypassword"
```

Redis 性能測試

- Redis 性能測試是通過同時執行多個命令實現的。
- Redis 性能測試的基本命令如下：

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

實例

測試環境如下：

測試環境如下：

```
$ OS: Ubuntu 18.04 bionic Kernel: x86_64 Linux 5.0.0-27-generic Shell: fish 2.7.1 CPU: Intel Core i5-8265U @ 8x 3.9GHz [53.0°C] GPU: Mesa DRI Intel(R) HD Graphics (Whiskey Lake 3x8 GT2) RAM: 4088MiB / 11857MiB
```

```
$  
OS: Ubuntu 18.04 bionic  
Kernel: x86_64 Linux 5.0.0-27-generic  
Shell: fish 2.7.1  
CPU: Intel Core i5-8265U @ 8x 3.9GHz [53.0°C]  
GPU: Mesa DRI Intel(R) HD Graphics (Whiskey Lake 3x8 GT2)  
RAM: 4088MiB / 11857MiB
```

以下實例同時執行 10000 個請求來檢測性能：

```
$ redis-benchmark -n 10000 -q  
PING_INLINE: 89285.71 requests per second  
PING_BULK: 151515.16 requests per second  
SET: 151515.16 requests per second  
GET: 149253.73 requests per second  
INCR: 147058.81 requests per second  
LPUSH: 151515.16 requests per second  
RPUSH: 131578.95 requests per second  
LPOP: 131578.95 requests per second  
RPOP: 149253.73 requests per second  
SADD: 147058.81 requests per second  
HSET: 144927.55 requests per second  
SPOP: 149253.73 requests per second  
LPUSH (needed to benchmark LRANGE): 144927.55 requests per second  
LRANGE_100 (first 100 elements): 140845.06 requests per second  
LRANGE_300 (first 300 elements): 147058.81 requests per second  
LRANGE_500 (first 450 elements): 147058.81 requests per second  
LRANGE_600 (first 600 elements): 149253.73 requests per second  
MSET (10 keys): 125000.00 requests per second
```

以下實例我們使用了多個參數來測試 redis 性能：

```
$ edis-benchmark -h 127.0.0.1 -p 6379 -t set,lpush -n 10000 -q  
SET: 113636.37 requests per second  
LPUSH: 153846.16 requests per second
```