# 1.介绍

## 1.1 Redis是什么

**RE**mote **DI**ctionary **S**erver(Redis) 是一个由Salvatore Sanfilippo写的key-value存储系统。Redis提供了一些丰富的数据结构，包括strings、lists, sets, ordered sets（zset） 以及hashes ，Redis包括了对这些数据结构的丰富操作。

Redis官方网站：http://redis.io

## 1.2 Redis的优点

* 性能极高 – Redis能支持超过 100K+ 每秒的读写频率。

根据官方测试，在50并发情况下请求10W次，写速度为110000次/s，读速度为81000次/s。

测试环境：

1、50并发，请求100000

2、读和写大小为256bytes的字符

3、Linux2.6 Xeon X3320 2.5GHz的服务器

* 丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型，对各中数据类型均提供了丰富的操作。
* 原子 – Redis的所有操作都是原子性的，同时Redis还支持对几个操作全并后的原子性执行。
* 丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

# 2.数据类型

## 2.1 String类型

Redis能存储二进制安全的字符串，最大长度为1GB

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> GET name

"John Doe"

String类型还支持批量的读写操作

redis 127.0.0.1:6379> MSET age 30 sex "male"

OK

redis 127.0.0.1:6379> MGET age sex

1) "30"

2) "male"

String类型其实也可以用来存储数字，并支持对数字的加减操作。

redis 127.0.0.1:6379> INCR age

(integer) 31

redis 127.0.0.1:6379> INCRBY age 4

(integer) 35

redis 127.0.0.1:6379> GET age

"35"

redis 127.0.0.1:6379> DECR age

(integer) 34

redis 127.0.0.1:6379> DECRBY age 4

(integer) 30

redis 127.0.0.1:6379> GET age

"30"

String类型还支持对其部分的修改和获取操作

redis 127.0.0.1:6379> APPEND name " Mr."

(integer) 12

redis 127.0.0.1:6379> GET name

"John Doe Mr."

redis 127.0.0.1:6379> STRLEN name

(integer) 12

redis 127.0.0.1:6379> SUBSTR name 0 3

"John"

## 2.2 List类型

Redis能够将数据存储成一个链表，并能对这个链表进行丰富的操作

redis 127.0.0.1:6379> LPUSH students "John Doe"

(integer) 1

redis 127.0.0.1:6379> LPUSH students "Captain Kirk"

(integer) 2

redis 127.0.0.1:6379> LPUSH students "Sheldon Cooper"

(integer) 3

redis 127.0.0.1:6379> LLEN students

(integer) 3

redis 127.0.0.1:6379> LRANGE students 0 2

1) "Sheldon Cooper"

2) "Captain Kirk"

3) "John Doe"

redis 127.0.0.1:6379> LPOP students

"Sheldon Cooper"

redis 127.0.0.1:6379> LLEN students

(integer) 2

redis 127.0.0.1:6379> LRANGE students 0 1

1) "Captain Kirk"

2) "John Doe"

redis 127.0.0.1:6379> LREM students 1 "John Doe"

(integer) 1

redis 127.0.0.1:6379> LLEN students

(integer) 1

redis 127.0.0.1:6379> LRANGE students 0 0

1) "Captain Kirk"

Redis也支持很多修改操作

redis 127.0.0.1:6379> LINSERT students BEFORE "Captain Kirk" "Dexter Morgan"

(integer) 3

redis 127.0.0.1:6379> LRANGE students 0 2

1) "Dexter Morgan"

2) "Captain Kirk"

3) "John Doe"

redis 127.0.0.1:6379> LPUSH students "Peter Parker"

(integer) 4

redis 127.0.0.1:6379> LRANGE students 0 3

1) "Peter Parker"

2) "Dexter Morgan"

3) "Captain Kirk"

4) "John Doe"

redis 127.0.0.1:6379> LTRIM students 1 3

OK

redis 127.0.0.1:6379> LLEN students

(integer) 3

redis 127.0.0.1:6379> LRANGE students 0 2

1) "Dexter Morgan"

2) "Captain Kirk"

3) "John Doe"

redis 127.0.0.1:6379> LREM students 1 "John Doe"

(integer) 1

redis 127.0.0.1:6379> LLEN students

(integer) 1

redis 127.0.0.1:6379> LRANGE students 0 1

1) "Captain Kirk"

## 2.3 集合（Sets）类型

Redis能够将一系列不重复的值存储成一个集合

redis 127.0.0.1:6379> SADD birds crow

(integer) 1

redis 127.0.0.1:6379> SADD birds pigeon

(integer) 1

redis 127.0.0.1:6379> SADD birds bat

(integer) 1

redis 127.0.0.1:6379> SADD mammals dog

(integer) 1

redis 127.0.0.1:6379> SADD mammals cat

(integer) 1

redis 127.0.0.1:6379> SADD mammals bat

(integer) 1

redis 127.0.0.1:6379> SMEMBERS birds

1) "bat"

2) "crow"

3) "pigeon"

redis 127.0.0.1:6379> SMEMBERS mammals

1) "bat"

2) "cat"

3) "dog"

Sets结构也支持相应的修改操作

redis 127.0.0.1:6379> SREM mammals cat

(integer) 1

redis 127.0.0.1:6379> SMEMBERS mammals

1) "bat"

2) "dog"

redis 127.0.0.1:6379> SADD mammals human

(integer) 1

redis 127.0.0.1:6379> SMEMBERS mammals

1) "bat"

2) "human"

3) "dog"

Redis还支持对集合的子交并补等操作

redis 127.0.0.1:6379> SINTER birds mammals

1) "bat"

redis 127.0.0.1:6379> SUNION birds mammals

1) "crow"

2) "bat"

3) "human"

4) "pigeon"

5) "dog"

redis 127.0.0.1:6379> SDIFF birds mammals

1) "crow"

2) "pigeon"

## 2.4 有序集合（Sorted Sets）类型

Sorted Sets和Sets结构相似，不同的是存在Sorted Sets中的数据会有一个score属性，并会在写入时就按这个score排好序。

redis 127.0.0.1:6379> ZADD days 0 mon

(integer) 1

redis 127.0.0.1:6379> ZADD days 1 tue

(integer) 1

redis 127.0.0.1:6379> ZADD days 2 wed

(integer) 1

redis 127.0.0.1:6379> ZADD days 3 thu

(integer) 1

redis 127.0.0.1:6379> ZADD days 4 fri

(integer) 1

redis 127.0.0.1:6379> ZADD days 5 sat

(integer) 1

redis 127.0.0.1:6379> ZADD days 6 sun

(integer) 1

redis 127.0.0.1:6379> ZCARD days

(integer) 7

redis 127.0.0.1:6379> ZRANGE days 0 6

1) "mon"

2) "tue"

3) "wed"

4) "thu"

5) "fri"

6) "sat"

7) "sun"

redis 127.0.0.1:6379> ZSCORE days sat

"5"

redis 127.0.0.1:6379> ZCOUNT days 3 6

(integer) 4

redis 127.0.0.1:6379> ZRANGEBYSCORE days 3 6

1) "thu"

2) "fri"

3) "sat"

4) "sun"

## 2.5 Hash类型

Redis能够存储key对多个属性的数据（比如user1.uname user1.passwd）

redis 127.0.0.1:6379> HKEYS student

1) "name"

2) "age"

3) "sex"

redis 127.0.0.1:6379> HVALS student

1) "Ganesh"

2) "30"

3) "Male"

redis 127.0.0.1:6379> HGETALL student

1) "name"

2) "Ganesh"

3) "age"

4) "30"

5) "sex"

6) "Male"

redis 127.0.0.1:6379> HDEL student sex

(integer) 1

redis 127.0.0.1:6379> HGETALL student

1) "name"

2) "Ganesh"

3) "age"

4) "30"

Hash数据结构能够批量修改和获取

redis 127.0.0.1:6379> HMSET kid name Akshi age 2 sex Female

OK

redis 127.0.0.1:6379> HMGET kid name age sex

1) "Akshi"

2) "2"

3) "Female"

# 3.Publish/Subscribe

Redis支持这样一种特性，你可以将数据推到某个信息管道中，然后其它人可以通过订阅这些管道来获取推送过来的信息。

## 3.1 订阅信息管道

用一个客户端订阅管道

redis 127.0.0.1:6379> SUBSCRIBE channelone

Reading messages... (press Ctrl-C to quit)

1) "subscribe"

2) "channelone"

3) (integer) 1

另一个客户端往这个管道推送信息

redis 127.0.0.1:6379> PUBLISH channelone hello

(integer) 1

redis 127.0.0.1:6379> PUBLISH channelone world

(integer) 1

然后第一个客户端就能获取到推送的信息

redis 127.0.0.1:6379> SUBSCRIBE channelone

Reading messages... (press Ctrl-C to quit)

1) "subscribe"

2) "channelone"

3) (integer) 1

1) "message"

2) "channelone"

3) "hello"

1) "message"

2) "channelone"

3) "world"

## 3.2 按一定模式批量订阅

用下面的命令订阅所有channel开头的信息通道

redis 127.0.0.1:6379> PSUBSCRIBE channel\*

Reading messages... (press Ctrl-C to quit)

1) "psubscribe"

2) "channel\*"

3) (integer) 1

在另一个客户端对两个推送信息

redis 127.0.0.1:6379> PUBLISH channelone hello

(integer) 1

redis 127.0.0.1:6379> PUBLISH channeltwo world

(integer) 1

然后在第一个客户端就能收到推送的信息

redis 127.0.0.1:6379> PSUBSCRIBE channel\*

Reading messages... (press Ctrl-C to quit)

1) "psubscribe"

2) "channel\*"

3) (integer) 1

1) "pmessage"

2) "channel\*"

3) "channelone"

4) "hello"

1) "pmessage"

2) "channel\*"

3) "channeltwo"

4) "world"

# 4.数据过期设置

Redis支持按key设置过期时间，过期后值将被删除

用TTL命令可以获取某个key值的过期时间（-1表示永不过期）

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> TTL name

(integer) -1

下面命令先用EXISTS命令查看key值是否存在，然后设置了5秒的过期时间

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> EXISTS name

(integer) 1

redis 127.0.0.1:6379> EXPIRE name 5

(integer) 1

5秒后再查看

redis 127.0.0.1:6379> EXISTS name

(integer) 0

redis 127.0.0.1:6379> GET name

(nil)

这个值已经没有了。

上在是直接设置多少秒后过期，你也可以设置在某个时间点过期，下面例子是设置2011-09-24 00:40:00过期。

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> EXPIREAT name 1316805000

(integer) 1

redis 127.0.0.1:6379> EXISTS name

(integer) 0

# 5.事务性

Redis本身支持一些简单的组合型的命令，比如以NX结尾命令都是判断在这个值没有时才进行某个命令。

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> SETNX name "Dexter Morgan"

(integer) 0

redis 127.0.0.1:6379> GET name

"John Doe"

redis 127.0.0.1:6379> GETSET name "Dexter Morgan"

"John Doe"

redis 127.0.0.1:6379> GET name

"Dexter Morgan"

当然，Redis还支持自定义的命令组合，通过MULTI和EXEC，将几个命令组合起来执行

redis 127.0.0.1:6379> SET counter 0

OK

redis 127.0.0.1:6379> MULTI

OK

redis 127.0.0.1:6379> INCR counter

QUEUED

redis 127.0.0.1:6379> INCR counter

QUEUED

redis 127.0.0.1:6379> INCR counter

QUEUED

redis 127.0.0.1:6379> EXEC

1) (integer) 1

2) (integer) 2

3) (integer) 3

redis 127.0.0.1:6379> GET counter

"3"

你还可以用DICARD命令来中断执行中的命令序列

redis 127.0.0.1:6379> SET newcounter 0

OK

redis 127.0.0.1:6379> MULTI

OK

redis 127.0.0.1:6379> INCR newcounter

QUEUED

redis 127.0.0.1:6379> INCR newcounter

QUEUED

redis 127.0.0.1:6379> INCR newcounter

QUEUED

redis 127.0.0.1:6379> DISCARD

OK

redis 127.0.0.1:6379> GET newcounter

"0"

# 6.持久化

Redis的所有数据都存储在内存中，但是他也提供对这些数据的持久化。

## 6.1 数据快照

数据快照的原理是将整个Redis中存的所有数据遍历一遍存到一个扩展名为rdb的数据文件中。通过SAVE命令可以调用这个过程。

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> SAVE

OK

redis 127.0.0.1:6379> SET name "Sheldon Cooper"

OK

redis 127.0.0.1:6379> BGSAVE

Background saving started

## 6.2 Append-Only File（追加式的操作日志记录）

Redis还支持一种追加式的操作日志记录，叫append only file，其日志文件以aof结尾。要开启aof日志的记录，你需要在配置文件中进行如下设置：

appendonly yes

这时候你所有的操作都会记录在aof日志文件中

redis 127.0.0.1:6379> GET name

(nil)

redis 127.0.0.1:6379> SET name "Ganesh Gunasegaran"

OK

redis 127.0.0.1:6379> EXIT

→ cat /usr/local/var/db/redis/appendonly.aof

\*2

$6

SELECT

$1

0

\*3

$3

SET

$4

name

$18

Ganesh Gunasegaran

# 7.管理命令

Redis支持多个DB，默认是16个，你可以设置将数据存在哪一个DB中，不同DB间的数据具有隔离性。也可以在多个DB间移动数据。

redis 127.0.0.1:6379> SELECT 0

OK

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> SELECT 1

OK

redis 127.0.0.1:6379[1]> GET name

(nil)

redis 127.0.0.1:6379[1]> SELECT 0

OK

redis 127.0.0.1:6379> MOVE name 1

(integer) 1

redis 127.0.0.1:6379> SELECT 1

OK

redis 127.0.0.1:6379[1]> GET name

"John Doe"

Redis还能进行一些如下操作，获取一些运行信息

redis 127.0.0.1:6379[1]> DBSIZE

(integer) 1

redis 127.0.0.1:6379[1]> INFO

redis\_version:2.2.13

redis\_git\_sha1:00000000

redis\_git\_dirty:0

arch\_bits:64

multiplexing\_api:kqueue

......

Redis还支持对某个DB数据进行清除（当然清空所有数据的操作也是支持的）

redis 127.0.0.1:6379> SET name "John Doe"

OK

redis 127.0.0.1:6379> DBSIZE

(integer) 1

redis 127.0.0.1:6379> SELECT 1

OK

redis 127.0.0.1:6379[1]> SET name "Sheldon Cooper"

OK

redis 127.0.0.1:6379[1]> DBSIZE

(integer) 1

redis 127.0.0.1:6379[1]> SELECT 0

OK

redis 127.0.0.1:6379> FLUSHDB

OK

redis 127.0.0.1:6379> DBSIZE

(integer) 0

redis 127.0.0.1:6379> SELECT 1

OK

redis 127.0.0.1:6379[1]> DBSIZE

(integer) 1

redis 127.0.0.1:6379[1]> FLUSHALL

OK

redis 127.0.0.1:6379[1]> DBSIZE

(integer) 0

# 8、应用场景

1.取最新N个数据的操作

比如典型的取你网站的最新文章，通过下面方式，我们可以将最新的5000条评论的ID放在Redis的List集合中，并将超出集合部分从数据库获取

* 使用LPUSH latest.comments<ID>命令，向list集合中插入数据
* 插入完成后再用LTRIM latest.comments 0 5000命令使其永远只保存最近5000个ID
* 然后我们在客户端获取某一页评论时可以用下面的逻辑（伪代码）

FUNCTION get\_latest\_comments(start,num\_items):

id\_list = redis.lrange("latest.comments",start,start+num\_items-1)

IF id\_list.length < num\_items

id\_list = SQL\_DB("SELECT ... ORDER BY time LIMIT ...")

END

RETURN id\_list

END

如果你还有不同的筛选维度，比如某个分类的最新N条，那么你可以再建一个按此分类的List，只存ID的话，Redis是非常高效的。

2.排行榜应用，取TOP N操作

这个需求与上面需求的不同之处在于，前面操作以时间为权重，这个是以某个条件为权重，比如按顶的次数排序，这时候就需要我们的sorted set出马了，将你要排序的值设置成sorted set的score，将具体的数据设置成相应的value，每次只需要执行一条ZADD命令即可。

3.需要精准设定过期时间的应用

比如你可以把上面说到的sorted set的score值设置成过期时间的时间戳，那么就可以简单地通过过期时间排序，定时清除过期数据了，不仅是清除Redis中的过期数据，你完全可以把Redis里这个过期时间当成是对数据库中数据的索引，用Redis来找出哪些数据需要过期删除，然后再精准地从数据库中删除相应的记录。

4.计数器应用

Redis的命令都是原子性的，你可以轻松地利用INCR，DECR命令来构建计数器系统。

5.Uniq操作，获取某段时间所有数据排重值

这个使用Redis的set数据结构最合适了，只需要不断地将数据往set中扔就行了，set意为集合，所以会自动排重。

6.实时系统，反垃圾系统

通过上面说到的set功能，你可以知道一个终端用户是否进行了某个操作，可以找到其操作的集合并进行分析统计对比等。没有做不到，只有想不到。

7.Pub/Sub构建实时消息系统

Redis的Pub/Sub系统可以构建实时的消息系统，比如很多用Pub/Sub构建的实时聊天系统的例子。

8.构建队列系统

使用list可以构建队列系统，使用sorted set甚至可以构建有优先级的队列系统。

9.缓存

这个不必说了，性能优于Memcached，数据结构更多样化。