Redis

First released in 2009, Redis (REmote DIctionary Service) is a simple-to-use (IN-Memory) key-value store with a sophisticated set of commands.

When it comes to speed, Redis is hard to beat.

Reads are fast, and writes are even faster, handling upwards of 100,000 SET operations per second by some benchmarks.

Data Structure Server Store

It can be a bit difficult to classify exactly what Redis *is*.

At a basic level, it’s a key-value store, of course, but that simple label doesn’t really do it justice

Redis supports advanced data structures, though not to the degree that a document-oriented database would.

It supports set-based query operations but not with the granularity or type support you’d find in a relational database.

And, of course, it’s *fast*, trading durability for raw speed.

In addition to being an advanced data structure server, Redis is a blocking queue (or stack) and a publish-subscribe system.

All of this makes Redis more of a toolkit of useful data structure algorithms and processes than a member of any specific database genre.

Redis’ expansive list of client libraries makes it a drop-in option for many programming languages.

It’s not simply easy to use; it’s a joy.

CRUD and Datatypes

Since the command-line interface (CLI) is of such primary importance to the Redis development team—and loved by users everywhere—we’re going to look at many of the 124 commands available.

Of primary importance is its sophisticated datatypes and how they can query in more ways than simply “retrieve the value of this key.”

Getting Started

Redis is available through a few package – see website (<http://redis.io/>) (also at <https://github.com/rgl/redis/downloads> which I have downloaded to Redis Download folder)

(You can also build it for windows: I built Redis on Windows using Visual Studio as follows:

Cloned Redis Windows from Github (I used Git GUI)

<https://github.com/MSOpenTech/redis.git>

•Open the solution file msvs\redisserver.sln in Visual Studio 2013, select a build configuration (Debug or Release) and target (Win32 or x64) then build.

This should create the following executables in the msvs\$(Target)\$(Configuration) folder:)

Now run redis-server.exe from c:/Program Files/Redis

Machine generated alternative text:
Administrator. CAWindcwsN.system32,cmd.exe - redis-server.exe 
176161 1? 
t config. 
conf 
176161 1? 
176161 1? 
6379 
Sep Warning: no config file specif ied. using the defaul 
In order to specify a config file use redis—seruer.exe / path/ to/ redis. 
Redis 2.8 .16 (øøøøøøøøZø) 64 bit 
Sep 
Sep 
14: 
14: 
44: 
44: 
ø33 
ø33 
Running in stand alone 
mode 
16 
connections 
Port: 6379 
PID: 7616 
http://redis. 
Server started. Redis version 
10 
2.8. 
po 
The server is now ready to accept 

Next run the command- line tool, which should connect to the default port 6379 automatically.

 redis-cli.exe

Machine generated alternative text:
Administrator: - redis-cli.exe 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
78. 
131. 
97. 
25b 
løl. 
97. 
76. 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
489 
156 
684 
738 
563 
369 
64ø 
984 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
rio . obj 
script ing . obj 
sds . obj 
sentinel. obj 
shal . obj 
slowlog . obj 
sort . obj 
sparkline . obj 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
lib 
63 File(s) 
4 Dir(s) bytes free 
C: XNew folderXmsusXx64XRe 

After you connect, let’s try to ping the server.

redis 127.0.0.1:6379> PING

PONG

Machine generated alternative text:
Administrator: - redis-cli.exe 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
97. 
25b 
løl. 
97. 
76. 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
684 
738 
563 
369 
64ø 
984 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
sds . obj 
sentinel. obj 
shal . obj 
slowlog . obj 
sort . obj 
sparkline . obj 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
lib 
63 File(s) 
4 Dir(s) bytes free 
C: XNew folderXmsusXx64XRe 
ping 
PONG 

If you cannot connect, you’ll receive an error message.

Typing *help* will display a list of help options.

Type *help* followed by a space and then start typing any command.

If you don’t know any Redis commands, just start pressing Tab to cycle through your options.

redis 127.0.0.1:6379> help  
Type: "help @<group>" to get a list of commands in <group>

"help <command>" for help on <command>  
 "help <tab>" to get a list of possible help topics  
 "quit" to exit

We’re going to use Redis to build the back end for a URL shortener, like tinyurl.com or bit.ly.

A URL shortener is a service that takes a really long URL and maps it to a shorter version on their own domain—like mapping <http://www.myveryververylongdomain.com/somelongpath.php> to <http://bit.ly/VLD>.

Visiting that short URL redirects users to the longer mapped URL, saves the visitors from text messaging long strings, and also provides the short URL creator some statistics like a count of visits.

In Redis we can use SET to key a short code like 7wks to a value like <http://www.sevenweeks.org>.

SET always requires two parameters, a key and a value.

Retrieving the value just needs GET and the key name.

SET 7wks <http://www.sevenweeks.org/>  
OK

GET 7wks  
"<http://www.sevenweeks.org/>"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - redis-cli exe 
lib 
1 ?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
63 File(s) 
4 Dir(s) bytes free 
: XNew folderXmsusXx64XRe lease hedis—cli.exe 
ONG 
red is— 
12?.ø. 
12?.ø. 
"http: 
ping 
help 
cli 2.8 .16 
"help e(groupY' to get a list of commands in (group) 
"help (command)" for help on (command) 
"help (tab)" to get a list of possible help topics 
'quit" to exit 
ø.I : 6379) SET ?wks http://www.seuenweeks.org/ 
GET ?wks 
//www.seuenweeks . or / ' ' 

To reduce traffic, we can also set multiple values with MSET, like any number of key-value pairs.

Here we map Google.com to gog and Yahoo.com to yah.

MSET gog <http://www.google.com> yah <http://www.yahoo.com>   
OK

Correlatively, MGET grabs multiple keys and returns values as an ordered list.

MGET gog yah  
1) "http://www.google.com/"  
2) "http://www.yahoo.com/"

Machine generated alternative text:
Administrator: CAWindcAvsNsystem32,cmd.exe - redis-cli exe 
C: XNew folderXmsusXx64XRe 
com 
12?.ø. 
PONG 
12?.ø. 
red is— 
T ype : 
12?.ø. 
12?.ø. 
"http: 
12?.ø. 
ping 
help 
cli 2.8 .16 
"help e(groupY' to get a list of commands in (group) 
"help (command)" for help on (command) 
"help (tab)" to get a list of possible help topics 
'quit" to exit 
ø.I : 6379) SET ?wks http://www.seuenweeks.org/ 
GET ?wks 
//www.sevenweeks.org/" 
ø.I "http://www.seuenweeks.org/" 
(error) ERR unknown command ' http://www.seuenweeks.org/' 
GET ?wks 
"http://www.sevenweeks.org/" 
127. 
ø.ø.l "SET gog http://www.google.com yah http://www.yahoo. 
127. 
"GET gog yah 
1) 
"http://www . google.com" 
2) 
"http://www . yahoo.com" 

Although Redis stores strings, it recognizes integers and provides some simple operations for them.

If we want to keep a running total of how many short keys are in our dataset, we can create a count and then increment it with the INCR command.

SET count 2  
OK

INCR count  
(integer) 3

GET count  
"3"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
red is— 
T ype : 
12?.ø. 
12?.ø. 
"http: 
12?.ø. 
cli 2.8 .16 
"help e(groupY' to get a list of commands in (group) 
"help (command)" for help on (command) 
"help (tab)" to get a list of possible help topics 
'quit" to exit 
ø.I : 6379) SET ?wks http://www.seuenweeks.org/ 
GET ?wks 
//www.sevenweeks.org/" 
ø.I "http://www.seuenweeks.org/" 
com 
(error) ERR unknown command ' http://www.seuenweeks.org/' 
GET ?wks 
"http://www.sevenweeks.org/" 
127. 
ø.ø.l "SET gog http://www.google.com yah http://www.yahoo. 
"GET gog yah 
"http://www . google.com" 
"http://www . yahoo.com" 
ø.ø.l: 
ø.ø.l: 
(integer) 
2?.ø.ø.1: 
12?.ø.ø.1: 
6379) SET count 2 
6379) 
6379) 
6379) 
INCR count 
GET count 

Although GET returns count as a string, INCR recognized it as an integer and added one to it.

Any attempt to increment a noninteger ends poorly.

SET bad\_count "a"  
OK

INCR bad\_count  
(error) ERR value is not an integer or out of range

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
'quit" to exit 
SET ?wks http://www.seuenweeks. 
GET ?wks 
"http://www.sevenweeks.org/" 
12?.ø.ø.I "http://www.seuenweeks.org/" 
org/ 
com 
(error) ERR unknown command ' http://www.seuenweeks.org/' 
GET ?wks 
"http://www.sevenweeks.org/" 
com yah http://www.yahoo. 
range 
127. 
"SET gog http://www.google. 
127. 
"GET gog yah 
1) 
"http://www . google.com" 
2) 
"http://www . yahoo.com" 
127. 
ø.ø.l: 
ø.ø.l: 
6379) 
6379) 
.63?" 
.63?" 
.63?" 
SET count 2 
INCR count 
GET count 
SET bad count "a" 
INCR bad count 
(integer) 
2?.ø.ø.1. 
2?.ø.ø.1. 
2?.ø.ø.1. 
(error) 
12?.ø.ø. 
ERR value is not an integer or out 

If the value can’t be resolved to an integer, Redis rightly complains.

You can also increment by any integer (INCRBY) or decrement (DECR, DECRBY).

Transactions

We’ve seen transactions in previous databases (Oracle), and Redis’ MULTI block atomic commands are a similar concept.

Wrapping two operations like SET and INCR in a single block will complete either successfully or not at all.

But you will never end up with a partial operation.

Let’s key another short code to a URL and also increment the count all in one transaction.

We begin the transaction with the MULTI command and execute it with EXEC.

MULTI  
OK

SET prag <http://pragprog.com>  
QUEUED

INCR count  
QUEUED

EXEC  
1) OK  
2) (integer) 2

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - redis-cli exe 
com 
127. 
ø.ø.l "SET gog http://www.google.com yah http://www.yahoo. 
"GET gog yah 
"http://www . google.com" 
"http://www . yahoo.com" 
ø.ø.l: 
ø.ø.l: 
6379) 
6379) 
.63?" 
.63?" 
.63?" 
SET count 2 
INCR count 
GET count 
SET bad count "a" 
INCR bad count 
(integer) 
2?.ø.ø.1. 
2?.ø.ø.1. 
2?.ø.ø.1. 
(error) 
2?.ø.ø.1. 
12?.ø.ø.1. 
QUEUED 
12?.ø.ø.1. 
QUEUED 
2?.ø.ø.1. 
range 
ERR value is not an integer or out of 
.63?" 
.63?" 
.63?" 
.63?" 
) (integer) 4 
MULTI 
SET prag http://pragprog.com 
INCR count 
EXEC 

When using MULTI, the commands aren’t actually executed when we define them (similar to Postgres transactions).

Instead, they are queued and then executed in sequence.

Similar to ROLLBACK in SQL, you can stop a transaction with the DISCARD command, which will clear the transaction queue.

Unlike ROLLBACK, it won’t revert the database; it will simply not run the transaction at all.

The effect is identical, although the underlying concept is a different mechanism (transaction rollback vs. operation cancellation).

Complex Datatypes

So far, we haven’t seen much complex behavior.

Storing string and integer values under keys—even as transactions—is all fine and good, but most programming and data storage problems deal with many types of data.

Storing lists, hashes, sets, and sorted sets natively helps explain Redis’ popularity, and after exploring the complex operations you can enact on them, you may find you agree.

These collection datatypes can contain a huge number of values (up to 2^32 elements or more than 4 billion) per key.

That’s more than enough for all Facebook accounts to live as a list under a single key.

While some Redis commands may appear cryptic, they generally follow a good pattern. SET commands begin with S, hashes with H, and sorted sets with Z.

List commands generally start with either an L (for left) or an R (for right), depending on the direction of the operation (such as LPUSH).

Hash

Hashes are like nested Redis objects that can take any number of key-value pairs.

Redis hashes are good for storing more complex data, to store objects with multiple attributes that need to be cached

Let’s use a hash to keep track of users who sign up for our URL- shortening service.

(Note that we used colons [:] within our key.

This is a valid character that often logically separates a key into segments.

It’s merely a matter of convention, with no deeper meaning in Redis.)

MSET user:eric:name "Eric Redmond" user:eric:password s3cret  
OK

MGET user:eric:name user:eric:password  
1) "Eric Redmond"

2) "s3cret"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - 
redis-cli exe 
range 
12?.ø.ø.1: 
6379) 
2?.ø.ø.1: 
6379) 
(integer) 
2?.ø.ø.1. 
.63?" 
2?.ø.ø.1. 
.63?" 
2?.ø.ø.1. 
.63?" 
(error) 
2?.ø.ø.1. 
2?.ø.ø.1. 
QUEUED 
12?.ø.ø.1. 
QUEUED 
2?.ø.ø.1. 
SET count 2 
INCR count 
GET count 
SET bad count "a" 
INCR bad count 
ERR value is not an integer or out of 
.63?" 
.63?" 
.63?" 
.63?" 
) (integer) 4 
MULTI 
SET prag http://pragprog.com 
INCR count 
EXEC 
"SET 
"GET 
user: epic: 
user: epic: 
name 
name 
password s3cret 
"Eric Redmond" user:eric: 
user: eric : password 
"Eric Redmond" 
"s3cret" 

Instead of separate keys, we can create a hash that contains its own key- value pairs.

HMSET user:eric name "Eric Redmond" password s3cret  
OK

We need only keep track of the single Redis key to retrieve all values of the hash.

HVALS user:eric  
1) "Eric Redmond"  
2) "s3cret"

Or we can retrieve all hash keys.

HKEYS user:eric  
1) "name"  
2) "password"

Or we can get a single value, by passing in the Redis key, followed by the hash key.

Here we get just the password.

HGET user:eric password  
"s3cret"

Machine generated alternative text:
Administrator: - redis-cli exe 
range 
12?.ø.ø. 
INCR bad_count 
(error) 
ERR value is not an integer or out of 
12?.ø.ø.1. 
12?.ø.ø.1. 
QUEUED 
12?.ø.ø.1. 
QUEUED 
2?.ø.ø.1. 
.63?" 
.63?" 
.63?" 
.63?" 
) (integer) 4 
MULTI 
SET prag http://pragprog.com 
INCR count 
EXEC 
"SET user:eric: 
"GET user: epic: 
name 
name 
password s3cret 
s 3 cret 
"Eric Redmond" user:eric: 
user: eric : password 
"Eric Redmond" 
"s3cret" 
ø.ø.l HMSET user:eric name "Eric Redmond" password 
HURLS user:eric 
"Eric Redmond" 
"s3cret" 
HHEYS user:eric 
'name " 
'password" 
127. 

Unlike the document datastores Mongo and CouchDB, hashes in Redis cannot nest (nor can any other complex datatype such as lists).

In other words, hashes can store only string values.

More commands exist to delete hash fields (HDEL), increment an integer field value by some count (HINCRBY), or retrieve the number of fields in a hash (HLEN).

List

Lists contain multiple ordered values that can act both as queues (first value in, first value out) and as stacks (last value in, first value out).

They also have more sophisticated actions for inserting somewhere in the middle of a list, constraining list size, and moving values between lists.

Since our URL-shortening service can now track users, we want to allow them to keep a wishlist of URLs they’d like to visit.

To create a list of short-coded websites we’d like to visit, we set the key to USERNAME:wishlist and push any number of values to the right (end) of the list.

RPUSH eric:wishlist 7wks

RPUSH eric:wishlist gog

RPUSH eric:wishlist prag  
(integer) 3

Like most collection value insertions, the Redis command returns the number of values pushed.

In other words, we pushed three values into the list so it returns 3.

You can get the list length at any time with LLEN.

Using the list range command LRANGE, we can retrieve any part of the list by specifying the first and last positions.

All list operations in Redis use a zero- based index.

A negative position means the number of steps from the end (-1 is the last element)

LRANGE eric:wishlist 0 -1  
1) "7wks"  
2) "gog"  
3) "prag"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - 
INCR count 
QUEUED 
EXEC 
) (integer) 4 
redis-cli exe 
"Eric Redmond" user:eric: 
user: eric : password 
password s3cret 
s 3 cret 
name 
name 
"SET user:eric: 
"GET user:eric: 
"Eric Redmond" 
"s3cret" 
ø.ø.l HMSET user:eric name "Eric Redmond" password 
HURLS user:eric 
"Eric Redmond" 
"s3cret" 
HHEYS user:eric 
'name " 
'password" 
ø.ø.l: 
(integer) 
6379) RPUSH eric:wishlist ?wks 
6379) LRRNGE eric:wishlist ø - 
6379) 
gog prag 
127. 
ø.ø.l: 
'prag" 
ø.ø.l: 

LREM removes from the given key some matching values.

It also requires a number to know how many matches to remove.

Setting the count to 0 as we do here just removes them all:

LREM eric:wishlist 0 gog

Setting the count greater than 0 will remove only that number of matches, and setting the count to a negative number will remove that number of matches but scan the list from the end (right side).

To remove and retrieve each value in the order we added them (like a queue), we can pop them off from the left (head) of the list.

LPOP eric:wishlist  
"7wks"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
2) (integer) 4 
127. 
ø.ø.l "SET user:eric : name "Eric Redmond" user:eric: 
127. 
ø.ø.l "GET user:eric : name user:eric : password 
1) 
"Eric Redmond" 
2) 
"s3cret" 
127. 
ø.ø.l HMSET user:eric name "Eric Redmond" password 
127. 
HURLS user:eric 
"Eric Redmond" 
"s3cret" 
HHEYS user:eric 
'name " 
'password" 
password s3cret 
s 3 cret 
ø.ø.l: 
(integer) 
2?.ø.ø.1: 
) ' 'prag" 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
"?wks " 
12?.ø.ø.1: 
6379) RPUSH eric:wishlist ?wks 
6379) LRRNGE eric:wishlist ø - 
6379) LREM eric:wishlist ø gog 
6379) LPOP eric:wishlist 
6379) 
gog prag 

To act as a stack, after you RPUSH the values, you would RPOP from the end of the list.

All of these operations are performed (on the head and tail of the list are ) in constant time.

On the previous combination of commands, you can use LPUSH and RPOP to similar effect (a queue) or LPUSH and LPOP to be a stack.

Suppose we wanted to remove values from our wishlist and move them to another list of visited sites.

To execute this move atomically, we could wrap pop and push actions within a multiblock.

Redis provides a single command for popping values from the tail of one list and pushing to the head of another.

It’s called RPOPLPUSH (right pop, left push).

RPOPLPUSH eric:wishlist eric:visited  
"prag"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - 
"s3cret" 
HHEYS user:eric 
'name " 
'password" 
redis-cli exe 
2) 
gog prag 
ø.ø.l: 
(integer) 
2?.ø.ø.1: 
) ' 'prag" 
2?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
"?wks " 
2?.ø.ø.1: 
(error) 
2?.ø.ø. 
Invalid 
2?.ø.ø. 
Invalid 
2?.ø.ø. 
(error) 
2?.ø.ø. 
'prag" 
12?.ø.ø. 
6379) RPUSH eric:wishlist ?wks 
6379) LRRNGE eric:wishlist ø - 
6379) LREM eric:wishlist ø gog 
6379) LPOP eric:wishlist 
6379) redis . multi do 
ERR unknown command ' redis . multi' 
site 
redis .rpop(' eric :wishlist') 
argument (s) 
redis . Ipush(' eric :uisited' 
site) 
argument (s) 
end 
ERR unknown command ' end' 
RPOPLPUSH eric:wishlist epic: 
visited 

If you find the range of the wishlist, prag will be gone; it now lives under visited.

This is a useful mechanism for queuing commands.

Blocking Lists

Now that our URL shortener is taking off, let’s add some social activities—like a real-time commenting system—where people can post about the websites they have visited.

Let’s write a simple messaging system where multiple clients can push comments and one client (the digester) pops messages from the queue.

We’d like the digester to just listen for new comments and pop them as they arrive.

Redis provides a few blocking commands for this sort of purpose.

First open another terminal and start another redis-cli client.

This will be our digester.

The command to block until a value exists to pop is BRPOP.

It requires the key to pop a value from and a timeout in seconds, which we’ll set to five minutes.

BRPOP comments 300

Machine generated alternative text:
Administrator: CAWindowsXsystem32Ncmd.exe - redis-cli exe 
lib 
1 ?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
131. 
97. 
25b 
løl. 
97. 
76. 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
156 
684 
738 
563 
369 
64ø 
984 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
script ing . obj 
sds . obj 
sentinel. obj 
shal . obj 
slowlog . obj 
sort . obj 
sparkline . obj 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
63 File(s) 
4 Dir(s) 641.?53.653.248 bytes free 
: XNew folderXmsusXx64XRe lease hedis—cli.exe 
BRPOP comments 3øø 

Then switch back to the first console and push a message to comments.

LPUSH comments "Prag is great! I buy all my books there."

If you switch back to the digester console, two lines will be returned: the key and the popped value.

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
I) "name" 
2) "password" 
gog prag 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
1) "?wks" 
3) ' 'prag" 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
"?wks " 
12?.ø.ø.1: 
(error) 
12?.ø.ø. 
Invalid 
2?.ø.ø. 
Invalid 
2?.ø.ø. 
(error) 
2?.ø.ø.1: 
'prag" 
2?.ø.ø.1: 
6379) RPUSH eric:wishlist ?wks 
6379) LRRNGE eric:wishlist ø - 
6379) LREM eric:wishlist ø gog 
6379) LPOP eric:wishlist 
6379) redis . multi do 
there . " 
ERR unknown command ' redis . multi' 
site 
redis .rpop(' eric :wishlist') 
argument (s) 
redis . Ipush(' eric :uisited' 
site) 
argument (s) 
end 
ERR unknown command ' end' 
6379) RPOPLPUSH 
6379) LPUSH comments "Prag is 
6379) 
eric :uisited 
great! I buy all my books 
(integer) 
12?.ø.ø.1: 

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli.exe 
lib 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
løl. 
97. 
76. 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
563 
369 
64ø 
984 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
shal . obj 
slowlog . obj 
sort . obj 
sparkline . obj 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
63 File(s) 
4 Dir(s) 641.?53.653.248 bytes free 
C: XNew folderXmsusXx64XRe 
127. 
BRPOP comments 3øø 
'comments " 
2) 
"Prag is great! I buy all my books there . " 
(15. 
91s) 
127. 

The console will also output the length of time it spent blocking.

1) "comments"  
2) "Prag is great! I buy all my books there."  
(50.22s)

There’s also a blocking version of left pop (BLPOP) and right pop, left push (BRPOPLPUSH).

Set

Our URL shortener is shaping up nicely, but it would be nice to group common URLs in some way.

Sets are unordered collections with no duplicate values and are an excellent choice for performing complex operations between two or more key values, such as unions or intersections.

If we wanted to categorize sets of URLs with a common key, we can add multiple values with SADD.

SADD news nytimes.com pragprog.com  
(integer) 2

Redis added two values. We can retrieve the full set, in no particular order, via SMEMBERS.

SMEMBERS news  
1) "pragprog.com"  
2) "nytimes.com"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
1) "?wks" 
3) ' 'prag" 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
"?wks " 
2?.ø.ø.1: 
(error) 
2?.ø.ø. 
Invalid 
2?.ø.ø. 
Invalid 
2?.ø.ø. 
(error) 
2?.ø.ø. 
'prag" 
2?.ø.ø. 
6379) LREM eric:wishlist ø gog 
6379) LPOP eric:wishlist 
6379) redis . multi do 
there . " 
ERR unknown command ' redis . multi' 
site 
redis .rpop(' eric :wishlist') 
argument (s) 
redis . Ipush(' eric :uisited' 
site) 
argument (s) 
end 
ERR unknown command ' end' 
I : 6379) RPOPLPUSH eric :wishlist eric : visited 
I LPUSH comments "Prag is great! I buy all my books 
(integer) I 
SADD news nytimes.com pragprog.com 
(integer) 2 
SMEMBERS news 
) "nytimes.com" 
) "pragprog.com" 

Let’s add another category called *tech* for technology-related sites.

SADD tech pragprog.com apple.com  
(integer) 2

To find the intersection of websites that both provide news and are technology focused, we use the SINTER command.

SINTER news tech  
1) "pragprog.com"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
(integer) I 
there . " 
12?.ø.ø. 
"?wks " 
12?.ø.ø. 
(error) 
12?.ø.ø. 
Invalid 
12?.ø.ø. 
Invalid 
12?.ø.ø. 
(error) 
2?.ø.ø. 
'prag" 
2?.ø.ø. 
LPOP eric:wishlist 
redis . multi do 
ERR unknown command ' redis . multi' 
site 
redis .rpop(' eric :wishlist') 
argument (s) 
redis . Ipush(' eric :uisited' 
site) 
argument (s) 
end 
ERR unknown command ' end' 
I : 6379) RPOPLPUSH eric :wishlist eric : visited 
I LPUSH comments "Prag is great! I buy all my books 
(integer) I 
SADD news nytimes.com pragprog.com 
(integer) 2 
SMEMBERS news 
) "nytimes.com" 
) "pragprog.com" 
SADD tech pragprog.com apple. 
(integer) 2 
SINTER news tech 
) "pragprog.com" 
com 

Just as easily, we can remove any matching values in one set from another.

To find all news sites that are not tech sites, use SDIFF:

SDIFF news tech  
1) "nytimes.com"

We can also build a union of websites that are either news or tech.

Since it’s a set, any duplicates are dropped.

SUNION news tech  
1) "apple.com"  
2) "pragprog.com"  
3) "nytimes.com"

Machine generated alternative text:
Administrator: C:NWindcwsNsystem32Ncmd.exe - redis-cli exe 
there . " 
Invalid 
12?.ø.ø. 
Invalid 
12?.ø.ø. 
(error) 
12?.ø.ø. 
'prag" 
12?.ø.ø. 
argument (s) 
redis . Ipush(' eric :uisited' 
site) 
argument (s) 
end 
ERR unknown command ' end' 
I : 6379) RPOPLPUSH eric :wishlist eric : visited 
I LPUSH comments "Prag is great! I buy all my books 
(integer) I 
SADD news nytimes.com pragprog.com 
(integer) 2 
SMEMBERS news 
com 
I) "nytimes.com" 
2) "pragprog.com" 
(integer) 2 
SADD tech pragprog.com apple . 
SINTER news tech 
127. 
127. 
127. 
'pragprog.com" 
'nytimes.com" 
SDIFF news 
tech 
tech 
SUNION news 
"nytimes.com" 
'apple.com" 
'pragprog.com" 

That set of values can also be stored directly into a new set (SUNIONSTORE destination key [key ...]).

SUNIONSTORE websites news tech

This also provides a useful trick for cloning a single key’s values to another key, such as SUNIONSTORE news\_copy news.

Similar commands exist for storing intersections (SINTERSTORE) and diffs (SDIFFSTORE).

Just like RPOPLPUSH moved values from one list to another, SMOVE does the same for sets; it’s just easier to remember.

And like LLEN finds the length of a list, SCARD (set cardinality) counts the set; it’s just harder to remember.

Since sets are not ordered, there are no left, right, or other positional commands.

Popping a random value from a set just requires SPOP key, and removing values is SREM key value [value ...].

Unlike lists, there are no blocking commands for sets.

Sorted Sets

Whereas other Redis datatypes we’ve looked at so far easily map to common programming language constructs, sorted sets take something from each of the previous datatypes.

They are ordered like lists and are unique like sets.

They have field-value pairs like hashes, but rather than string fields, they are

instead numeric scores that denote the order of the values.

Example:

Let’s add a few selected hackers with their year of birth as “score”.

zadd hackers 1940 "Alan Kay"  
(integer) 1

zadd hackers 1953 "Richard Stallman"

zadd hackers 1965 "Yukihiro Matsumoto"

zadd hackers 1916 "Claude Shannon"

zadd hackers 1969 "Linus Torvalds"

zadd hackers 1912 "Alan Turing"

For sorted sets it’s a joke to return these hackers sorted by their birth year because actually **they are already sorted**.

zrange hackers 0 -1

1. Alan Turing  
2. Claude Shannon  
3. Alan Kay  
4. Richard Stallman  
5. Yukihiro Matsumoto  
6. Linus Torvalds

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - redis-cli exe 
I) "nytimes.com" 
) "apple.com" 
) "pragprog.com" 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
ø.ø.l: 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
SUNI ONS TORE webs ites news tech 
z add 
z add 
z add 
z add 
z add 
z add 
hackers 
hackers 
hackers 
hackers 
hackers 
hackers 
194ø " 
R Ian Kay" 
1953 
"Richard Stallman " 
1965 
"Yukihiro Matsumoto " 
1916 
"Claude Shannon" 
1969 
"Linus Torualds " 
1912 
"R Ian Turing" 
6) 
z range hackers ø 
"R Ian Turing" 
"Claude Shannon" 
"R Ian Kay" 
"Richard Stallman " 
"Yukihiro Matsumoto " 
"Linus Torualds " 

Pasted from <<http://redis-docs.readthedocs.org/en/latest/IntroductionToRedisDataTypes.html>>

You can think of sorted sets as like a random access priority queue.

This power has a trade- off, however.

Internally, sorted sets keep values in order, so inserts can take log(N) time to insert (where N is the size of the set), rather than the constant time complexity of hashes or lists.

Next we want to keep track of the popularity of specific shortcodes.

Every time someone visits a URL, the score gets increased.

Like a hash, adding a value to a sorted set requires two values after the Redis key name: the score and the member.

ZADD visits 500 7wks 9 gog 9999 prag  
(integer) 3

To increment a score, we can either re-add it with the new score, which just updates the score but does not add a new value, or increment by some number, which will return the new value.

ZINCRBY visits 1 prag  
"10000"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - 
redis-cli exe 
194ø " 
R Ian Kay" 
1953 
"Richard Stallman " 
1965 
"Yukihiro Matsumoto " 
1916 
"Claude Shannon" 
1969 
"Linus Torualds " 
1912 
"R Ian Turing" 
(integer) 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
ø.ø.l: 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
z add 
z add 
z add 
z add 
z add 
z add 
hackers 
hackers 
hackers 
hackers 
hackers 
hackers 
prag 
z range hackers ø 
"R Ian Turing" 
"Claude Shannon" 
"R Ian Kay" 
"Richard Stallman " 
"Yukihiro Matsumoto " 
"Linus 
ø.ø.l: 
(integer) 
2?.ø.ø.1: 
"1 øøøø" 
12?.ø.ø.1: 
Torualds" 
6379) ZRDD visits Søø ?wks 9 gog 9999 
6379) ZINCRBY visits 1 prag 
6379) 

You can decrement also by setting a negative number for ZINCRBY.

Do a Google search to find out uses for sorted sets in games, text search, chat rooms etc

Ranges

To get values from our visits set, we can issue a range command, ZRANGE, which returns by position, just like the list datatype’s LRANGE command.

Except in the case of a sorted set, the position is ordered by score from lowest to highest.

So, to get the top two scoring visited sites (zero-based), use this:

ZRANGE visits 0 1  
1) "gog"  
2) "7wks"

To get the scores of each element as well, append WITHSCORES to the previous code.

To get them in reverse, insert the word REV, as in ZREVRANGE.

ZREVRANGE visits 0 -1 WITHSCORES  
1) "prag"  
2) "10000"  
3) "7wks"

4) "500"  
5) "gog"  
6) "9"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
(integer) I 
zadd hackers 1912 
(integer) I 
"R Ian Turing" 
127. 
ø.ø.l z range hackers ø 
1) 
"R Ian Turing" 
2) 
"Claude Shannon" 
3) 
"R Ian Kay" 
4) 
"Richard Stallman " 
5) 
"Yukihiro Matsumoto " 
6) 
127. 
"Linus 
ø.ø.l: 
(integer) 
12?.ø.ø.1: 
"1 øøøø" 
127. 
ø.ø.l: 
ø.ø.l: 
'prag" 
) "løøøø" 
) "5øø" 
ø.ø.l: 
Torualds" 
6379) 
6379) 
6379) 
6379) 
6379) 
ZRDD visits Søø ?wks 9 gog 9999 
ZINCRBY visits I prag 
prag 
ZRRNGE visits ø 1 
ZREURRNGE visits ø - 
1 WITHSCORES 

But if we’re using a sorted set, it’s more likely we want to range by score, rather than by position.

ZRANGEBYSCORE set min max has a slightly different syntax from ZRANGE.

Since the low and high range numbers are *inclusive* by default, we can make a score number *exclusive* by prefixing it with an opening paren: (. So, this will return all scores where 9 <= score <= 9

ZRANGEBYSCORE visits 9 10000  
1) "gog"  
2) "7wks"

But the following will return 9 < score <= 10,000:

ZRANGEBYSCORE visits (9 10000

1) "7wks"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - 
redis-cli exe 
prag 
2) 
"Claude Shannon" 
3) 
"R Ian Kay" 
4) 
"Richard Stallman " 
5) 
"Yukihiro Matsumoto " 
6) 
127. 
gog 
"Linus 
ø.ø.l: 
(integer) 
12?.ø.ø.1: 
"1 øøøø" 
Torualds" 
6379) 
ZRDD visits Søø 
ZINCRBY visits 
ZRRNGE visits ø 
?wks 9 
I prag 
6379) 
ø.ø.l: 
6379) 
ø.ø.l: 
6379) 
'prag" 
) "løøøø" 
) "5øø" 
.63?" 
.63?" 
.63?" 
127. 
ZREURRNGE visits ø - 
1 WITHSCORES 
ø. 
ø. 
ø. 
ZRRNGEBYSCORE visits 9 
ZRRNGEBYSCORE visits 
(9 9999 

We can also range by both positive and negative values, including infinities.

This returns the entire set.

ZRANGEBYSCORE visits -inf inf

You can list them in reverse too, with ZREVRANGEBYSCORE.

Machine generated alternative text:
Administrator: C:NWindcwsNsystem32Ncmd.exe - redis-cli exe 
prag 
6) "Linus 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
"1 øøøø" 
127. 
ø.ø.l: 
2) 
"?wks " 
127. 
ø.ø.l: 
'prag" 
2) 
"1 øøøø" 
3) 
"?wks " 
4) 
"5øø" 
127. 
ø.ø.l: 
2) 
"?wks " 
127. 
ø.ø.l: 
1) 
"?wks " 
127. 
ø.ø.l: 
'prag" 
ø.ø.l: 
Torualds" 
?wks 9 gog 9999 
I prag 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
ZRDD visits Søø 
ZINCRBY visits 
ZRRNGE visits ø 
ZREURRNGE visits ø - 
1 WITHSCORES 
ZRRNGEBYSCORE visits 9 9999 
ZRRNGEBYSCORE visits 
ZRRNGEBYSCORE visits 
(9 9999 
—inf inf 

Along with retrieving a range of values by rank (index) or score, ZREMRANGE-

BYRANK and ZREMRANGEBYSCORE, respectively, remove values by rank or score.

Unions

Just like the set datatype, we can create a destination key that contains the union or intersection of one or more keys.

This is one of the more complex commands in Redis, since it must not only join the keys—a relatively simple operation—but also merge (possibly) differing scores.

The union operation looks like this:

ZUNIONSTORE destination numkeys key [key ...]  
 [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]

destination is the key to store into, and key is one or more keys to union.

numkeys is simply the number of keys you’re about to join, while weight is the optional number to multiply each score of the relative key by (if you have two keys, you can have two weights, and so on).

Finally, aggregate is the optional rule for resolving each weighted score and summing by default, but you can also choose the min or max between many scores.

Let’s use this command to measure the importance of a sorted set of short- codes.

First we’ll create another key that scores our short codes by votes.

Each visitor to a site can vote if they like the site or not, and each vote adds a point.

ZADD votes 2 7wks 0 gog 9001 prag  
(integer) 3

We want to figure out the most important websites in our system, as some combination of votes and visits.

Votes are important, but to a lesser extent, website visits also carry some weight (perhaps people are so enchanted by the website, they simply forget to vote).

We want to add the two types of scores together to compute a new importance score, while giving votes a weight of double importance—multiplied by two.

ZUNIONSTORE importance 2 visits votes WEIGHTS 1 2 AGGREGATE SUM  
(integer) 3

ZRANGEBYSCORE importance -inf inf WITHSCORES  
1) "gog"

2) "9"  
3) "7wks"  
4) "504"  
5) "prag"  
6) "28002"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - redis-cli exe 
4) 
"5øø" 
ø.ø.l: 
ø.ø.l: 
ø.ø.l: 
'prag" 
ø.ø.l: 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
ZRRNGEBYSCORE visits 9 9999 
ZRRNGEBYSCORE visits 
ZRRNGEBYSCORE visits 
(9 9999 
—inf inf 
ZRDD votes 2 ?wks ø gog 9øøI prag 
ZUNIONSTORE importance 2 visits votes WEIGHTS 
ZRRNGEBYSCORE importance -inf inf WITHSCORES 
1 2 AGGREGATE SUM 
(integer) 
2?.ø.ø.1: 
(integer) 
ø.ø.l: 
'prag" 
) "28øø2" 
127. 
ø.ø.l: 

This command is powerful in other ways too.

For example, if we need to double all scores of a set, we can union a single key with a weight of 2 and store it back into itself.

ZUNIONSTORE votes 1 votes WEIGHTS 2  
(integer) 2

redis 127.0.0.1:6379> ZRANGE votes 0 -1 WITHSCORES  
1) "gog"

2) "0"  
3) "7wks"  
4) "4"  
5) "prag"  
6) "18002"

Machine generated alternative text:
Administrator: CAWindcAvsNsystem32,cmd.exe - redis-cli exe 
1 2 AGGREGATE SUM 
2) "?wks" 
3) ' 'prag" 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
127. 
ø.ø.l: 
3) 
"?wks " 
4) 
"5ø4" 
'prag" 
6) 
"28øø2" 
127. 
ø.ø.l: 
(integer) 
127. 
ø.ø.l: 
3) 
"?wks " 
'prag" 
) "18øø2" 
ø.ø.l: 
6379) ZRDD votes 2 ?wks ø gog 9øø1 prag 
6379) ZUNIONSTORE importance 2 visits votes WEIGHTS 
6379) ZRRNGEBYSCORE importance -inf inf WITHSCORES 
6379) ZUNIONSTORE votes 
6379) ZRRNGE votes ø -1 
6379) 
1 votes WEIGHTS 
WITHSCORES 

Sorted sets contain a similar command (ZINTERSTORE)to perform intersections.

<http://redis.io/topics/data-types>

Expiry

A common use case for a key-value system like Redis is as a fast-access cache for data that’s more expensive to retrieve or compute.

Expiration helps keep the total key set from growing unbounded, by tasking Redis to delete a key- value after a certain time has passed.

Marking a key for expiration requires the EXPIRE command, an existing key, and a time to live in seconds.

Here we set a key and set it to expire in ten seconds.

We can check whether the key EXISTS within ten seconds and it returns a 1 (true).

If we wait to execute, it will eventually return a 0 (false).

SET ice "I'm melting..."  
OK

EXPIRE ice 10  
(integer) 1

EXISTS ice  
(integer) 1

EXISTS ice  
(integer) 0

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - redis-cli exe 
inf 
WITHSCORES 
12?.ø.ø.1: 
) ' 'prag" 
) "28øø2" 
2?.ø.ø.1: 
(integer) 
ø.ø.l: 
'prag" 
) "18øø2" 
ø.ø.l: 
ø.ø.l: 
(integer) 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
6379) 
ø 
6379) 
ZRRNGEBYSCORE importance -inf 
ZUNI ONSTORE votes 
ZRRNGE votes ø -1 
1 votes WEIGHTS 
WITHSCORES 
SET ice 
EXPIRE 
EXISTS 
EXISTS 
"I'm melting..." 
ice Iø 
Ice 
Ice 

Setting and expiring keys is so common that Redis provides a shortcut command called SETEX.

SETEX ice 10 "I'm melting..."

You can query the time a key has to live with TTL.

Setting ice to expire as shown earlier and checking its TTL will return the number of seconds left.

TTL ice  
(integer) 4

Machine generated alternative text:
Administrator: CAWindcAvsNsystem32,cmd.exe - redis-cli exe 
(integer) 
lø "1' 
lø "1' 
ZRRNGE votes ø -1 WITHSCORES 
127. 
ø.ø.l: 
3) 
"?wks " 
'prag" 
6) 
"18øø2" 
127. 
ø.ø.l: 
127. 
ø.ø.l: 
(integer) 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
2?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
6379) 
6379) 
6379) 
6379) 
6379) 
ø 
6379) 
6379) 
ø 
6379) 
6379) 
6379) 
SET ice 
EXPIRE 
EXISTS 
EXISTS 
"I'm melting.. 
ice Iø 
Ice 
Ice 
SETEX ice 
TTL ice 
SETEX ice 
TTL ice 
m melting. 
m melting.. 

At any moment before the key expires, you can remove the timeout by running PERSIST key.

PERSIST ice

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - 
lø "1' 
lø "1' 
lø "1' 
redis-cli exe 
3) 
"?wks " 
'prag" 
) "18øø2" 
127. 
ø.ø.l: 
127. 
ø.ø.l: 
(integer) 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
6379) 
6379) 
6379) 
6379) 
ø 
6379) 
6379) 
ø 
6379) 
6379) 
6379) 
6379) 
6379) 
SET ice 
EXPIRE 
EXISTS 
EXISTS 
"I'm melting.. 
ice Iø 
Ice 
Ice 
m melting..." 
m melting.. 
m melting..." 
SETEX ice 
TTL ice 
SETEX ice 
TTL ice 
SETEX ice 
PERSIST ice 

For marking a countdown to a specific time, EXPIREAT accepts a Unix timestamp (as seconds since January 1, 1970) rather than a number of seconds to count up to. <http://www.unixtimestamp.com/index.php>

e.g. 8/10/2014 17:00 = **1412787600)**

In other words, EXPIREAT is for absolute timeouts, and EXPIRE is for relative timeouts.

A common trick for keeping only recently used keys is to update the expire time whenever you retrieve a value.

This is the most recently used (MRU) caching algorithm to ensure your most recently used keys will remain in Redis, while the neglected keys will just expire as normal.

Database Namespaces

So far, we’ve interacted only with a single namespace.

Just like buckets in Riak, sometimes we need to separate keys by namespace.

For example, if you wrote an internationalized key-value store, you could store different translated responses in different namespaces.

The key greeting could be set to “guten tag” in a German namespace and “bonjour” in French.

When a user selects their language, the application just pulls all values from the namespace assigned.

In Redis nomenclature, a namespace is called a *database* and is keyed by number.

So far, we’ve always interacted with the default namespace 0 (also known as database 0).

Here we set greeting to the English hello.

SET greeting hello  
OK

GET greeting  
"hello"

But if we switch to another database via the SELECT command, that key is unavailable.

SELECT 1  
OK  
GET greeting  
(nil)

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - 
lø "1' 
lø "1' 
lø "1' 
ø.l: 
redis-cli exe 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
2?.ø.ø.1: 
(integer) 
6379) 
6379) 
ø 
6379) 
6379) 
ø 
6379) 
6379) 
6379) 
6379) 
2?.ø.ø.1. 
.63?" 
2?.ø.ø.1. 
.63?" 
"he 110 " 
.63?" 
EXISTS ice 
EXISTS ice 
SETEX ice 
TTL ice 
SETEX ice 
TTL ice 
SETEX ice 
PERSIST ice 
m melting..." 
m melting..." 
m melting..." 
SET greeting hello 
GET greeting 
SELECT 1 
12?.ø. 
(nil) 
12?.ø. 
6379 11 D GET greeting 
.63?9 

And setting a value to this database’s namespace will not affect the value of the original.

SET greeting "guten tag"  
OK

SELECT 0  
OK

GET greeting  
"hello"

Machine generated alternative text:
Administrator: CAWindcAvsXsystem32,cmd.exe - redis-cli exe 
ø.l: 
ø.l: 
ø.l: 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
6379) 
ø 
6379) 
6379) 
6379) 
6379) 
12?.ø.ø.1. 
.63?" 
12?.ø.ø.1. 
.63?" 
"he 110 " 
.63?" 
TTL ice 
SETEX ice lø 
TTL ice 
SETEX ice lø 
PERSIST ice 
"I'm melting..." 
"I'm melting.. 
tag" 
SET greeting hello 
GET greeting 
SELECT 1 
12?.ø. 
12?.ø. 
(nil) 
12?.ø. 
12?.ø. 
12?.ø. 
6379 11 D GET greeting 
6379 SET greeting "guten 
-6379 LID SELECT ø 
6379) GET greeting 
.63?" 
"he 110 " 
12?.ø.ø.1. 

Since all databases are running in the same server instance, Redis lets us shuffle keys around with the MOVE command.

Here we move greeting to database 2:

MOVE greeting 2  
(integer) 2

SELECT 2  
OK

GET greeting  
"hello"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli exe 
ø.l: 
ø.l: 
ø.l: 
12?.ø.ø.1: 
12?.ø.ø.1: 
(integer) 
6379) 
6379) 
12?.ø.ø.1. 
.63?" 
12?.ø.ø.1. 
.63?" 
"he 110 " 
.63?" 
SET EX ice Iø "I'm melting..." 
PERSIST ice 
SET greeting hello 
GET greeting 
SELECT 1 
tag" 
12?.ø. 
12?.ø. 
(nil) 
12?.ø. 
12?.ø. 
6379 11 D GET greeting 
6379 SET greeting "guten 
-6379 LID SELECT ø 
6379) GET greeting 
6379) MOUE greeting 2 
.63?" SELECT 2 
6379 12 D GET greeting 
.63?9 
"he 110 " 
2?.ø.ø.1: 
(integer) 
2?.ø.ø.1. 
2?.ø.ø.1: 
"he 110 " 
12?.ø.ø.1. 

This can be useful for running different applications against a single Redis server but still allow these multiple applications to trade data between each other.

And There’s More

Redis has plenty of other commands for actions such as renaming keys (RENAME), determining the type of a key’s value (TYPE), and deleting a key-value (DEL).

There’s also the painfully dangerous FLUSHDB, which removes all keys from this Redis database, and its apocalyptic cousin, FLUSHALL, which removes all keys from all Redis databases.

Check out the online documentation for the full list of Redis commands. <http://redis.io/commands>

Summary

The datatypes of Redis and the complex queries it can perform make it much more than a standard key-value store.

It can act as a stack, queue, or priority queue; can be an object store (via hashes); and even can perform complex set operations such as unions, intersections, and subtractions (diff).

It provides many atomic commands, and for those multistep commands, it provides a transaction mechanism.

It has a built-in ability to expire keys, which is useful as a cache.

Distribution

Next we’ll look at some of the advanced functions provided by Redis, such as the publish-subscribe model, system configuration, and replication

Beyond that, we’ll look at how to create a Redis cluster and how to store a lot of data quickly.

A Simple Interface

At 20,000 lines of source code, Redis is a fairly simple project.

But beyond code size, it has a simple interface that accepts the very strings we have been writing in the console.

Telnet

We can interact without the command-line interface by streaming commands through TCP on our own via telnet and terminating the command with a carriage return line feed (CRLF, or \r\n).

telnet localhost 6379

Trying 127.0.0.1... Connected to localhost. Escape character is *'^]'*.

**SET** test hello

+OK

GET test

$5 hello

Machine generated alternative text:
Telnet localhost 
ET test 

SADD stest 1 99

:2

SMEMBERS stest

\*2

$1 1

$2 99

CTRL-]

We can see that our input is the same as we provided to the console, but the

console cleaned up the responses a bit.

1 Redis streams the OK status prefixed by a + sign.

2 Before it returned the string *hello*, it sent $5, which means “the following string is five characters.”

3 The number 2 after we add two set items to the test key is prefixed by : to represent an integer (two values were added successfully).

4 Finally, when we requested two items, the first line returned begins with an asterisk and the number 2—meaning there are two complex values about to be returned.

The next two lines are just like the *hello* string but contain the string *1*, followed by the string *99*.

publish-subscribe

Earlier we were able to implement a rudimentary blocking queue using the list datatype.

We queued data that could be read by a blocking pop command.

Using that queue, we made a very basic publish-subscribe model.

Any number of messages could be pushed to this queue, and a single queue reader would pop messages as they were available.

This is powerful but limited.

Under many circumstances we want a slightly inverted behavior, where several subscribers want to read the announcements of a single publisher, as shown on page 278.

Redis provides some specialized publish-subscribe (or pub-sub) commands.

Let’s improve on the commenting mechanism we made yesterday using blocking lists, by allowing a user to post a comment to multiple subscribers (as opposed to just one).

We start with some subscribers that connect to a key, known as a *channel* in pub-sub nomenclature.

Let’s start two more clients and subscribe to the comments channel.

Subscribing will cause the CLI to block.

SUBSCRIBE comments  
Reading messages... (press Ctrl-C to quit)  
1) "subscribe"  
2) "comments"

3) (integer) 1

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli.exe 
lib 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
63 File(s) 
4 Dir(s) 641.?46.644.992 bytes free 
C: XNew folderXmsusXx64XRe 
SUBSCRIBE comments 
Reading messages. 
(press Ctrl—C to quit) 
I) "subscribe " 
2) "comments" 
3) (integer) I 

Machine generated alternative text:
Administrator: C:NWindcwsN.system32Ncmd.exe - redis-cli exe 
lib 
1 ?zø9nø14 
?zø9nø14 
?zø9nø14 
?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
97. 
76. 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
369 
64ø 
984 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
slowlog . obj 
sort . obj 
sparkline . obj 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
63 File(s) 
4 Dir(s) 641.?46.644.992 bytes free 
C: XNew folderXmsusXx64XRe 
SUBSCRIBE comments 
Reading messages... 
(press Ctrl—C to quit) 
I) "subscribe " 
2) "comments" 
3) (integer) I 

With two subscribers, we can publish any string we want as a message to the comments channel.

The PUBLISH command will return the integer 2, meaning two subscribers received it.

PUBLISH comments "Check out this shortcoded site! 7wks"  
(integer) 2

Machine generated alternative text:
Administrator: CAWindowsXsystem32,cmd.exe - redis-cli.exe 
12?.ø.ø.1: 
(integer) 
ø.l: 
ø.l: 
ø.l: 
this 
shortcoded site! 
?wks " 
6379) PERSIST ice 
12?.ø.ø.1: 
6379) SET greeting hello 
12?.ø.ø.1: 
6379) GET greeting 
"he 110 " 
.63?" SELECT 1 
6379 11 D GET greeting 
6379 SET greeting "guten 
-6379 LID SELECT ø 
6379) GET greeting 
6379) MOUE greeting 2 
6379) SELECT 2 
tag" 
"Check out 
12?.ø. 
12?.ø. 
(nil) 
12?.ø. 
12?.ø. 
12?.ø. 
"he 110 " 
12?.ø.ø.1: 
(integer) 
12?.ø.ø.1: 
12?.ø.ø.1: 
"he 110 " 
12?.ø.ø.1: 
(integer) 
2?.ø.ø.1: 
6379 
6379 
6379 
GET greeting 
PUBLISH comments 

Both of the subscribers will receive a *multibulk reply* (a list) of three items: the string “message,” the channel name, and the published message value.

1) "message"  
2) "comments"  
3) "Check out this shortcoded site! 7wks"

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - redis-cli.exe 
lib 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
.2M bytes 
63 File(s) 
4 Dir(s) 641.?46.644.992 bytes free 
C: XNew folderXmsusXx64XRe 
SUBSCRIBE comments 
Reading messages. 
(press Ctrl—C to quit) 
1) 
3) 
3) 
"subscribe " 
'comments " 
(integer) I 
'message " 
'comments " 
"Check out this 
shortcoded site! 
?wks " 

When your clients want to no longer receive correspondence, they can execute the UNSUBSCRIBE comments command to disconnect from the comments channel or simply UNSUBSCRIBE alone to disconnect from all channels.

However, note in redis-cli that you will have to press CTRL+C to break the connection.

Server Info

Before getting into changing Redis’s system settings, it’s worth taking a quick look at the INFO command, since changing settings values will alter some of these values as well.

INFO outputs a list of server data, including version, process ID, memory used, and uptime.

INFO

redis\_version:2.4.5  
redis\_git\_sha1:00000000  
redis\_git\_dirty:0

arch\_bits:64  
multiplexing\_api:kqueue  
process\_id:54046  
uptime\_in\_seconds:4  
uptime\_in\_days:0  
lru\_clock:1807217

...

Machine generated alternative text:
Administrator: - 
redis-cli exe 
ke yspace_misses : I 
pubsub channels : I 
pubsub_patterns : ø 
latest_fork usec:Ø 
Replication 
role : master 
connected slaves:Ø 
master_repl offset:Ø 
repl_backlog active 
ize : I ø485?6 
irst _byte _of f set : 
len : 
sed_cpu_sys : . ø5 
sed_cpu_s ys _children : . øø 
children :ø.øø 
He yspace 
dbø : ke ys —19. expires aug_tt I 
dbl : ke ys —I expires aug_tt 
db2 : ke ys —I expires aug_tt 
ø 

You may want to revisit this command later, because it provides a useful snapshot of this server’s global information and settings.

It even provides information on durability, memory fragmentation, and replication server status.

Redis Configuration

So far, we’ve only used Redis out of the box.

Much of Redis’s power comes from its configurability, allowing you to tailor settings to your use case.

The redis.conf file (or redis.windows.conf) that comes with the distribution—found in /etc/redis on \*nix systems (in msvs/x64/Release/pub in Windows) —is fairly self-explanatory, so we’re going to cover only a portion of the file.

We’ll go through a few of the common settings in order.

daemonize no  
port 6379  
loglevel verbose  
logfile stdout  
database 16

By default daemonize is set to *no*, which is why the server always starts up in the foreground.

This is nice for testing but not very production friendly.

Changing this value to *yes* will run the server in the background while setting the server’s process ID in a pid file.

The next line is the default port number for this server, port 6379.

This can be especially useful when running multiple Redis servers on a single machine.

loglevel defaults to verbose, but it’s good to set it to notice or warning in production

logfile outputs to stdout (standard output, the console), but a filename is necessary if you run in daemonize mode.

database sets the number of Redis databases we have available.

We saw how to switch between databases yesterday.

If you plan to only ever use a single database namespace, it’s not a bad idea to set this to 1.

Durability

Redis has a few persistence options.

First is no persistence at all, which will simply keep all values in main memory.

If you’re running a basic caching server, this is a reasonable choice since durability always increases latency.

One of the things that sets Redis apart from other fast-access caches like memcached is its built-in support for storing values to disk.

By default, key- value pairs are only occasionally saved.

You can run the LASTSAVE command to get a Unix timestamp of the last time a Redis disk write succeeded, or you can read the last\_save\_time field from the server INFO output.

You can force durability by executing the SAVE command (or BGSAVE, to asynchronously save in the background).

SAVE

Machine generated alternative text:
Administrator: CAWindowsXsystem32Ncmd.exe - redis-cli.exe 
lib 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
1 ?zø9nø14 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
14: 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
42 
97. 
25b 
løl. 
97. 
76. 
74. 
114. 
13b 
115. 
98. 
2øe. 
84. 
225. 
9.467. 
:tøe. 
83. 
71. 
41. 
684 
738 
563 
369 
64ø 
984 
69ø 
øsø 
919 
2ø9 
963 
523 
28ø 
572 
ø35 
144 
eø2 
eø6 
sds . obj 
sentinel. obj 
shal . obj 
slowlog . obj 
sort . obj 
sparkline . obj 
sync io . obj 
t_hash. obj 
t _ list . obj 
t set. obj 
t string. obj 
t_zset . obj 
util. obj 
uc12ø.pdb 
Win32 _ Interop. 
ziplist . obj 
z ipmap . obj 
zmaIIoc . obj 
-965 bytes 
64 File(s) 
4 Dir(s) 641.?83.988.224 bytes free 
C: XRedis Xmsus Xx64XRe lease hedis—c Ii . exe 
SAUE 

If you read the redis-server log, you will see lines similar to this:

[46421] 10 Oct 19:11:50 \* Background saving started by pid 52123  
[52123] 10 Oct 19:11:50 \* DB saved on disk  
[46421] 10 Oct 19:11:50 \* Background saving terminated with success

Machine generated alternative text:
Administrator: C:NWindowsNsystem32Ncmd.exe - 
redis-server.äe 
Redis 2.8 .16 (øøøøøøøøZø) 64 bit 
Running in stand alone mode 
conf 
159561 19 
159561 19 
159561 19 
t 6379 
159561 19 
Sep 
Sep 
Sep 
Sep 
ø9: 
ø9: 
ø9: 
ø9: 
19: 
19: 
19: 
19: 
21. 
21. 
21. 
Port: 6379 
PID: 5956 
http://redis. 
79? Server started. Redis version 
10 
2.8 .16 
813 
813 
521 
DB loaded from disk: ø.ø16 seconds 
The server is now ready to accept connections 
DB saved on disk 
po 

Another durability method is to alter the snapshotting settings in the configuration file.

Snapshotting

We can alter the rate of storage to disk by adding, removing, or altering one of the save fields.

By default there are three, prefixed by the save keyword followed by a time in seconds and a minimum number of keys that must change before a write to disk occurs.

Machine generated alternative text:
C:XRedisXmsvsXx64XReIeaseXpubXredis.windows.conf - Notepad* (Administrator) 
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? 
change log | sqldeveloperbat | sqldeveloper.conf | unlock examsql 
It is also possible to rerave all the previously configured save 
# points by adding a save directive with a single empty string argument 
like in the following example : 
s ave 
100 
101 
20s 
IOS 
106 
110 
111 
112 
113 
11 s 
116 
117 
119 
save 
save 
s ave 
# By 
900 1 
300 10 
60 10000 
default will stop accepting writes if RDB snapshots are enabled 
UTE-8 w/0 80M 
(at least one save point) and the latest background save failed 
This will make the user aware (in an hard way) that data is not persisting 
on disk properly, otherwise chances are that no one will notice and some 
I I happen 
If the background saving process will start working again will 
automatically allow writes again 
However if have W3ur proper monitoring of the server 
and persistence, may want to disable this feature so that will 
continue to work as usually even if there are problems with disk, 
permissions, and so forth 
yes 
# Compress string objects using LZF when dump databases? 
# For default that's set to ' yes' as it's almost always a win 
# If want to save some CPU in the saving child set it to 'no' but 
# the dataset will I ikely be bigger if you have compressible values or keys . 
length : 29147 lines: 646 
Ln:gg col:2 sel:010 
DoslWindows 
Normal text file 

For example, to trigger a save every 5 minutes (300 seconds) if any keys change at all, you would write the following:

save 300 1

The configuration has a good set of defaults.

The set means if 10,000 keys change, save in 60 seconds; if 10 keys change, save in 300 seconds, and any key changes will be saved in at least 900 seconds (15 minutes).

save 900 1  
save 300 10  
save 60 10000

You can add as many or few save lines as necessary to specify precise thresholds.

Append-Only File

Redis is *eventually durable* by default, in that it asynchronously writes values to disk in intervals defined by our save settings, or it is forced to write by client-initiated commands.

This is acceptable for a second-level cache or session server but is insufficient for storing data you need to be durable, like financial data. (**2nd Level Cache – on disk, 1st Level Cache in memory**, which reduces the calls made to the DB server, by conserving data already loaded from the database. Database access is therefore, necessary only when the retrieving data is currently not available in the cache.)

If a Redis server crashes, our users might not appreciate having lost money.

Redis provides an append-only file (appendonly.aof) that keeps a record of all write commands.

If the server crashes before a value is saved, it executes the commands on startup, restoring its state; appendonly must be enabled by setting it to yes in the redis.conf file.

appendonly yes

Then we must decide how often a command is appended to the file.

Setting always is the more durable, since every command is saved.

It’s also slow, which often negates the reason people have for using Redis.

By default everysec is enabled, which saves up and writes commands only once a second.

This is a decent trade-off, since it’s fast enough, and worst case you’ll lose only the last one second of data.

Finally, no is an option, which just lets the OS handle flushing.

It can be fairly infrequent, and you’re often better off skipping the append-only file altogether rather than choosing it.

# appendfsync always  
appendfsync everysec  
# appendfsync no

Append-only has more detailed parameters, which may be worth reading about in the config file when you need to respond to specific production issues.

Security

Although Redis is not natively built to be a fully secure server, you may run across the requirepass setting and AUTH command in the Redis documentation.

These can be safely ignored, since they are merely a scheme for setting a plain-text password.

Since a client can try nearly 100,000 passwords a second, it’s almost a moot point, beyond the fact that plain-text passwords are inherently unsafe anyway.

If you want Redis security, you’re better off with a good firewall and SSH security.

Interestingly, Redis provides command-level security through obscurity, by allowing you to hide or suppress commands.

This will rename the FLUSHALL command (remove all keys from the system) into some hard-to-guess value like c283d93ac9528f986023793b411e4ba2:

his feature is available as a statement that can be used inside the redis.conf configuration file.

rename-command FLUSHALL c283d93ac9528f986023793b411e4ba2

If we attempt to execute FLUSHALL against this server, we’ll be hit with an error.

The secret command works instead.

FLUSHALL  
(error) ERR unknown command 'FLUSHALL'

c283d93ac9528f986023793b411e4ba2  
OK

Or better yet, we can disable the command entirely by setting it to a blank string.

rename-command FLUSHALL ""

You can set any number of commands to a blank string, allowing you a modicum of customization over your command environment.

Tweaking Parameters

There are several more advanced settings for speeding up slow query logs, encoding details, making latency tweaks, and importing external config files.

To aid in testing your server configuration, Redis provides an excellent benchmarking tool.

It connects locally to port 6379 by default and issues 10,000 requests using 50 parallel clients.

We can execute 100,000 requests with the -n argument.

**$ redis-benchmark -n 100000**

====== PING (inline) ======

100000 requests completed in 3.05 seconds  
 50 parallel clients  
 3 bytes payload  
 keep alive: 1

5.03% <= 1 milliseconds  
98.44% <= 2 milliseconds  
99.92% <= 3 milliseconds  
100.00% <= 3 milliseconds  
32808.40 requests per second  
...

Machine generated alternative text:
Administrator: C:NWindowsXsystem32Ncmd.exe 
ACRNGE_6øø (first 6øø elements): 
C : XRed is Xmsus Xx64XRe lease he d is —benchmark . exe 
n løø 
- PING_INLINE 
Iøø requests completed in 
5ø parallel clients 
3 bytes payload 
keep alive: I 
Iøø.øøz ø milliseconds 
I . "J requests per second 
- PING_BULK 
Iøø requests completed in 
5ø parallel clients 
3 bytes payload 
keep alive: I 
Iøø.øøz ø milliseconds 
I . "J requests per second 
Iøø requests completed in 
5ø parallel clients 
3 b tes a load 
ø.øø 
ø.øø 
ø.23 
seconds 
seconds 
seconds 

Other commands are tested as well, like SADD and LRANGE; the more complex ones generally taking more time.

Master-Slave Replication

Just like other NoSQL databases (such as MongoDB and Neo4j), Redis supports master-slave replication.

One server is the master by default if you don’t set it as a slave of anything. Data will be replicated to any number of slave servers.

Making slave servers is easy.

We first need a copy of our redis.conf file.

**$ copy redis.conf redis-s1.conf**

The file will remain largely the same but with the following changes:

port 6380

slaveof 127.0.0.1 6379

If all went according to plan, you should see something similar to the following in the slave server’s log when you start it:

**$ redis-server ./conf/redis-s1.conf**

[9003] 16 Oct 23:51:52 \* Connecting to MASTER...  
[9003] 16 Oct 23:51:52 \* MASTER <-> SLAVE sync started  
[9003] 16 Oct 23:51:52 \* Non blocking connect for SYNC fired the event.  
[9003] 16 Oct 23:51:52 \* MASTER <-> SLAVE sync: receiving 28 bytes from master  
[9003] 16 Oct 23:51:52 \* MASTER <-> SLAVE sync: Loading DB in memory  
[9003] 16 Oct 23:51:52 \* MASTER <-> SLAVE sync: Finished with success

Machine generated alternative text:
Administrator: CAWindowsXsystem32,cmd.exe - 
redis-server.exe redis.windows-sl.conf 
171561 19 
171561 19 
t 638ø 
171561 19 
171561 19 
171561 19 
171561 19 
171561 19 
step) 
171561 19 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
39: 
39: 
39: 
39: 
39: 
39: 
39: 
39: 
.39: 
.39: 
.39: 
.39: 
12. 
12. 
12. 
12. 
12. 
12. 
12. 
12. 
12. 
13. 
13. 
13. 
759 
759 
993 
øøe 
øøe 
ø24 
Server started. Redis version 2.8 .16 
The server is now ready to accept connections on po 
Connecting to MASTER 
MASTER SLRUE sync started 
Non blocking connect for SYNC fired the event. 
Master replied to PING. replication can continue . 
Partial resynchronization not possible (no cached m 
Full resync from 
master: 
MASTER 
MASTER 
MASTER 
MASTER 
SLRUE 
SLRUE 
SLRUE 
SLRUE 
sync : 
sync : 
sync : 
sync : 
receiving "1226 bytes from 
Flushing old data 
Loading DB in memory 
Finished with success 
8ccd3469e4819b:1 
17156 1 
aster 
17156 1 
17156 1 
17156 1 
19 
19 
19 
19 
sep ø9. 
sep ø9. 
sep ø9. 
sep ø9. 

And you should see the string 1 slaves output in the master log.

Machine generated alternative text:
Select Administrator: C:NWindowsNsystem32Ncmd.exe - 
redis-server.äe 
po 
159561 19 
159561 19 
159561 19 
6379 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
Sep 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
ø9: 
19: 
19: 
19: 
19: 
31: 
31: 
31: 
32: 
32: 
32: 
39: 
39: 
39: 
39: 
39: 
39: 
21. 
21. 
21. 
53. 
54. 
54. 
55. 
55. 
55. 
12. 
12. 
12. 
12. 
12. 
12. 
79? 
813 
813 
521 
12ø 
166 
løø 
224 
271 
946 
993 
993 
Server started. Redis version 2.8 .16 
DB loaded from disk: ø.ø16 seconds 
The server is now ready to accept connections 
DB saved on disk 
Iøø changes in 3øø seconds. Saving.. 
fork operation complete 
Background saving terminated with success 
Iøøøø changes in 6ø seconds. Saving... 
fork operation complete 
Background saving terminated with success 
Slave asks for synchronization 
Full resync requested by slave. 
Starting BGSRUE for SYNC 
fork operation complete 
Back round savin terminated with success 
nchronization with slave succeeded 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
159561 
19 
19 
19 
19 
19 
19 
19 
19 
19 
19 
19 
19 
19 

SADD meetings "StarTrek Pastry Chefs" "LARPers Intl."

If we connect the command line to our slave, we should receive our meeting list.

SMEMBERS meetings  
1) "StarTrek Pastry Chefs"  
2) "LARPers Intl."

Machine generated alternative text:
Administrator: CAWindowsIsystem32,cmd.exe - redis-cli.exe -h 127.0.0.1 -p 6380 
SCAN 
cursor LMRTCH pattern) 'COUNT count) 
Incrementally iterate the keys space 
summary: 
SORT 
key IBY pattern) L LIMIT offset count) 'GET pattern 
:DESCI LRLPHRI 'STORE destination) 
'GET 
pattern 
summary: 
summary: 
summary: 
Sort the elements in a list. set or sorted set 
Get the time to live for a key 
Determine the type stored at key 
c I i. exe 
-h 12?.ø.ø.1 
quit 
: XRedis Xmsus Xx64XRe lease Xpub)redis— 
SMEMBERS meetings 
) "Starr rek Pastry Chefs" 
) "LRRPers Intl." 
-p 638ø 

In production, you’ll generally want to implement replication for availability or backup purposes and thus have Redis slaves on different machines.

Summary

We moved beyond simple operations into squeezing every last bit of speed out of a very fast system.

Redis provides for fast and flexible data structure storage and simple manipulations but is equally adept at more complex behaviors by way of built-in publish-subscribe functions.

It’s also highly configurable, with many durability and replication settings that conform to whatever your needs may be.

It also supports some nice third- party enhancements, like Bloom filters and clustering (not covered)

Databases are often and increasingly used in concert with each other.

See the final chapter for a rich multidatabase music solution including Redis, CouchDB, Neo4J, and Postgres—using Node.js to cement it together.

Wrap-Up

The Redis key-value (or data structure) store is light and compact, with a variety of uses.

Above all, Redis is fast, simple, and as durable as you choose.

While rarely a stand-alone database, Redis is a perfect **complement** to any polyglot ecosystem as an ever-present helper for transforming data, caching requests, or managing messages by way of its blocking commands.

Redis’s Strengths

The obvious strength of Redis is speed, like so many key-value stores of its ilk.

But more than most key-value stores, Redis provides the ability to store complex values like lists, hashes, and sets, and retrieve them based through operations specific to those datatypes.

Beyond even a data structure store, however, Redis’s durability options allow you to trade speed for data safety up to a fairly fine point.

Built-in master-slave replication is another nice way of ensuring better durability without requiring the slowness of syncing an append-only file to disk on every operation.

Additionally, replication is great for very high-read systems.

Redis’s Weaknesses

Redis is fast largely because it resides in memory.

Some may consider this cheating, since of course a database that never hits the disk will be fast.

A main memory database has an inherent durability problem; namely, if you shut down the database before a snapshot occurs, you can lose data.

Even if you set the append-only file to disk sync on every operation, you run a risk with playing back expiry values, since time-based events can never be counted on to replay in exactly the same manner—though in fairness this case is more hypothetical than practical.

Redis also does not support datasets larger than your available RAM (Redis is removing virtual memory support), so its size has a practical limitation.

Although there is a Redis Cluster currently in development to grow beyond a single-machine’s RAM requirements, anyone wanting to cluster Redis must currently roll their own with a client that supports it

Parting Thoughts

Redis is chock-full of commands—more than 120 of them.

Most commands are straightforward enough to understand by their names alone, once you get used to the idea that seemingly random letters will be removed (for example, INCRBY) or that mathematical precision can sometimes be more confusing than helpful (for example, ZCOUNT, or sorted set count, vs. SCARD, or set cardinality).

Redis is already becoming an integral part of many systems.

Several open source projects rely on Redis, from Resque, a Ruby-based asynchronous job queueing service, to session management in the Node.js project SocketStream.

Regardless of the database you choose as your SOR, you should certainly add Redis to the mix.

References:

Seven Databases

<https://en.wikipedia.org/wiki/Redis>