redis the swiss army knife of NoSQL

redis

@PeterBell

- introduction
- installation
- optimization
- deployment
- use cases
- resources

introduction

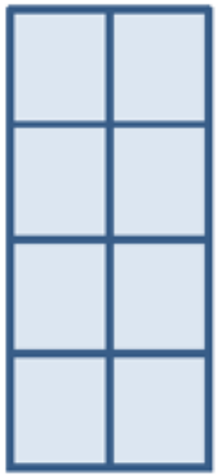
- what is redis?
- key concepts
- benchmarks
- comparisons

introduction

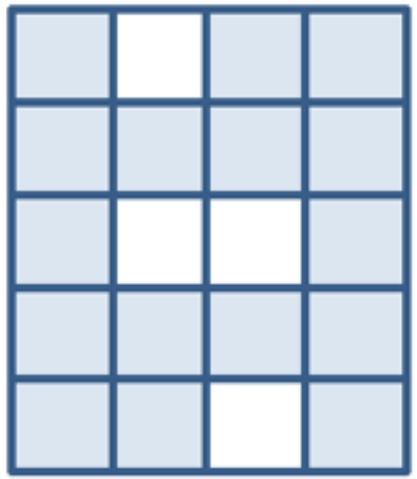
what is redis?

NoSQL

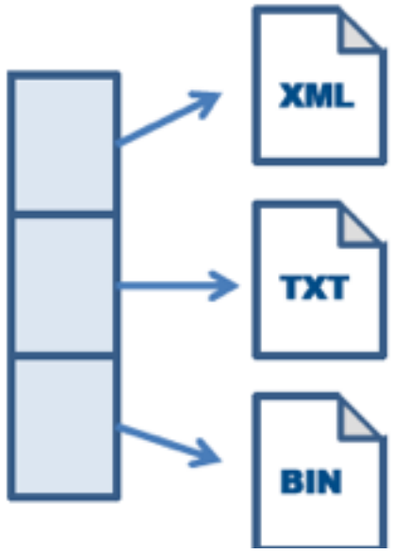
Key / Value



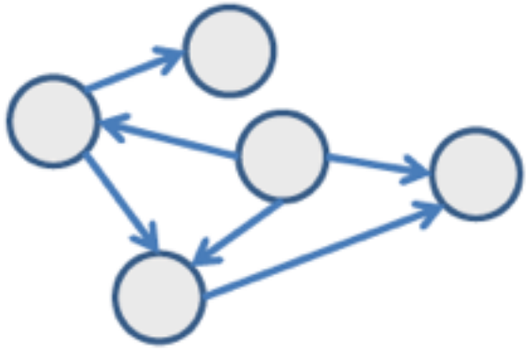
Column



Document



Graph



key/value?

Redis is an open source, BSD licensed, advanced **key-value store**. It is often referred to as a **data structure server** since keys can contain strings, hashes, lists, sets and sorted sets.

data structure server

“I see Redis definitely more as a flexible tool than as a solution specialized to solve a specific problem.”

- Salvatore Sanfilippo

key concepts

in memory

persistence options

master-slave replication

failover/HA/recovery
is **your** problem

really high performance

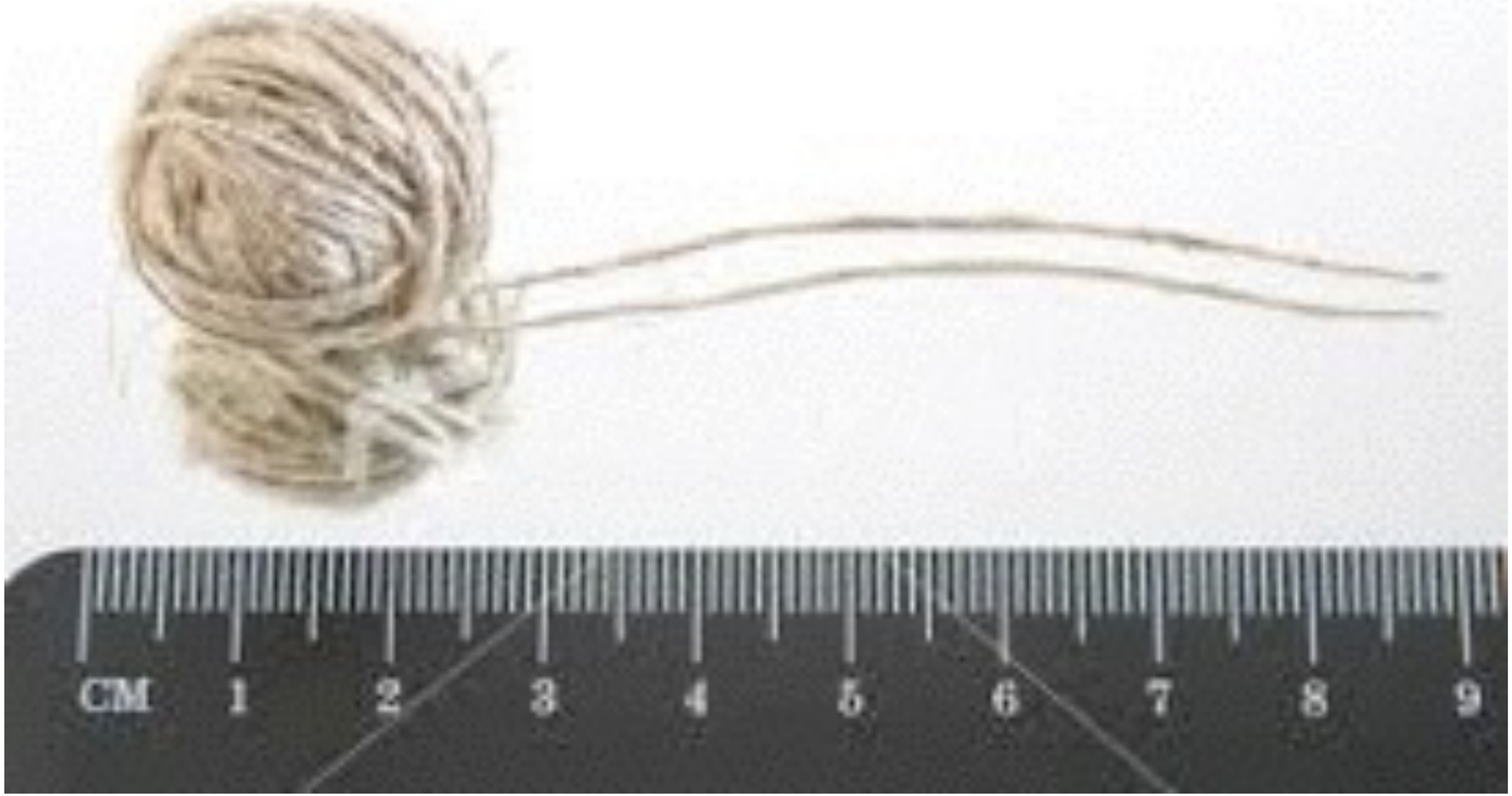
data types are key

optimized for linux

supports transactions and pipelining

supports lua scripting

benchmarks


CM
1
2
3
4
5
6
7
8
9

```
Usage: redis-benchmark [-h <host>] [-p <port>] [-

  -h <hostname>      Server hostname (default 127.
  -p <port>          Server port (default 6379)
  -s <socket>        Server socket (overrides host
  -c <clients>       Number of parallel connection
  -n <requests>      Total number of requests (def
  -d <size>          Data size of SET/GET value in
  -k <boolean>       1=keep alive 0=reconnect (def
  -r <keyspacelen>   Use random keys for SET/GET/I
```

```
$ ./redis-benchmark -r 1000000 -n 2000000 -t get,set,
SET: 552028.75 requests per second
GET: 707463.75 requests per second
LPUSH: 767459.75 requests per second
LPOP: 770119.38 requests per second
```

Intel(R) Xeon(R) CPU E5520 @ 2.27GHz (with pipelining)

```
$ ./redis-benchmark -r 1000000 -n 2000000 -t get,set,
SET: 122556.53 requests per second
GET: 123601.76 requests per second
LPUSH: 136752.14 requests per second
LPOP: 132424.03 requests per second
```

Intel(R) Xeon(R) CPU E5520 @ 2.27GHz (no pipelining)

Who's using Redis?

Logos are linked to the relevant story when available.

http://redis.io/topics/whos-using-redis

39 systems in

Rank	Last Month	DBMS	Database Model
1.	1.	Redis	Key-value store
2.	2.	Memcached	Key-value store
3.	3.	Riak	Key-value store
4.	↑ 5.	DynamoDB	Key-value store
5.	↓ 4.	Ehcache	Key-value store
6.	↑ 7.	SimpleDB	Key-value store
7.	↓ 6.	Berkeley DB	Key-value store
8.	8.	Hazelcast	Key-value store

http://db-engines.com/en/ranking/key-value+store

comparisons

ORACLE®

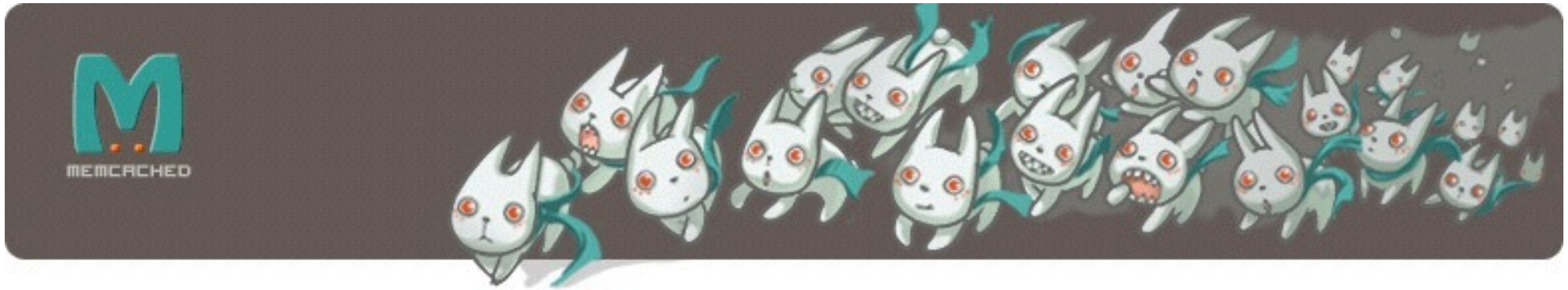
ORACLE®

Neo4j

the world's
leading graph database

mongoDB

Cassandra


MEMCACHED

installation

http://try.redis.io/

versions

```
$ wget http://download.redis.io/releases/redis-2.8.6.tar.gz
$ tar xzf redis-2.8.6.tar.gz
$ cd redis-2.8.6
$ make
```

Run Redis with:

```
$ src/redis-server
$ src/redis-cli
```

Microsoft®

Windows®

dmajkic / redis

forked from antirez/redis

Watch 65 Sta

Redis key-value store (Win32 / Win64 port) http://code.google.com/p/redis

2,086 commits 10 branches 17 releases 34 contributors

branch: 2.4 redis /

This branch is 0 commits ahead and 0 commits behind 2.4 Pull Request Compare

Update README 1 comment

dmajkic authored a year ago latest commit 6fe0b6c345

https://github.com/dmajkic/redis/

MSOpenTech / redis

forked from antirez/redis

Watch 210 St

Redis is an in-memory database that persists on disk. The data model is key-value, but many different kind of values are supported: Strings, Lists, Sets, Sorted Sets, Hashes http://redis.io

3,038 commits | 5 branches | 97 releases | 90 contributors

branch: 2.6 redis /

This branch is 0 commits ahead and 0 commits behind 2.6 #62 Compare

Merge pull request #58 from Gusi/2.6 ...

jepickett authored 13 days ago latest commit 8f0f6f64fc

https://github.com/MSOpenTech/redis

key data types

string

string

- set <key> <value>
- get <key>
- del <key>

numerical strings

numerical strings

- incr <key>
- incrby <key> <value>
- decr <key>
- decrby <key> <value>
- incrbyfloat <key> <fp value>

bitwise with string

list

list

- [l|r]push <key> <value>
- [l|r]pop <key>
- lindex <key> <index>
- lrange <key> <start> <end>

set

set

- sadd <key> <value>
- smembers <key>
- sismember <key>
- srem <key>

hash

hash

- hset <hash> <key> <value>
- hget <hash> <key>
- hgetall <hash>
- hdel <hash> <key>

sorted set

sorted set

- zadd <set> <score> <value>
- zrange <set> <start> <end>
- zrangebyscore <set> <start> <end>
- zrem <set> <value>

key data types

- string (counter, bitwise)
- list
- set
- hash
- sorted set (zset)

optimizations

- pub/sub
- transactions
- expiry and caching
- mass insertion tricks

optimizations

pub/sub

the concept

native

- publish <channel> <message>
- subscribe <channel>
- unsubscribe <channel>

- psubscribe <pattern>
- punsubscribe <pattern>

- psubscribe <pattern>
- punsubscribe <pattern>

Supported glob-style patterns:

- `h?llo` matches `hello`, `hallo` and `hxllo`
- `h*llo` matches `hllo` and `heeeello`
- `h[ae]llo` matches `hello` and `hallo`, but not `hillo`

Use \ to escape special characters if you want to match them verbatim.

limitations

transactions

- multi
- exec

- multi
- exec
- watch
- discard

why not lock?

expiry and caching

- expire <key> t(s)
- expireat <key> ts
- ttl <key>
- persist <key>

- pexpire <key> t(ms)
- pexpireat <key> ts (ms)
- pttl <key>

limitations

mass insertion tricks

transactions!

multiple commands

deployment

- replication
- persistence
- security

deployment

replication

master:slave

Example performance:
2.4 Ghz Core 2 Duo
7-8ms SUNIONSTORE
2x10k SETs -> 20k set

master:slave

master:
dir and dbfilename in
config writable by redis

slave:
slaveof host port

simple, but number of details

simple, but number of details

for example …

- asynchronous
- slave of slave
- non blocking

http://redis.io/topics/replication

2.8 enhancements

2.8 enhancements

- Require minimum number of slaves to write
- Partial resync

persistence

available!

imperfect

two options

snapshot

```
save 60 100
stop-writes-on-bgsave-error no
rdbcompression yes
dbfilename whatever.rdb
```

append only file

```
appendonly yes

appendfsync always | everysec | no

bgrewriteaof
```

picking a persistence strategy

http://redis.io/topics/persistence

security

there isn’t much!

Redis general security model

Redis is designed to be accessed by trusted clients inside trusted environments. This means that usually it is not a good idea to expose the Redis instance directly to the internet or, in general, to an environment where untrusted clients can directly access the Redis TCP port or UNIX socket.

For instance, in the common context of a web application implemented using Redis as a database, cache, or messaging system, the clients inside the front-end (web side) of the application will query Redis to generate pages or to perform operations requested or triggered by the web application user.

In this case, the web application mediates access between Redis and untrusted clients (the user browsers accessing the web application).

This is a specific example, but, in general, untrusted access to Redis should always be mediated by a layer implementing ACLs, validating user input, and deciding what operations to perform against the Redis instance.

In general, Redis is not optimized for maximum security but for maximum performance and simplicity.

http://redis.io/topics/security

Network security

Access to the Redis port should be denied to everybody but trusted clients in the network, so the servers running Redis should be directly accessible only by the computers implementing the application using Redis.

In the common case of a single computer directly exposed to the internet, such as a virtualized Linux instance (Linode, EC2, ...), the Redis port should be firewalled to prevent access from the outside. Clients will still be able to access Redis using the loopback interface.

http://redis.io/topics/security

Authentication feature

While Redis does not try to implement Access Control, it provides a tiny layer of authentication that is optionally turned on editing the **redis.conf** file.

When the authorization layer is enabled, Redis will refuse any query by unauthenticated clients. A client can authenticate itself by sending the **AUTH** command followed by the password.

The password is set by the system administrator in clear text inside the redis.conf file. It should be long enough to prevent brute force attacks for two reasons:

- Redis is very fast at serving queries. Many passwords per second can be tested by an external client.
- The Redis password is stored inside the **redis.conf** file and inside the client configuration, so it does not need to be remembered by the system administrator, and thus it can be very long.

http://redis.io/topics/security

there isn’t much

```
rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52
```

“sql” injection?

use cases

pub/sub?

chat server
activity feed
message queue?
...

simple keys and speed?

cache web pages
db row caching
authentication
shopping cart
...

lists?

log

...

counters?

analytics
page view counter
counter generator
...

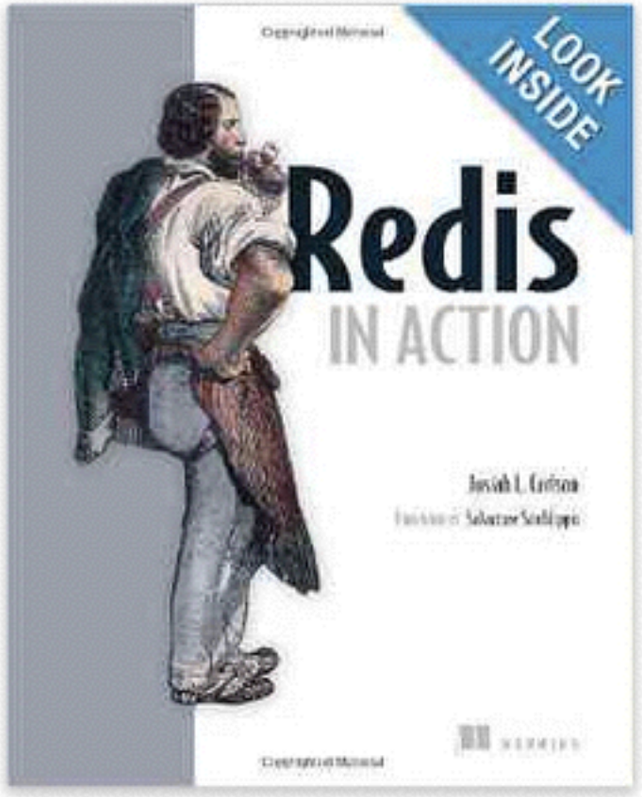
sets?

uniqueness test
tagging system
...

sorted sets?

set intersections?

resources



Redis in Action Paperback – June 25, 2013

by Josiah L. Carlson (Author)

★★★★☆ 5 customer reviews

Paperback
$29.14 Prime

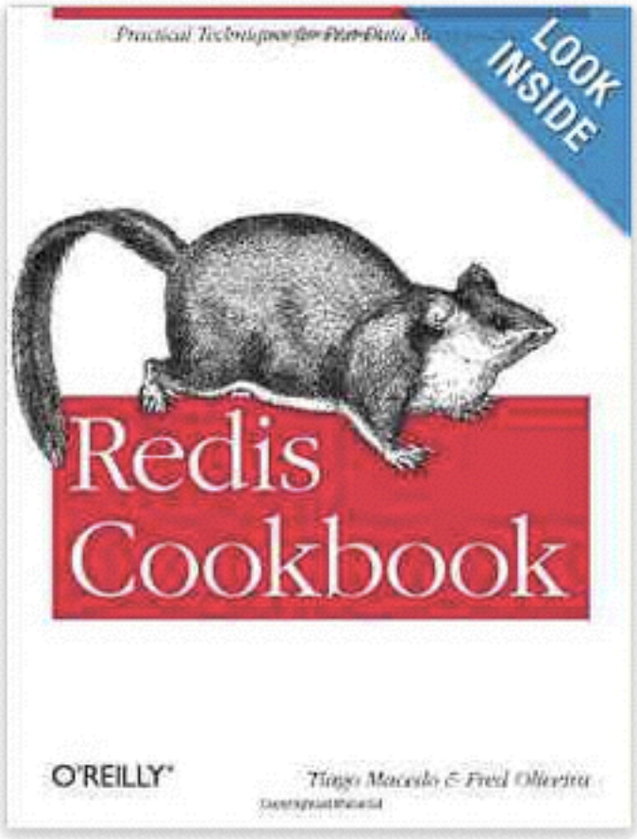
17 Used from $26.83
47 New from $22.94

Summary

Redis in Action introduces Redis and walks you through examples that demonstrate how to use it effectively. You'll

http://www.amazon.com/Redis-Action-Josiah-L-Carlson/dp/1617290858

resources



Redis Cookbook Paperback

by Tiago Macedo (Author), Fred Oliveira (Author)

4 customer reviews

See all 2 formats and editions

Kindle	Paperback
$10.49	$17.99 Prime

12 Used from $11.03
35 New from $11.82

Two years since its initial release, Redis already has an impressive list of adopters, including Engine Yard, GitHub,

http://www.amazon.com/Redis-Cookbook-Tiago-Macedo/dp/1449305040

resources

redis the swiss army knife of NoSQL

redis

@PeterBell