Redis: Data Cheeseburgers

Nick Quaranto

@qrush / nick@quaran.to
i work at thoughtbot.com
we use Redis
on

hoptoadapp.com
and

rubygems.org
“an advanced key-value store”
no

• tables
• relations
• documents
• graphs
yes

data structures
a big hash

_ => _
really it’s just

key => data structure
run commands

> GET foo

“bar”
memory

• all keys & values in RAM
• sync to disk when necessary
• various levels of paranoia
• data structures
• getting started
• use cases
data structures
strings

• like memcached
• chunk of data
• binary aware
> get fries (nil)

> set fries over OK

> get fries over
> strlen fries
4

> getrange fries 0 1
ov

> append fries over
done

over
overdone
> rename fries chips
OK

> del chips
1

> exists chips
0
counters

- like strings, cast to an integer
- atomic increment/decrement
- very, very fast
> get burgers
nil

> incr burgers
1

> incrby burgers 41
42
lists

• push, pop
• random access
• blocking actions
thanks to peter cooper (@peterc) for the graphs!
order

> rpush order burger
1
> rpush order hotdog
2
> rpush order fries
3
> lrange order 0 1
1. burger
2. hotdog

> lindex order 2
fries
> lrange order 0 -1
1. burger
2. hotdog
3. fries
order

> lpop order

burger
sets

- unique elements
- intersect, union, difference
sadd meat bacon
sadd meat turkey
sadd toppings bacon
sadd toppings bacon
meat
- burger
- turkey
- ham
- capicola
- jerky
- bacon

toppings
- relish
- bacon
- ketchup
- mustard
- pickles

thanks to peter cooper (@peterc) for the graphs!
> sinter meat toppings
1. bacon

meat

burger turkey ham capicola jerky

bacon

relish ketchup mustard pickles

toppings
sorted sets

• high score list
• set algebra
• ranges by score or rank
zadd menu 4.99 burger
zadd menu 2.99 shake
zadd menu 1.99 fries
### Menu

<table>
<thead>
<tr>
<th>Key</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fries</td>
<td>$1.99</td>
</tr>
<tr>
<td>Shake</td>
<td>$2.99</td>
</tr>
<tr>
<td>Burger</td>
<td>$4.99</td>
</tr>
</tbody>
</table>
### Menu

<table>
<thead>
<tr>
<th>Key</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>fries</td>
<td>1.99</td>
</tr>
<tr>
<td>shake</td>
<td>2.99</td>
</tr>
<tr>
<td>burger</td>
<td>4.99</td>
</tr>
</tbody>
</table>

```
> zrange menu 0 -1
1. fries
2. shake
3. burger
```
> zrank menu fries
  0

> zscore menu fries
  1.99
### Menu Items and Scores

<table>
<thead>
<tr>
<th>Key</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>fries</td>
<td>1.99</td>
</tr>
<tr>
<td>shake</td>
<td>2.99</td>
</tr>
<tr>
<td>burger</td>
<td>4.99</td>
</tr>
</tbody>
</table>

```
$ zrangebyscore menu 2 5

1. shake
2. burger
```
<table>
<thead>
<tr>
<th>key</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>fries</td>
<td>1.99</td>
</tr>
<tr>
<td>shake</td>
<td>2.99</td>
</tr>
<tr>
<td>burger</td>
<td>4.99</td>
</tr>
</tbody>
</table>

Deleted!

```
> zremrangebyscore menu 1.50 4.50
2
```
hashes

• easier to get all keys, values
• space saver
orders:1

nick    burger
john    fries
mike    shake
joe     salad
created_at  1298686121

> hset orders:1 nick burger
1
> hset orders:1 john fries
1

thanks to peter cooper (@peterc) for the graphs!
orders:1

<table>
<thead>
<tr>
<th>nick</th>
<th>burger</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td>fries</td>
</tr>
<tr>
<td>mike</td>
<td>shake</td>
</tr>
<tr>
<td>joe</td>
<td>salad</td>
</tr>
<tr>
<td>created_at</td>
<td>1298686121</td>
</tr>
</tbody>
</table>

> hget orders:1 mike
shake

> hlen orders:1
5

> hexists orders:1 brian
0
<table>
<thead>
<tr>
<th>Created At</th>
<th>Nick</th>
<th>Burger</th>
<th>Fries</th>
<th>Shake</th>
<th>Salad</th>
</tr>
</thead>
<tbody>
<tr>
<td>1298686121</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.</td>
</tr>
</tbody>
</table>

> hvals orders
1. burger
2. fries
3. shake
4. salad
5. 1298686121
getting started
community

BSD, on GitHub

Awesome maintainer (antirez)

IRC, wiki, mailing list

Hosting providers:
Redis TO GO, OpenRedis
installing

git clone git://github.com/antirez/redis

make

redis-server

or... $YOUR_DISTRO's package manager
sysadmin

simple telnet interface

no authorization/authentication by default

AUTH password

use appendonly & daily cron BGREWRITEAOF
performance

depends on system and configuration
ranges from 5,000 to 120,000 ops/sec
commands have $O(n)$ complexity listed
master/slave replication
diskstore soon
cluster support someday?
use cases
Whenever you want to store data fast that doesn't need to be 100% consistent.

-Mathias Meyer
http://www.paperplanes.de/2009/10/29/when_to_redis.html
hit counter

ideal with speed and INCR/INCRBY

total counts = counters

daily per URL = sorted set (ZINCRBY)
on a URL hit...

HTTP:
GET /index.html

Redis:
INCR index.html
ZINCRBY hits 1 index.html
basic stats

> get index.html
3910

> get pages/docs.html
2983

> get public/404.html
199
> zrange hits 0 -1 withscores
1) "public/404.html"
2) "199"
3) "pages/docs.html"
4) "2983"
5) "index.html"
6) "3910"
queue

atomic pops = multiple workers

start a job:
RPUSH key '{"some":"data"}'

worker daemon(s):
BLPOP key