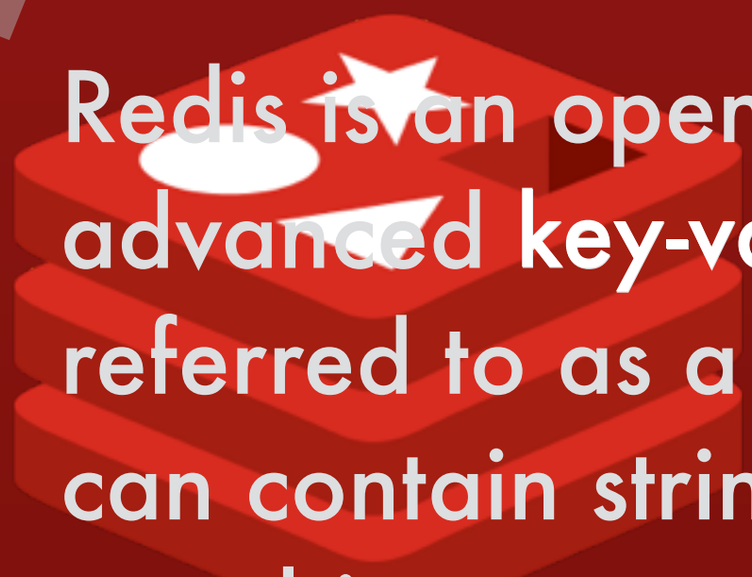
Session 302

# You Should Be Using Redis

John Hobbs

# PACK



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

redis.io

redis

# Why Redis?

redis

# Why Redis?

- Simple, small, well documented
- It speaks plain text
- It is fast
- Built in replication/clustering
- Transactions
- Lua Scripting
- Pub/Sub support
- Geospatial indexing (SOON!)

redis

# Get It

http://redis.io/download

```
$ wget http://download.redis.io/releases/redis-3.0.4.tar.gz
$ tar -zxf redis-3.0.4.tar.gz
$ cd redis-3.0.4/
$ make
$ cd src
$ ./redis-server
…snip…
8724:M 08 Sep 23:07:42.479 # Server started, Redis version 3.0.4
```

redis

# Caching

redis

# SET, GET & EXPIRE

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> SET HDC15 "Heartland Developer Conference 2015"
OK
127.0.0.1:6379> GET HDC15
"Heartland Developer Conference 2015"
127.0.0.1:6379> EXPIRE HDC15 5
(integer) 1
127.0.0.1:6379> GET HDC15
(nil)
127.0.0.1:6379>
```

redis

# TTL & PERSIST

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> SET HDC15 2015
OK
127.0.0.1:6379> EXPIRE HDC15 200
(integer) 1
127.0.0.1:6379> TTL HDC15
(integer) 194
127.0.0.1:6379> PERSIST HDC15
(integer) 1
127.0.0.1:6379> TTL HDC15
(integer) -1
127.0.0.1:6379>
```

redis

# MULTI & EXEC

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379> SET HDC15 "Heartland Developer Conference 2015"
QUEUED
127.0.0.1:6379> EXPIRE HDC15 5
QUEUED
127.0.0.1:6379> EXEC
1) OK
2) (integer) 1
127.0.0.1:6379>
```


# Write-Through Caching

redis

# Write-Through Caching

```
SELECT *
FROM `posts`
ORDER BY `created`
DESC LIMIT 10
```

redis

# Lists

- Ordered list of values
- Push & pop from head or tail
- Capable of arbitrary insertion

## Python 2 ➡ redis

```
list.append     ➡     RPUSH
list.count      ➡     LLEN
list.pop        ➡     RPOP
```

redis

# LPUSH, LRANGE & LTRIM

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> LPUSH posts 20
(integer) 20
127.0.0.1:6379> LRANGE posts 0 4
1) "20"
2) "19"
3) "18"
4) "17"
5) "16"
127.0.0.1:6379> LTRIM posts 0 2
OK
127.0.0.1:6379>
```

redis

# LREM & LLEN

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> LREM posts 0 18
(integer) 1
127.0.0.1:6379> LRANGE posts 0 4
1) "20"
2) "19"
3) "17"
4) "16"
5) "15"
127.0.0.1:6379> LLEN posts
(integer) 19
127.0.0.1:6379>
```

redis

# Write-Through Caching

Post

Dog Dog Dog

Area Breed Area Breed Area

redis

# Queues

redis

# Queues

**Ruby**
https://github.com/resque/resque

**Python**
http://python-rq.org/

**PHP**
https://github.com/chrisboulton/php-resque

**golang**
https://www.goworker.org/

redis

# Counting Stuff

redis

# INCR & DECR

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> SET downloads:2015:09:10 "11"
OK
127.0.0.1:6379> INCR downloads:2015:09:10
(integer) 12
127.0.0.1:6379> DECR downloads:2015:09:10
(integer) 11
127.0.0.1:6379> GET downloads:2015:09:10
"11"
127.0.0.1:6379>
```


# Counting Uniques

redis

# SADD, SCARD & SMEMBERS

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> SADD visitors:13:30 192.168.1.12
(integer) 1
127.0.0.1:6379> SADD visitors:13:30 192.168.1.37
(integer) 1
127.0.0.1:6379> SADD visitors:13:30 192.168.1.12
(integer) 0
127.0.0.1:6379> SCARD visitors:13:30
(integer) 2
127.0.0.1:6379> SMEMBERS visitors:13:30
1) "192.168.1.12"
2) "192.168.1.36"
127.0.0.1:6379>
```

redis

# SDIFF, SINTER & SUNION

**visitors:2015:09:09**

192.168.1.12
192.168.1.126

**visitors:2015:09:10**

192.168.1.12
192.168.1.37

redis

# SDIFF, SINTER & SUNION

```
127.0.0.1:6379> SUNIONSTORE visitors:2015:09 visitors:2015:09:10
visitors:2015:09:09
(integer) 3
127.0.0.1:6379> SCARD visitors:2015:09
(integer) 3
127.0.0.1:6379>
```


# Counting Uniques

(again)

redis

# PFADD & PFCOUNT

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> PFADD visitors:2015:09:10 192.168.1.12
(integer) 1
127.0.0.1:6379> PFADD visitors:2015:09:10 192.168.1.37
(integer) 1
127.0.0.1:6379> PFADD visitors:2015:09:10 192.168.1.12
(integer) 0
127.0.0.1:6379> PFCOUNT visitors:2015:09:10
(integer) 2
127.0.0.1:6379>
```

redis

# Leaderboards

redis

# ZADD, ZREVRANGE & ZREVRANK

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> ZADD leaderboard 5 "1"
(integer) 1
127.0.0.1:6379> ZADD leaderboard 50 "2"
(integer) 1
127.0.0.1:6379> ZADD leaderboard 100 "3"
(integer) 1
127.0.0.1:6379> ZREVRANGE leaderboard 0 5
1) "3"
2) "2"
3) "1"
127.0.0.1:6379>
```

redis

# ZADD, ZREVRANGE & ZREVRANK

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> ZREVRANGE leaderboard 0 "1"
1) "3"
2) "2"
127.0.0.1:6379> ZREVRANK "1"
(integer) 2
127.0.0.1:6379> ZREVRANGE leaderboard 1 1
1) "2"
127.0.0.1:6379>
```

redis

# Chat

redis

# SUBSCRIBE & PUBLISH

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> SUBSCRIBE chat
Reading messages...
1) "subscribe"
2) "chat"
3) (integer) 1
1) "message"
2) "chat"
3) "Hey!"
```

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> PUBLISH chat "Hey!"
(integer) 1
127.0.0.1:6379>
```


# Object Storage

redis

HMSET HGETALL

```
class User {
    name
    email
    password
}
```

redis

# save-user.lua

```lua
local new_user_id = redis.call("INCR", "users")
redis.call("HSET", "user:" .. new_user_id, "name", KEYS[1])
redis.call("HSET", "user:" .. new_user_id, "email", KEYS[2])
redis.call("HSET", "user:" .. new_user_id, "password", KEYS[3])
redis.call("ZADD", "user:emails", new_user_id, KEYS[2])
```

redis

# SCRIPT LOAD & EVALSHA

```
jmhobbs@venera:~ ✪ redis-cli SCRIPT LOAD "$(cat save-user.lua)"
“3cb690bebf59b2904446cbae41804ba607103dd3"
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> EVALSHA 3cb690bebf59b2904446cbae41804ba607103dd3
3 John john@velvetcache.org secret
(nil)
127.0.0.1:6379> KEYS user*
1) "user:1"
2) "users"
3) "user:emails"
127.0.0.1:6379>
```

redis

# Who's Nearby?

redis
(unstable)

# GEOADD, GEORADIUS & GEODIST

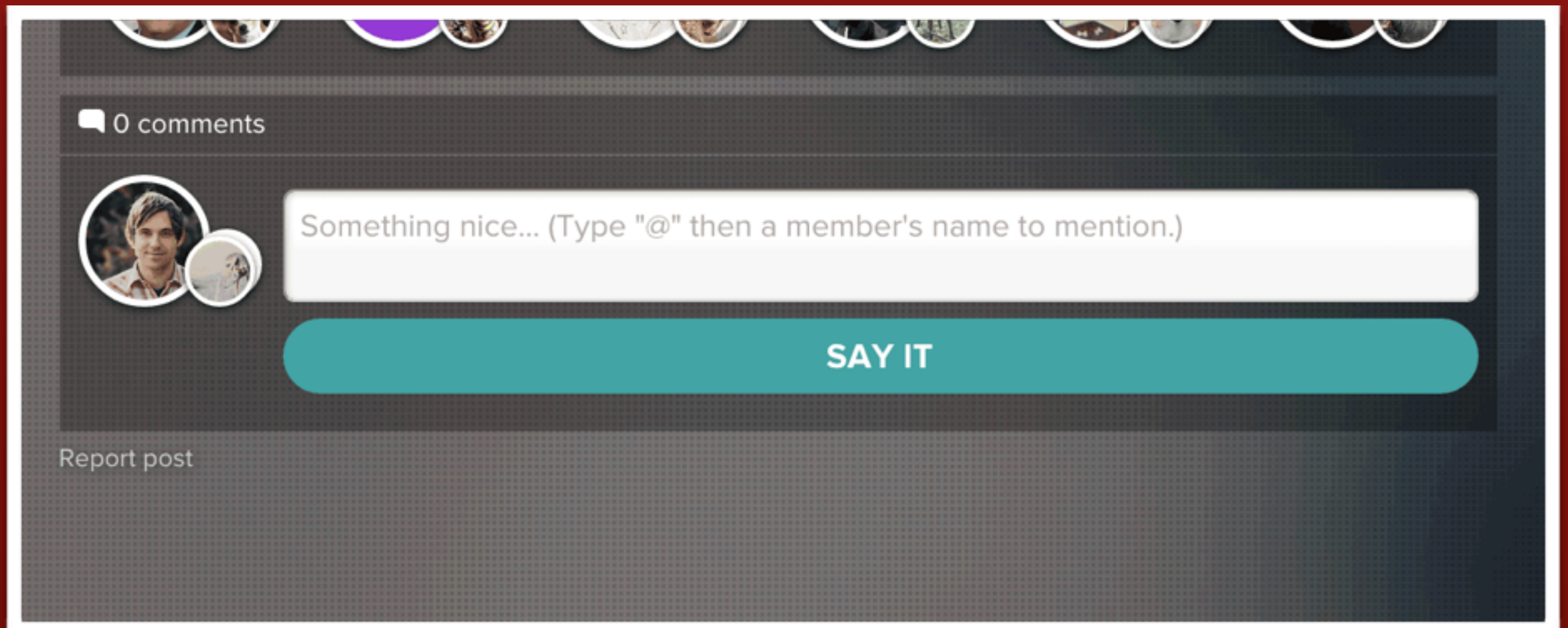
```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> GEOADD location -96.1074167 41.1812599 jmhobbs
(integer) 1
127.0.0.1:6379> GEOADD location -96.1055351 41.182195 alexpgates
(integer) 1
127.0.0.1:6379> GEORADIUSBYMEMBER location jmhobbs 5 mi
1) "jmhobbs"
2) "alexpgates"
127.0.0.1:6379> GEODIST location jmhobbs alexpgates mi
"0.11702141329908003"
127.0.0.1:6379> GEODIST location jmhobbs alexpgates ft
"617.8715265050572"
127.0.0.1:6379>
```

redis

# Autocomplete

redis

# Autocomplete



redis

# Autocomplete

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> ZADD names 0 J
(integer) 1
127.0.0.1:6379> ZADD names 0 Jo
(integer) 1
127.0.0.1:6379> ZADD names 0 Joe*
(integer) 1
127.0.0.1:6379>
```

redis

# Autocomplete

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> ZRANGE names 0 -1
 1) "B"
 2) "Bi"
 3) "Bil"
 4) "Bill*"
 5) "J"
 6) "Jo"
 7) "Joe*"
 8) "Joh"
 9) "John*"
10) “Jon"
etc…
```

redis

# Autocomplete

```
jmhobbs@venera:~ ✪ redis-cli
127.0.0.1:6379> ZRANK names Jo
(integer) 5
127.0.0.1:6379> ZRANGE names 6 -1
1) "Joe*"
2) "Joh"
3) "John*"
4) "Jon"
5) "Jona"
6) "Jonat"
7) "Jonath"
8) "Jonatha"
9) "Jonathan*"
```


# Replication

redis

# Replication

- One Master, one or more slaves
- Asynchronous
- Non-blocking (mostly)
- Read-only (mostly)
- Delivery to slaves not guaranteed
- You should turn disk persistence on Master
- SLAVEOF, SYNC

redis

# Cluster

redis

# Cluster

- Cluster communication is out of band
- Some caveats on multiple key operations
- Master slave model for failover
- Not strongly consistent
- Synchronous writes are possible with WAIT

redis

# Sentinel

redis

# Sentinel

- Monitors master and slaves
- Notifications to sysadmin or other programs
- Automatic failover of master
- Provides configuration for nodes
- Is a distributed system itself

redis

# Who uses redis?

redis

# Who uses redis?

- Instagram
- Weibo
- Twitter
- Tumblr

redis

# Thank You

redis

# Thank You

http://velvetcache.org

http://twitter.com/jmhobbs

john@velvetcache.org