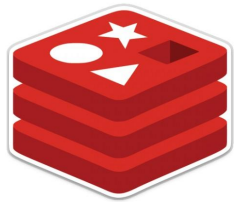
Redis



Presentation by Atreyee Maiti

What is redis?

- an in-memory key-value store, with persistence
- open source
- written in C
- “can handle up to 2^{32} keys, and was tested in practice to handle at least 250 million of keys per instance.” - *http://redis.io/topics/faq*
- most popular key-value store - *http://db-engines.com/en/ranking*

History:

- REmote DIctionary Server
- released in March 2009
- built in order to scale http://lloogg.com/

Features

- abstract data types - Lists, Sets, Sorted sets of strings (collections of non-repeating elements ordered by a floating-point number called score), Hashes
- two kinds of persistence mechanisms
- replication
- publish subscribe
- transactions (with optimistic locking)
- Lua scripting
- command line tool

Redis protocol

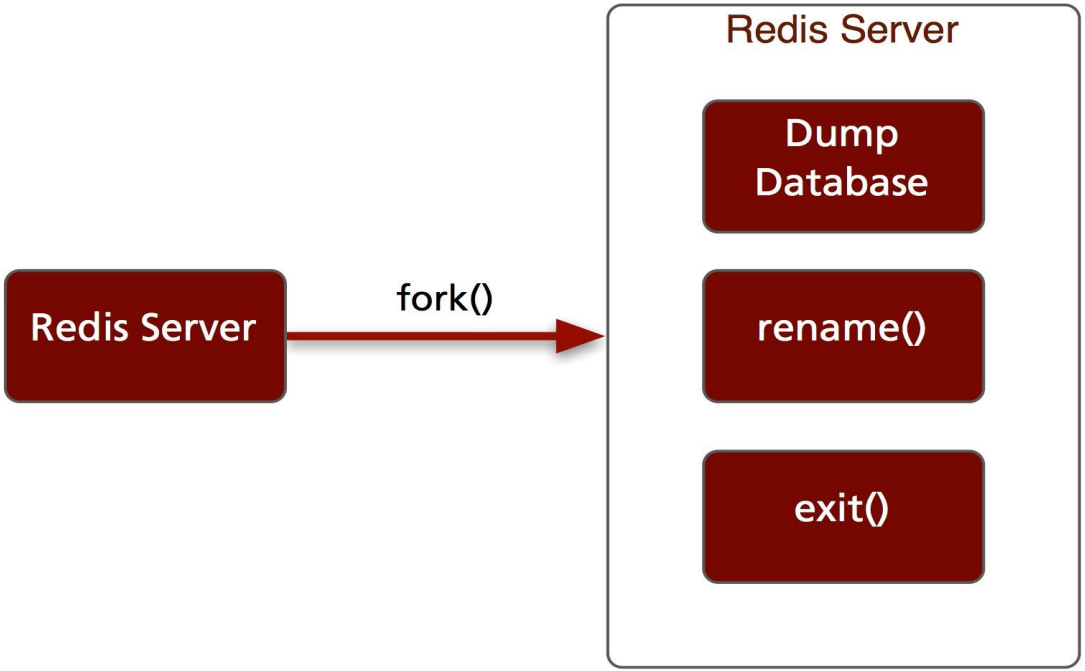
```
new-host-3:redis-2.8.0 atreyeemaiti$ src/redis-cli
127.0.0.1:6379> keys "*"
(empty list or set)
127.0.0.1:6379> set foo 45
OK
127.0.0.1:6379> get foo
"45"
127.0.0.1:6379> keys "*"
1) "foo"
127.0.0.1:6379>
```

Persistence mechanism

RDB(Redis snapshotting) and AOF(Append-only file)

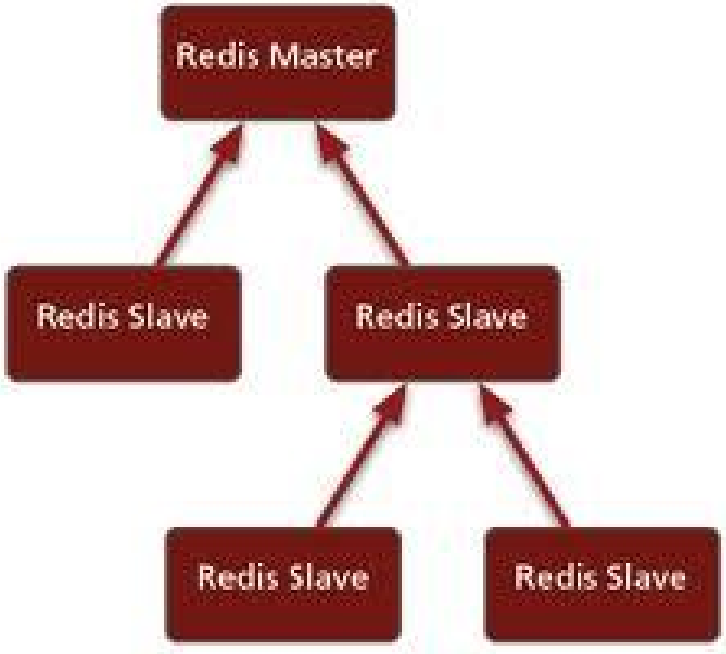
- The RDB persistence performs point-in-time snapshots of dataset at specified intervals.
- The AOF persistence logs every write operation received by the server, that will be played again at server startup, reconstructing the original dataset. Commands are logged using the same format as the Redis protocol itself, in an append-only fashion.

Copy on write via forking



http://nosqlberlin.de/slides/NoSQLBerlin-Redis.pdf

Replication



- redis master - handles all reads/ writes
- slave - hot standby - can also be configured for scalability to serve reads
- sentinels - mechanism for monitoring, failover - uses pub/sub, gossip and agreement protocols

Transactions

```
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> set foo 100
OK
127.0.0.1:6379> set bar 250
OK
127.0.0.1:6379> multi
OK
127.0.0.1:6379> incr foo
QUEUED
127.0.0.1:6379> decr bar
QUEUED
127.0.0.1:6379> exec
1) (integer) 101
2) (integer) 249
127.0.0.1:6379>
```

To redis or not?

- Counting Downloads
- High Score tables
- Caching
- Queues
- coupling with other databases

My two cents :)

- using resque for managing background jobs
- jobs placed on queues
- used sentinels for HA

So who is using redis?

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

USP

- in-memory with persistence
- supported data types
- pub/sub
- queues
- support for a whole lot of clients

	Redis	Memcached
In Memory	X	X
Virtual Memory	Deprecated	
Persistent	X	
Atomic	X	X
Consistent	X	X
Replication	X	
Authentication	X	???
Key / Value	X	X
Key Enumeration	X	
Key / Value "buckets"	X	
Data Structures	X	
Channel Pub/Sub	X	
Memory usage	10-20% Smaller	
Speed		Slightly Faster

Source: http://redis4you.com/articles.php?id=003

Comparison

Redis	**Mongodb**	**Couchdb**
Stock prices. Analytics. Real-time data collection. Real-time communication. And wherever you used memcached before.	For most things that you would do with MySQL or PostgreSQL, but having predefined columns really holds you back.	CRM, CMS systems. For accumulating, occasionally changing data, on which pre-defined queries are to be run. Places where versioning is important.

References

http://nosqlberlin.de/slides/NoSQLBerlin-Redis.pdf

http://stackoverflow.com/questions/7888880/what-is-redis-and-what-do-i-use-it-for

http://highscalability.com/blog/2011/7/6/11-common-web-use-cases-solved-in-redis.html

http://openmymind.net/redis.pdf

http://oldblog.antirez.com/post/redis-persistence-demystified.html

http://redis4you.com/articles.php?id=003

http://blog.siyelo.com/redis-in-the-nosql-ecosystem

http://www.slideshare.net/tim.lossen.de/cassandra-vs-redis