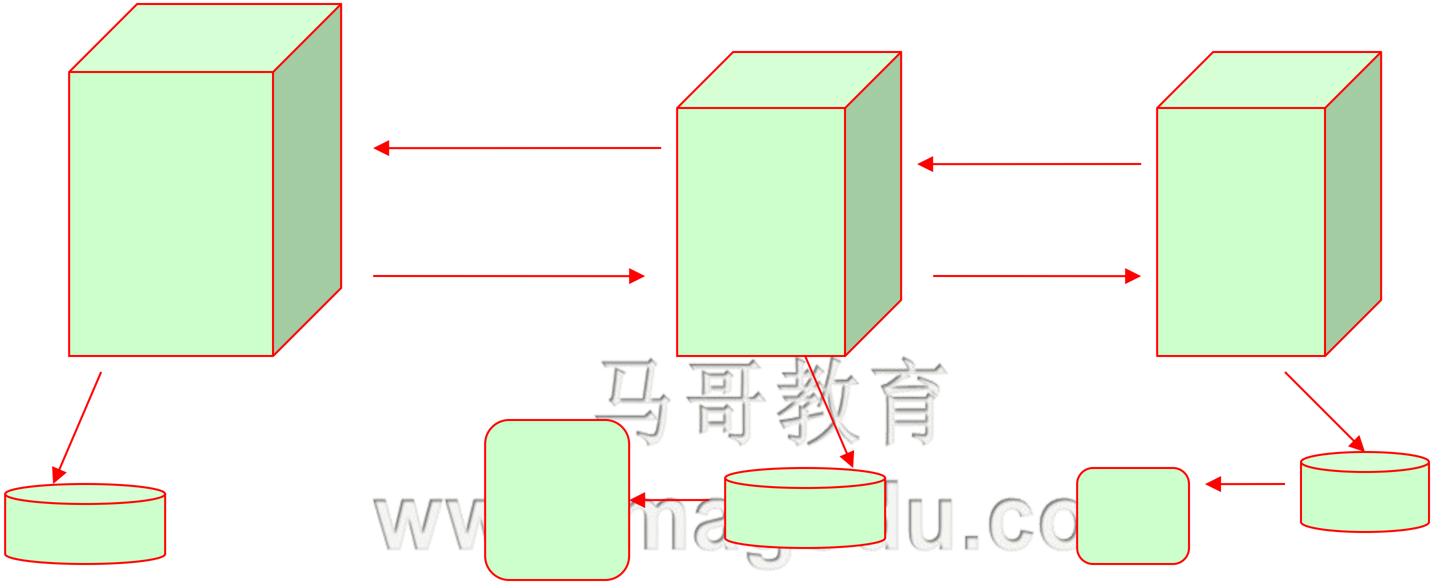
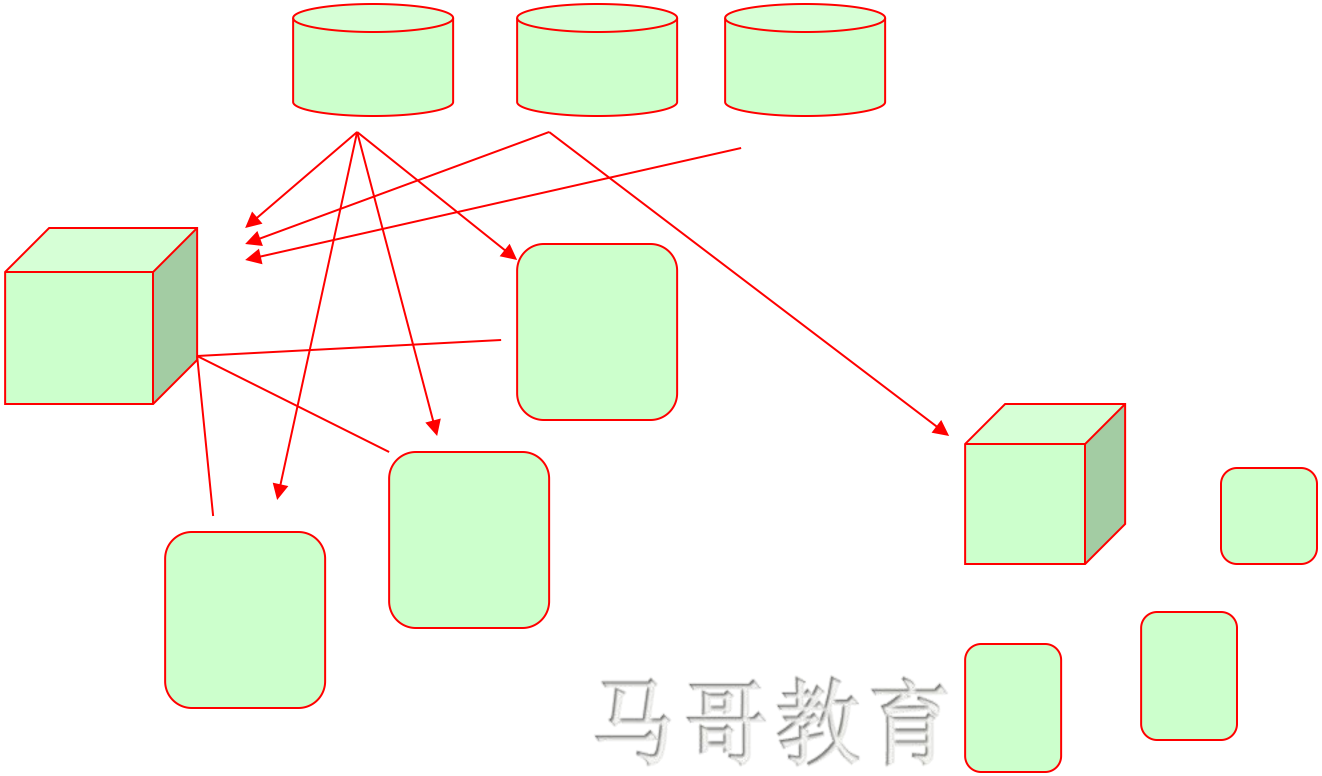
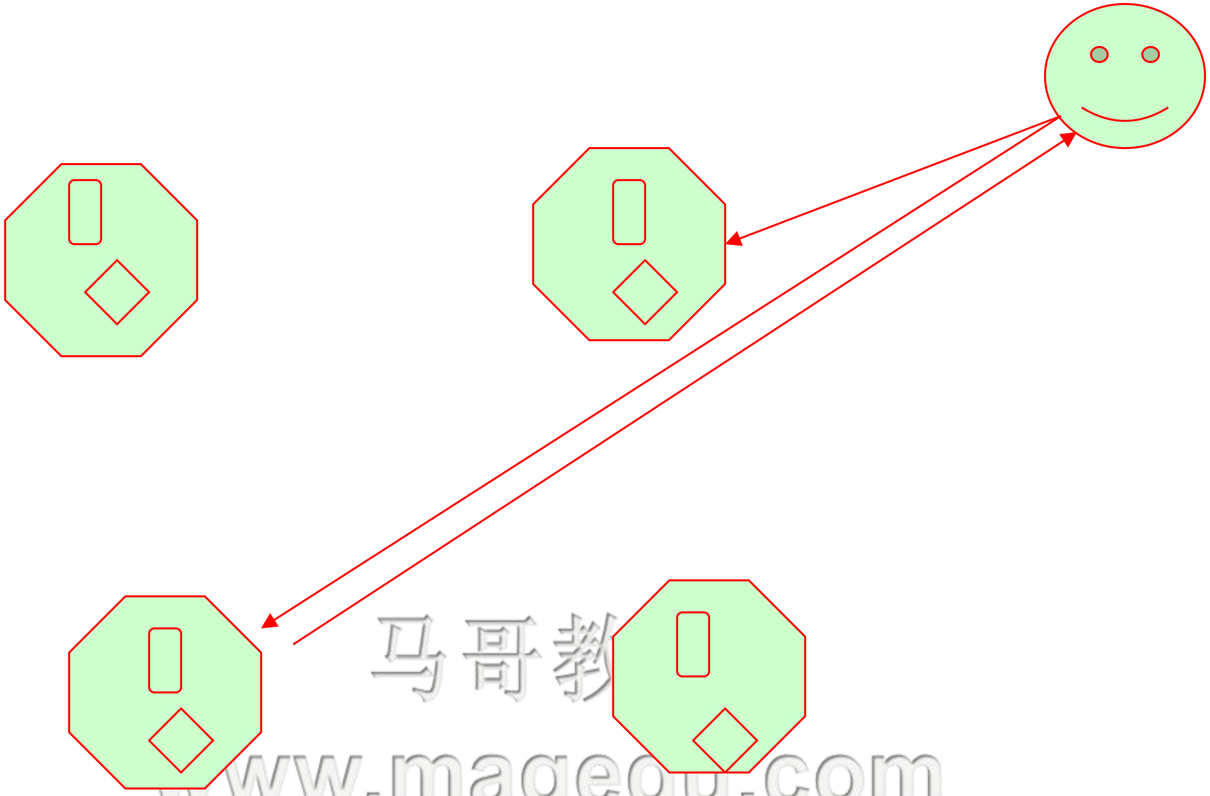
马哥教育

redis

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com







Overview of Redis

- The word Redis means REmote DIctionary Server
- Initial release in 2009
- It is an advanced key-value store or a data structure store
- Runs entirely in memory
  - All data is kept in memory
  - Quick data access since it is maintained in memory
  - Data can be backed up to disk periodically
  - Single threaded server
- Extensible via Lua scripts
- Able to replicate data between servers
- Clustering also available

About Redis

“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

- Redis is an in-memory but persistent on disk database
- 1 Million small Key -> String value pairs use ~ 100 MB of memory
- Single threaded – but CPU should not be the bottleneck
  - Average Linux system can deliver even 500k requests per second
- Limit is likely the available memory in your system
  - max. 232 keys

Persistence

- Snapshotting
  - Data is asynchronously transferred from memory to disk
- AOF (Append Only File)
  - Each modifying operation is written to a file
  - Can recreate data store by replaying operations
  - Without interrupting service, will rebuild AOF as the shortest sequence of commands needed to rebuild the current dataset in memory

Replication

- Redis supports master-slave replication
- Master-slave replication can be chained
- Be careful:
  - Slaves are writeable!
  - Potential for data inconsistency
- Fully compatible with Pub/Sub features

Differences to Memcached

- Memcached is a “distributed memory object caching system”
- Redis persists data to disk eventually
- Memcached is an LRU cache
- Redis has different data types and more features
- Memcached is multithreaded
- Similar speed

- Redis的优势
  - 丰富的(资料形态)操作
    - Hashs, Lists, Sets, Sorted Sets, HyperLogLog 等
  - 内建replication及cluster
  - 就地更新(in-place update)操作
  - 支援持久化(磁盘)
    - 避免雪崩效应
- Memcached的优势
  - 多线程
    - 善用多核CPU
    - 更少的阻塞操作
  - 更少的内存开销
  - 更少的内存分配压力
  - 可能有更少的内存碎片

Prominent Adopters

- Twitter
- Pinterest
- Tumblr
- GitHub
- Stack Overflow
- digg
- Blizard
- flickr
- WeiBo
- ……

Redis 3.0

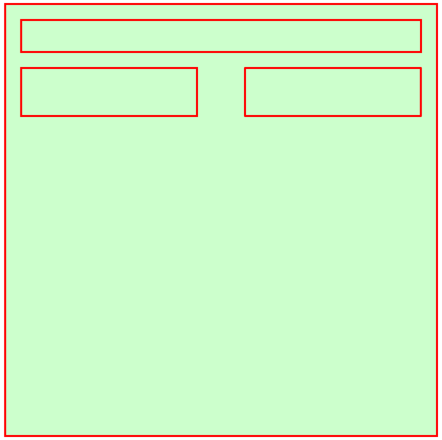
- 2015年4月1日正式推出
  - Redis Cluster
  - 新的 " embedded string"
  - LRU演算法的改进
    - 预设随机取5个样本，插入并排序至一个pool，移除最佳者，如此反复，直到内存用量小于maxmemory的设定
    - 樣本5比先前的3多
    - 从局部最优趋向全局最优

Redis特性

- RDBMS
  - Oracle, DB2, PostgreSQL, MySQL, SQL Server, ...
- NoSQL
  - Cassandra, HBase, Memcached, MongoDB, Redis, ...
- NewSQL
  - Aerospike, FoundationDB, RethinkDB, ...

Redis特性

- Key-value NoSQL
  - Memcached, Redis, ...
- Column family NoSQL
  - Cassandra, HBase, ...
- Documentation NoSQL
  - MongoDB, ...
- Graph NoSQL
  - Neo4j, ...



Commands

- redis-server
- redis-cli
  - Command line interface
- redis-benchmark
  - Benchmarking utility
- redis-check-dump & redis-check-aof
  - Corrupted RDB/AOF files utilities

Overview

- Family of fundamental data structures
  - Strings and string containers
  - Accessed / indexed by key
  - Directly exposed — No abstraction layers
- Rich set of atomic operations over the structures
  - Detailed reference using big-O notation for complexities
- Basic publish / subscribe infrastructure

Keys

- Arbitrary ASCII strings
  - Define some format convention and adhere to it
  - Key length matters!
- Multiple name spaces are available
  - Separate DBs indexed by an integer value
    - SELECT command
    - Multiples DBs vs. Single DB + key prefixes
- Keys can expire automatically

Data structures

- Strings
  - Caching, counters, realtime metrics…
- Hashes
  - “Object” storage…
- Lists
  - Logs, queues, message passing…
- Sets
  - Membership, tracking…
- Ordered sets
  - Leaderboards, activity feeds…

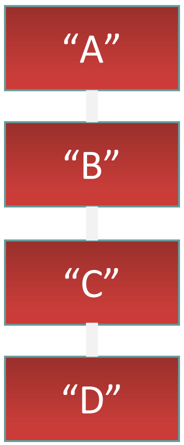
Strings

- help @string
  - SET
  - GET
  - EXISTS

- Integers
  - DECR
  - INCR

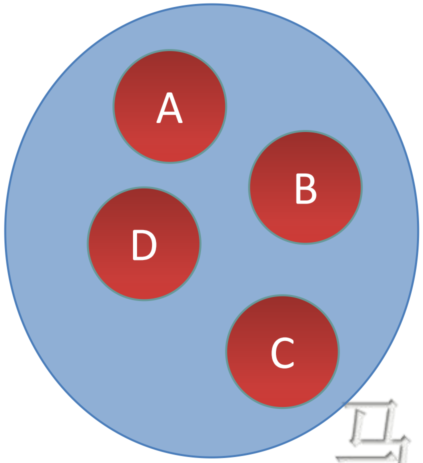
Advanced Data Structures

List



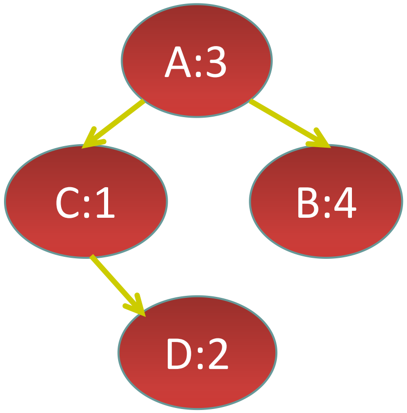
[A, B, C, D]

Set



{A, B, C, D}

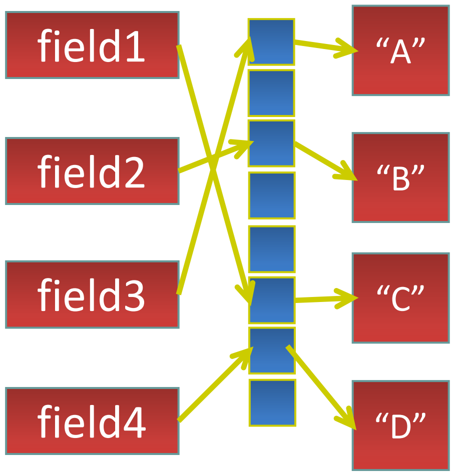
Sorted Set



{value:score}

{C:1, D:2, A:3, B:4}

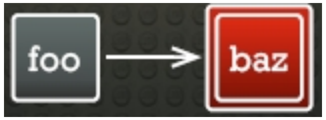
Hash



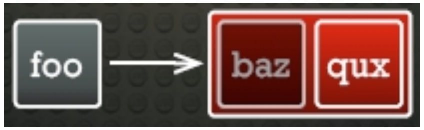
{key:value}

{field1:"A", field2:"B"…}

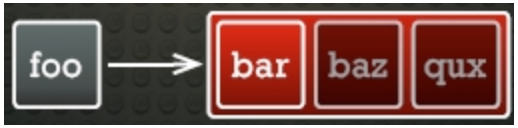
Lists



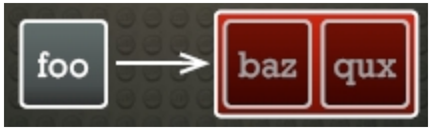
- help @list
  - RPUSH



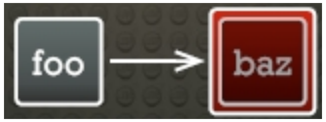
  - LPUSH



  - LPOP

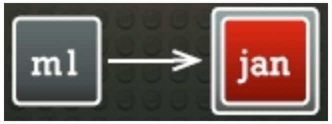


  - RPOP

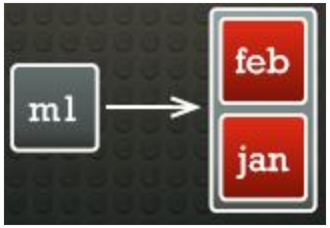


Sets

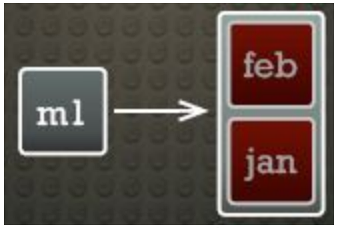
- help @set



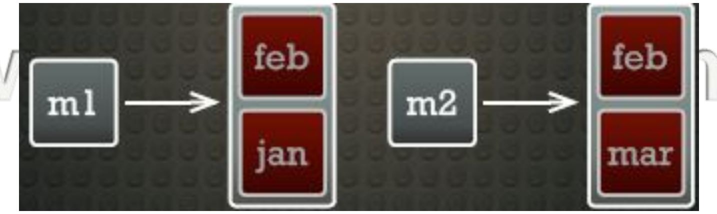
- SADD

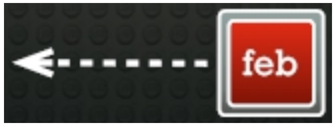


- SMEMBERS

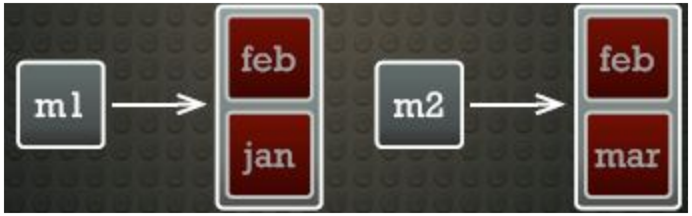


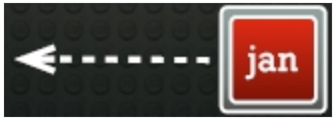
- SINTER



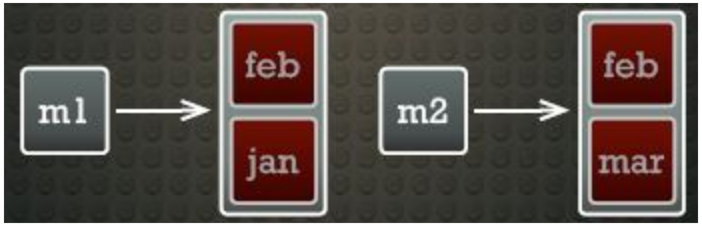


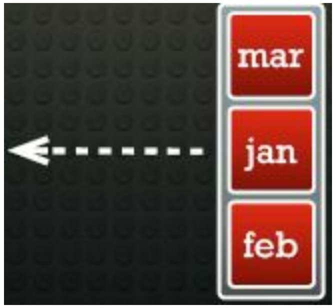
- help @set
  - SDIFF





  - SUNION

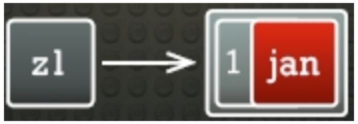




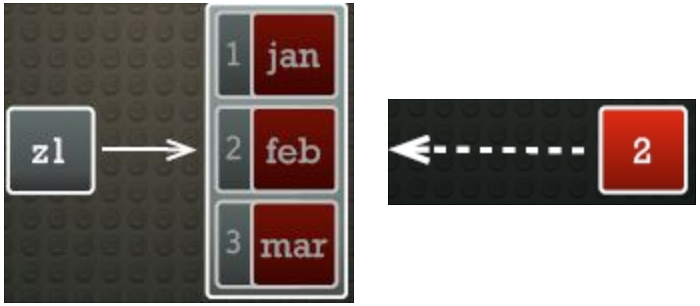
  - SISMEMER

Sorted Sets

- help @sorted_set
  - ZADD



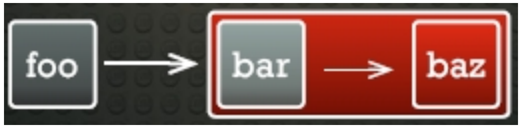
  - ZSCORE



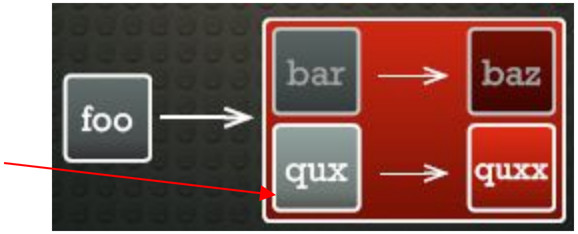
  - ZRANGE
  - ZRANGEBYSCORE

Hashes

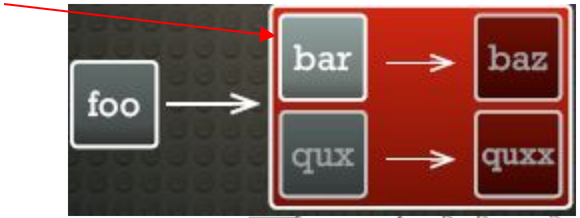
- help @hash

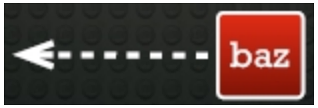


- ➲ HSET

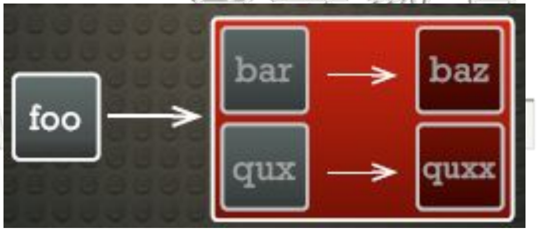


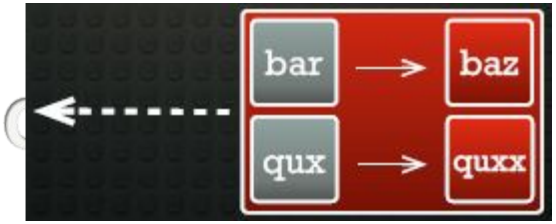
- ➲ HGET





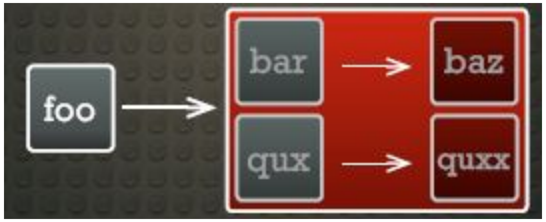
- ➲ HGETALL

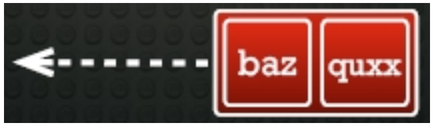




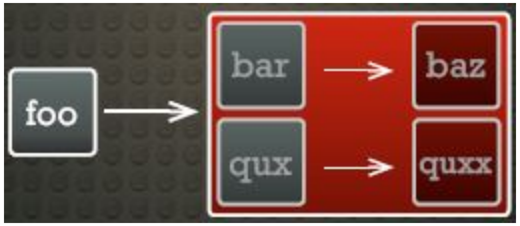
Hashes

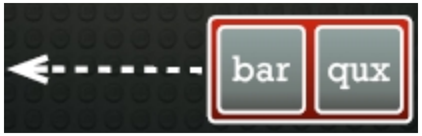
- help @hash
  - HVALS





  - HKEYS





Publish / Subscribe

- Classic pattern decoupling publishers & subscribers
  - You can subscribe to channels; when someone publish in a channel matching your interests Redis will send it to you
  - SUBSCRIBE, UNSUBSCRIBE & PUBLISH commands
- Fire and forget notifications
  - Not suitable for reliable off-line notification of events
- Pattern-matching subscriptions
  - PSUBSCRIBE & PUNSUBSCRIBE commands

Publish / Subscribe

- Available since Redis 2.8
  - Disabled in the default configuration
  - Key-space vs. keys-event notifications
- Delay of key expiration events
  - Expired events are generated when Redis deletes the key; not when the TTL is consumed
    - Lazy (i.e. on access time) key eviction
    - Background key eviction process

Pipelining

- Redis pipelines are just a RTT optimization
  - Deliver multiple commands together without waiting for replies
  - Fetch all replies in a single step
    - Server needs to buffer all replies!
- Pipelines are NOT transactional or atomic
- Redis scripting FTW!
  - Much more flexible alternative

Transactions

- Or, more precisely, "transactions"
  - Commands are executed as an atomic & single isolated operation
    - Partial execution is possible due to pre/post EXEC failures!
  - Rollback is not supported!
- MULTI, EXEC & DISCARD commands
  - Conditional EXEC with WATCH
- Redis scripting FTW!
  - Redis transactions are complex and cumbersome

Scripting

- Added in Redis 2.6
- Uses the LUA 5.1 programming language▸
  - Base, Table, String, Math & Debug libraries
  - Built-in support for JSON and MessagePack
  - No global variables
  - redis.{call(), pcall()}
  - redis.{error_reply(), status_reply(), log()}

Scripting

- Scripts are atomic, like any other command
- Scripts add minimal overhead
  - Single thread ⇒ Shared LUA context
- Scripts are replicated on slaves by sending the script (i.e. not the resulting commands)
  - Scripts are required to be pure functions
  - Maximum execution time vs. Atomic execution

马哥教育

Mastering Redis

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

Persistence

- The whole dataset needs to feet in memory
  - Durability is optional
  - Very high read & write rates
  - Optimal & simple memory and disk representations
- What if Redis runs out of memory?
  - Swapping ⇒ Performance degradation
  - Hit maxmemory limit ⇒ Failed writes or eviction policy

Snapshotting — RDB

- Periodic asynchronous point-in-time dump to disk
  - Every S seconds and C changes
  - Fast service restarts
- Possible data lost during a crash
- Compact files
- Minimal overhead during operation
- Huge data sets may experience short delays during fork()
- Copy-on-write fork() semantics ⇒ 2x memory problem

Append only file — AOF

- Journal file logging every write operation
  - Configurable fsync frequency: speed vs. safety
  - Commands replayed when server restarts
- No as compact as RDB
  - Safe background AOF file rewrite fork()
- Overhead during operation depends on fsync behavior
- Recommended to use both RDB + AOF
  - RDB is the way to of for backups & disaster recovery

Security

- Designed for trusted clients in trusted environments
  - No users, no access control, no connection filtering…
- Basic unencrypted AUTH command
  - requirepass s3cr3t
- Command renaming
  - rename-command FLUSHALL f1u5hc0mm4nd
  - rename-command FLUSHALL ""

Replication Overview

- One master — Multiple slaves
  - Scalability & redundancy
    - Client side failover, eviction, query routing…
  - Lightweight master
- Slaves are able to accept other slave connections
- Non-blocking in the master, but blocking on the slaves
- Asynchronous but periodically acknowledged

Replication Overview

- Automatic slave reconnection
- Partial resynchronization: PSYNC vs. SYNC
  - RDB snapshots are used during initial SYNC
- Read-write slaves
  - slave-read-only no
  - Ephemeral data storage
- Minimum replication factor

Replication

- Some commands & configuration
  - Trivial setup
    - slaveof <host> <port>
    - SLAVEOF [<host> <port >| NO ONE]
  - Some more configuration tips
    - slave-serve-stale-data [yes|no]
    - repl-ping-slave-period <seconds>
    - masterauth <password>
  - Inconsistencies are possible when using some eviction policy in a replicated setup
    - Set slave's maxmemory to 0

Performance

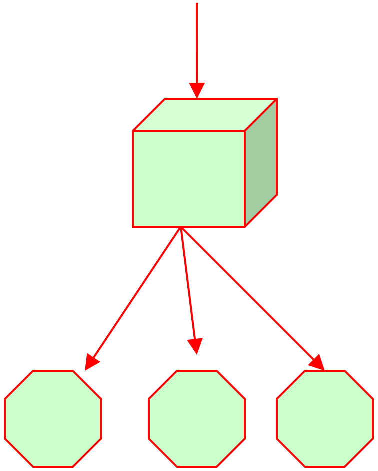
- Fast CPUs with large caches and not many cores
- Do not invest on expensive fast memory modules
- Avoid virtual machines
- Use UNIX domain sockets when possible
- Aggregate commands when possible
- Keep low the number of client connections

Avalibility

- Twemproxy (Twitter)
- Codis (豌豆荚)
- Redis Cluster (官方)
- Cerberus (芒果TV)

Avalibility

- Twemproxy (Twitter)
  - 代理分片机制
  - 优点
    - 非常稳定，企业级方案
  - 缺点
    - 单点故障
    - 需依赖第三方软件，如Keepalived
    - 无法平滑地横向扩展
    - 沒有后台界面
    - 代理分片机制引入更多的来回次数并提高延迟
    - 单核模式，无法充份利用多核，除非多实例
    - Twitter官方内部不再继续使用Twemproxy



Avalibility

- Codis (豌豆荚)
  - 代理分片机制
  - 2014年11月开源
  - 基于Go以及C语言开发
  - 优点
    - 非常穩定，企业级方案
    - 数据自动平衡
    - 高性能
    - 简单的测试显示较Twemproxy快一倍
    - 善用多核CPU
    - 简单
      - 沒有Paxos类的协调机制
      - 沒有主从复制
    - 有后台界面
  - 缺点
    - 代理分片机制引入更多的来回次数并提高延迟
    - 需要第三方软件支持协调机制
      - 目前支持Zookeeper及Etcd
    - 不支持主从复制，需要另外实现
    - Codis采用了Proxy的方案，所以必然会带来单机性能的损失
      - 经测试，在不开pipeline的情况下，大概会损失40%左右的性能

Avalibility

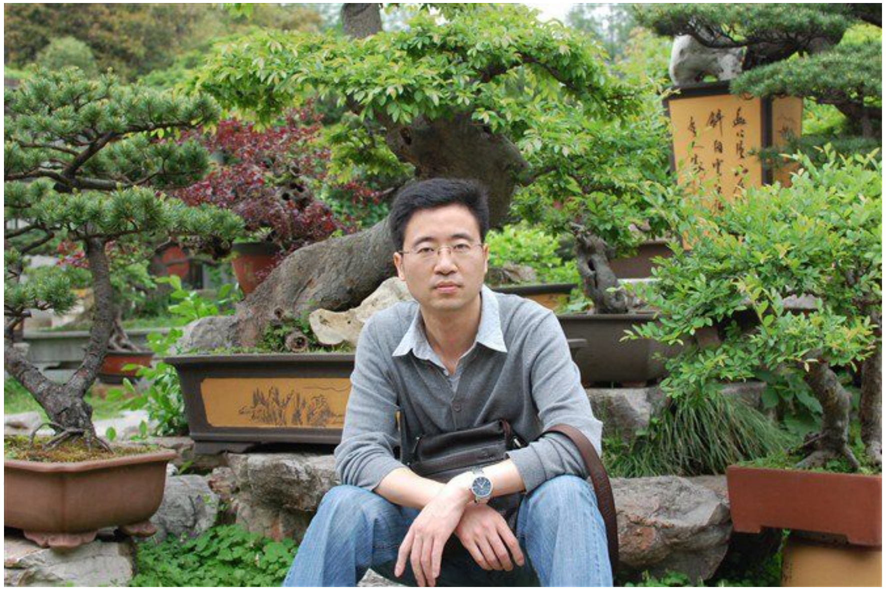
- Redis Cluster (官方)
  - 官方实现
  - 需要Redis 3.0或更高版本
  - 优点
    - 无中心的P2P Gossip分散式模式
    - 更少的來回次数并降低延迟
    - 自动于多个Redis节点进行分片
    - 不需要第三方软件支持协调机制
  - 缺点
    - 依赖于Redis 3.0或更高版本
    - 需要时间验证其稳定性
    - 沒有后台界面
    - 需要智能客戶端
    - Redis客戶端必须支持Redis Cluster架构
    - 较Codis有更多的维护升级成本

Avalibility

- Cerberus (芒果TV)
  - 优点
    - 数据自动平衡
    - 本身实现了Redis的Smart Client
    - 支持读写分离
  - 缺点
    - 依赖Redis 3.0或更高版本
    - 代理分片机制引入更多的來回次数并增大延迟
    - 需要时间验证其稳定性
    - 没有后台界面

关于马哥教育

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：1661815153，113228115
- QQ群：203585050，279599283



马哥教育

Thank You!