redis

介绍：Redis 是一个开源的使用 ANSI C 语言编写、遵守 BSD 协议、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API的非关系型数据库。

传统数据库遵循 ACID 规则。而 Nosql（Not Only SQL 的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称） 一般为分布式而分布式一般遵循 CAP 定理。

**Redis支持的数据类型？**

String字符串：

格式: set key value

string类型是二进制安全的。意思是redis的string可以包含任何数据。比如jpg图片或者序列化的对象 。

string类型是Redis最基本的数据类型，一个键最大能存储512MB。

保存

如果设置的键不存在则为添加，如果设置的键已经存在则修改

**设置键值**

set key value

例1：设置键为name值为itcast的数据

set name zhangsan

设置键值及过期时间，以秒为单位

setex key seconds value

例2：设置键为aa值为aa过期时间为3秒的数据

setex aa 3 aa

设置多个键值

mset key1 value1 key2 value2 ...

例3：设置键为'a1'值为'python'、键为'a2'值为'java'、键为'a3'值为'c'

mset a1 python a2 java a3 c

追加值

append key value

例4：向键为a1中追加值' haha'

append 'a1' 'haha'

**获取：根据键获取值，如果不存在此键则返回nil**

get key

例5：获取键'name'的值

get 'name'

根据多个键获取多个值

mget key1 key2 ...

例6：获取键a1、a2、a3'的值

mget a1 a2 a3

查找键，参数⽀持正则表达式

keys pattern

例1：查看所有键

keys \*

例2：查看名称中包含a的键

keys 'a\*'

判断键是否存在，如果存在返回1，不存在返回0

exists key1

例3：判断键a1是否存在

exists a1

查看键对应的value的类型

type key

例4：查看键a1的值类型，为redis⽀持的五种类型中的⼀种

type a1

**删除键及对应的值**

del key1 key2 ...

例5：删除键a2、a3

del a2 a3

设置过期时间，以秒为单位

如果没有指定过期时间则⼀直存在，直到使⽤DEL移除

expire key seconds

例6：设置键'a1'的过期时间为3秒

expire 'a1' 3

查看有效时间，以秒为单位

ttl key

例7：查看键'bb'的有效时间

ttl bb

Hash（哈希）

格式: hmset name  key1 value1 key2 value2

Redis hash 是一个键值(key=>value)对集合。

Redis hash是一个string类型的field和value的映射表，hash特别适合用于存储对象。

设置多个属性

hmset key field1 value1 field2 value2 ...

例2：设置键u2的属性name为itcast、属性age为11

hmset u2 name itcast age 11

获取 获取指定键所有的属性

hkeys key

例3：获取键u2的所有属性

hkeys u2

获取⼀个属性的值

hget key field

例4：获取键u2属性'name'的值

hget u2 'name'

获取多个属性的值

hmget key field1 field2 ...

例5：获取键u2属性'name'、'age的值

hmget u2 name age

获取所有属性的值

hvals key

例6：获取键'u2'所有属性的值

hvals u2

删除 删除整个hash键及值，使⽤del命令 删除属性，属性对应的值会被⼀起删除

hdel key field1 field2 ...

例7：删除键'u2'的属性'age'

hdel u2 age

List（列表）

Redis 列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）

格式: lpush  name  value

在 key 对应 list 的头部添加字符串元素

格式: rpush  name  value

在 key 对应 list 的尾部添加字符串元素

格式: lrem name  index

key 对应 list 中删除 count 个和 value 相同的元素

格式: llen name

返回 key 对应 list 的长度

lpush key value1 value2 ...

例1：从键为'a1'的列表左侧加⼊数据a 、 b 、c

lpush a1 a b c

在右侧插⼊数据

rpush key value1 value2 ...

例2：从键为'a1'的列表右侧加⼊数据0 1

rpush a1 0 1

在指定元素的前或后插⼊新元素

linsert key before或after 现有元素 新元素

例3：在键为'a1'的列表中元素'b'前加⼊'3'

linsert a1 before b 3

获取 返回列表⾥指定范围内的元素

start、stop为元素的下标索引

索引从左侧开始，第⼀个元素为0

索引可以是负数，表示从尾部开始计数，如-1表示最后⼀个元素

lrange key start stop

例4：获取键为'a1'的列表所有元素

lrange a1 0 -1

设置指定索引位置的元素值

索引从左侧开始，第⼀个元素为0

索引可以是负数，表示尾部开始计数，如-1表示最后⼀个元素

lset key index value

例5：修改键为'a1'的列表中下标为1的元素值为'z'

lset a 1 z

删除 删除指定元素

将列表中前count次出现的值为value的元素移除

count > 0: 从头往尾移除

count < 0: 从尾往头移除

count = 0: 移除所有

lrem key count value

例6.1：向列表'a2'中加⼊元素'a'、'b'、'a'、'b'、'a'、'b'

lpush a2 a b a b a b

例6.2：从'a2'列表右侧开始删除2个'b'

lrem a2 -2 b

例6.3：查看列表'a2'的所有元素

lrange a2 0 -1

Set（集合）

格式: sadd  name  value

Redis的Set是string类型的无序集合。

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是O(1)。

元素具有唯⼀性，不重复

说明：对于集合没有修改操作

增加

添加元素

sadd key member1 member2 ...

例1：向键'a3'的集合中添加元素'zhangsan'、'lisi'、'wangwu'

sadd a3 zhangsan sili wangwu

获取

返回所有的元素

smembers key

例2：获取键'a3'的集合中所有元素

smembers a3

删除

删除指定元素

srem key

例3：删除键'a3'的集合中元素'wangwu'

srem a3 wangwu

zset(sorted set：有序集合)

格式: zadd  name score value

Redis zset 和 set 一样也是string类型元素的集合,且不允许重复的成员。

不同的是每个元素都会关联一个double类型的分数。redis正是通过分数来为集合中的成员进行从小到大的排序。

zset的成员是唯一的,但分数(score)却可以重复。

添加

zadd key score1 member1 score2 member2 ...

例1：向键'a4'的集合中添加元素'lisi'、'wangwu'、'zhaoliu'、'zhangsan'，权重分别为4、5、6、3

zadd a4 4 lisi 5 wangwu 6 zhaoliu 3 zhangsan

获取 返回指定范围内的元素

start、stop为元素的下标索引

索引从左侧开始，第⼀个元素为0

索引可以是负数，表示从尾部开始计数，如-1表示最后⼀个元素

zrange key start stop

例2：获取键'a4'的集合中所有元素

zrange a4 0 -1

返回score值在min和max之间的成员

zrangebyscore key min max

例3：获取键'a4'的集合中权限值在5和6之间的成员

zrangebyscore a4 5 6

返回成员member的score值

zscore key member

例4：获取键'a4'的集合中元素'zhangsan'的权重

zscore a4 zhangsan

删除 删除指定元素

zrem key member1 member2 ...

例5：删除集合'a4'中元素'zhangsan'

zrem a4 zhangsan

删除权重在指定范围的元素

zremrangebyscore key min max

例6：删除集合'a4'中权限在5、6之间的元素

zremrangebyscore a4 5 6

**什么是Redis持久化？Redis有哪几种持久化方式？优缺点是什么？**

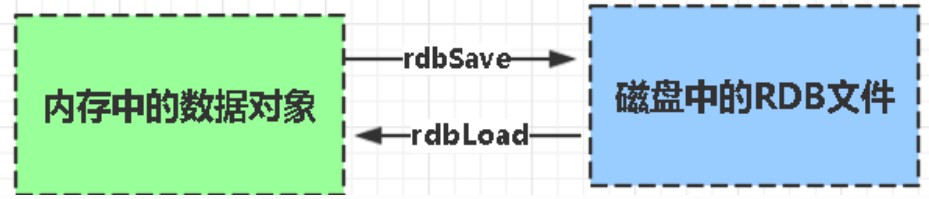
持久化就是把内存的数据写到磁盘中去，防止服务宕机了内存数据丢失。

Redis 提供了两种持久化方式:RDB（默认） 和AOF

**RDB：**

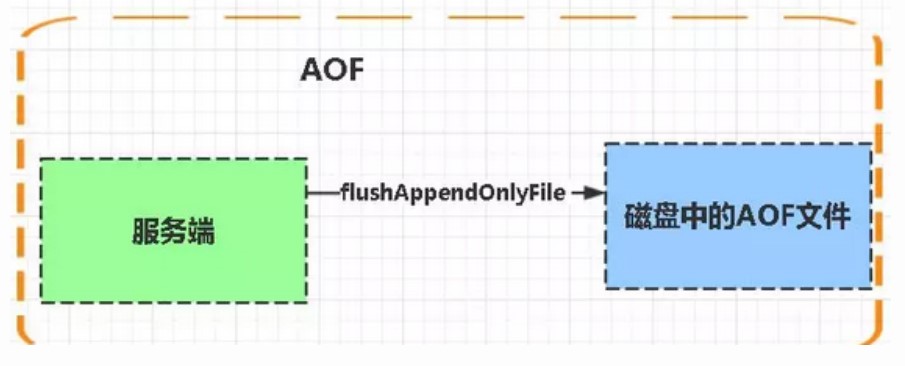
rdb是Redis DataBase缩写

功能核心函数rdbSave(生成RDB文件)和rdbLoad（从文件加载内存）两个函数



**AOF:**

Aof是Append-only file缩写



每当执行服务器(定时)任务或者函数时flushAppendOnlyFile 函数都会被调用， 这个函数执行以下两个工作

aof写入保存：

WRITE：根据条件，将 aof\_buf 中的缓存写入到 AOF 文件

SAVE：根据条件，调用 fsync 或 fdatasync 函数，将 AOF 文件保存到磁盘中。

**存储结构:**

  内容是redis通讯协议(RESP )格式的命令文本存储。

**比较**：

1、aof文件比rdb更新频率高，优先使用aof还原数据。

2、aof比rdb更安全也更大

3、rdb性能比aof好

4、如果两个都配了优先加载AOF

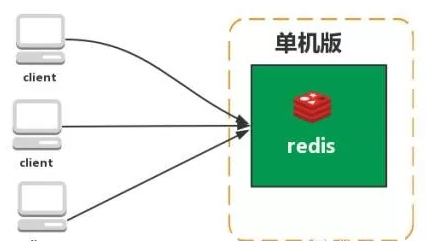
**刚刚上面你有提到redis通讯协议(RESP )，能解释下什么是RESP？有什么特点？（可以看到很多面试其实都是连环炮，面试官其实在等着你回答到这个点，如果你答上了对你的评价就又加了一分）**

RESP 是redis客户端和服务端之前使用的一种通讯协议；

RESP 的特点：实现简单、快速解析、可读性好

**Redis 有哪些架构模式？讲讲各自的特点**

**单机版**

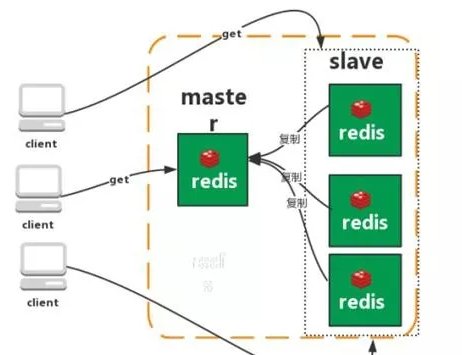


特点：简单

问题：

1、内存容量有限 2、处理能力有限 3、无法高可用。

**主从复制**



Redis 的复制（replication）功能允许用户根据一个 Redis 服务器来创建任意多个该服务器的复制品，其中被复制的服务器为主服务器（master），而通过复制创建出来的服务器复制品则为从服务器（slave）。 只要主从服务器之间的网络连接正常，主从服务器两者会具有相同的数据，主服务器就会一直将发生在自己身上的数据更新同步 给从服务器，从而一直保证主从服务器的数据相同。

特点：

1、master/slave 角色

2、master/slave 数据相同

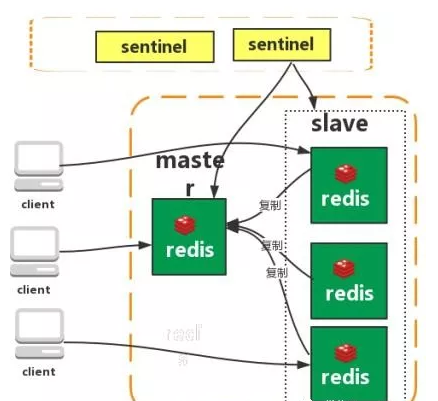
3、降低 master 读压力在转交从库

问题：

无法保证高可用

没有解决 master 写的压力

**哨兵**



Redis sentinel 是一个分布式系统中监控 redis 主从服务器，并在主服务器下线时自动进行故障转移。其中三个特性：

监控（Monitoring）：    Sentinel  会不断地检查你的主服务器和从服务器是否运作正常。

提醒（Notification）： 当被监控的某个 Redis 服务器出现问题时， Sentinel 可以通过 API 向管理员或者其他应用程序发送通知。

自动故障迁移（Automatic failover）： 当一个主服务器不能正常工作时， Sentinel 会开始一次自动故障迁移操作。（失效备援（为系统备援能力的一种，当系统中其中一项设备失效而无法运作时，另一项设备即可自动接手原失效系统所执行的工作））

特点：

1、保证高可用

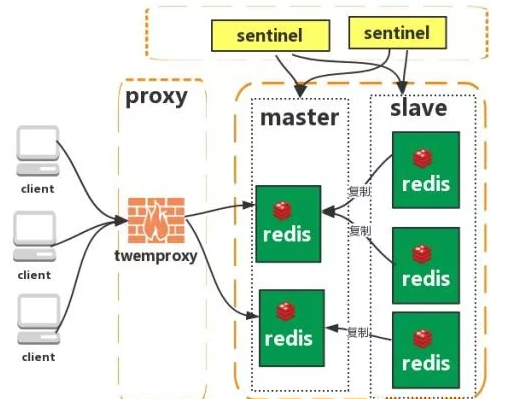
2、监控各个节点

3、自动故障迁移

缺点：主从模式，切换需要时间丢数据

没有解决 master 写的压力

**集群（proxy 型）：**



Twemproxy 是一个 Twitter 开源的一个 redis 和 memcache 快速/轻量级代理服务器； Twemproxy 是一个快速的单线程代理程序，支持 Memcached ASCII 协议和 redis 协议。

特点：1、多种 hash 算法：MD5、CRC16、CRC32、CRC32a、hsieh、murmur、Jenkins

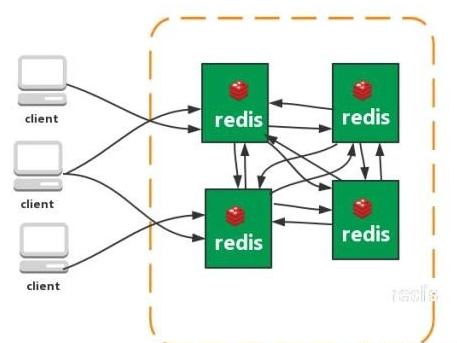
2、支持失败节点自动删除

3、后端 Sharding 分片逻辑对业务透明，业务方的读写方式和操作单个 Redis 一致

缺点：增加了新的 proxy，需要维护其高可用。

failover 逻辑需要自己实现，其本身不能支持故障的自动转移可扩展性差，进行扩缩容都需要手动干预

**集群（直连型）：**



从redis 3.0之后版本支持redis-cluster集群，Redis-Cluster采用无中心结构，每个节点保存数据和整个集群状态,每个节点都和其他所有节点连接。

特点：

1、无中心架构（不存在哪个节点影响性能瓶颈），少了 proxy 层。

2、数据按照 slot 存储分布在多个节点，节点间数据共享，可动态调整数据分布。

3、可扩展性，可线性扩展到 1000 个节点，节点可动态添加或删除。

4、高可用性，部分节点不可用时，集群仍可用。通过增加 Slave 做备份数据副本

5、实现故障自动 failover，节点之间通过 gossip 协议交换状态信息，用投票机制完成 Slave到 Master 的角色提升。

缺点：

1、资源隔离性较差，容易出现相互影响的情况。

2、数据通过异步复制,不保证数据的强一致性

**什么是一致性哈希算法？什么是哈希槽？**

这两个问题篇幅过长 网上找了两个解锁的不错的文章

<https://www.cnblogs.com/lpfuture/p/5796398.html>

<https://blog.csdn.net/z15732621582/article/details/79121213>

**Redis常用命令？**

Keys pattern

\*表示区配所有

以bit开头的

查看Exists  key是否存在

Set

设置 key 对应的值为 string 类型的 value。

setnx

设置 key 对应的值为 string 类型的 value。如果 key 已经存在，返回 0，nx 是 not exist 的意思。

删除某个key

第一次返回1 删除了 第二次返回0

Expire 设置过期时间（单位秒）

TTL查看剩下多少时间

返回负数则key失效，key不存在了

Setex

设置 key 对应的值为 string 类型的 value，并指定此键值对应的有效期。

Mset

一次设置多个 key 的值，成功返回 ok 表示所有的值都设置了，失败返回 0 表示没有任何值被设置。

Getset

设置 key 的值，并返回 key 的旧值。

Mget

一次获取多个 key 的值，如果对应 key 不存在，则对应返回 nil。

Incr

对 key 的值做加加操作,并返回新的值。注意 incr 一个不是 int 的 value 会返回错误，incr 一个不存在的 key，则设置 key 为 1

incrby

同 incr 类似，加指定值 ，key 不存在时候会设置 key，并认为原来的 value 是 0

Decr

对 key 的值做的是减减操作，decr 一个不存在 key，则设置 key 为-1

Decrby

同 decr，减指定值。

Append

给指定 key 的字符串值追加 value,返回新字符串值的长度。

Strlen

取指定 key 的 value 值的长度。

persist xxx(取消过期时间)

选择数据库（0-15库）

Select 0 //选择数据库

move age 1//把age 移动到1库

Randomkey随机返回一个key

Rename重命名

Type 返回数据类型

**08**

**使用过Redis分布式锁么，它是怎么实现的？**

先拿setnx来争抢锁，抢到之后，再用expire给锁加一个过期时间防止锁忘记了释放。

**如果在setnx之后执行expire之前进程意外crash或者要重启维护了，那会怎么样？**

set指令有非常复杂的参数，这个应该是可以同时把setnx和expire合成一条指令来用的！

**09**

**使用过Redis做异步队列么，你是怎么用的？有什么缺点？**

一般使用list结构作为队列，rpush生产消息，lpop消费消息。当lpop没有消息的时候，要适当sleep一会再重试。

缺点：

在消费者下线的情况下，生产的消息会丢失，得使用专业的消息队列如rabbitmq等。

**能不能生产一次消费多次呢？**

使用pub/sub主题订阅者模式，可以实现1:N的消息队列。

**10**

**什么是缓存穿透？如何避免？什么是缓存雪崩？何如避免？**

缓存穿透

一般的缓存系统，都是按照key去缓存查询，如果不存在对应的value，就应该去后端系统查找（比如DB）。一些恶意的请求会故意查询不存在的key,请求量很大，就会对后端系统造成很大的压力。这就叫做缓存穿透。

如何避免？

1：对查询结果为空的情况也进行缓存，缓存时间设置短一点，或者该key对应的数据insert了之后清理缓存。

2：对一定不存在的key进行过滤。可以把所有的可能存在的key放到一个大的Bitmap中，查询时通过该bitmap过滤。

缓存雪崩

当缓存服务器重启或者大量缓存集中在某一个时间段失效，这样在失效的时候，会给后端系统带来很大压力。导致系统崩溃。

如何避免？

1：在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量。比如对某个key只允许一个线程查询数据和写缓存，其他线程等待。

2：做二级缓存，A1为原始缓存，A2为拷贝缓存，A1失效时，可以访问A2，A1缓存失效时间设置为短期，A2设置为长期

3：不同的key，设置不同的过期时间，让缓存失效的时间点尽量均匀。

4.redis相比memecached相比有哪些优势

1) memcached所有值都是字符串,而redis支持丰富数据类型

2) redis速度比memcached快很多

3)redis可以数据持久化

4)灾难恢复 -memcache挂掉后，数据不可恢复;

而redis数据丢失后可以通过aof恢复:

(快照:顾名思义可以理解为拍照一样，把整个内存数据映射到硬盘中，保存一份到硬盘，因此恢复数据起来比较快，把数据映射回去即可，

AOF:一条条的执行操作命令。产生快照的过程)

5)过期策略--memcache在\*\*set\*\*时就指定，例如set key1 0 0 8,即永不过期。

Redis可以通过例如\*\*expire \*\*设定，例如expire name 10；

6)分布式--设定memcache集群，利用magent做一主多从;redis可以做一主多从。都可以一主一从；

应用场景

redis：数据量较小的更性能操作和运算上

memcache：用于在动态系统中减少数据库负载，提升性能;做缓存，提高性能（适合读多写少，对于数据量比较大，可以采用sharding）

MongoDB:主要解决海量数据的访问效率问题

来自 <<https://www.cnblogs.com/jasontec/p/9699242.html>>