
Redis简介

- NoSQL:泛指非关系型数据库,不支持SQL语法,存储数据格式都是KV形式
  - NoSQL都有自己的api和语法
  - 种类还有MongoDB...
  - 适合关系简单数据查询场景,不支持事务
- Redis 属于NoSQL(not only sql)
- Redis:
  - 所有数据放到内存中,也会保存在磁盘中
  - 支持多数据结构存储
  - 支持数据备份,及是master-slave主群模式
  - 优势:
    1. 读写*速度高*,读110000次/s
    2. 所有操作都是*原子性*
    3. 特性丰富,publish...key过期等

安装

- sudo apt-get install redis 安装缺点:
  - 不能指定版本
  - 配置文件目录不好找
  - redis.conf find / -name redis.conf
- 手动安装:好处:能安装最新版本
- 步骤:

i. 下载:

```
wget http://download.redis.io/releases/redis-x.x.x.tar.gz  # x.x.x 版本号
```

ii. 解压:

```
tar -xzf redis-x.x.x.tar.gz
```

iii. 移动: 放到usr/local 目录下

```
sudo mv ./redis-x.x.x/usr/local/redis/
```

iv. 进入redis目录

```
cd /usr/local/redis/
```

v. 生成redis.conf配置文件,若已有.configure 直接make 执行它

```
sudo make
```

vi. 安装,将redis 命令安装到 /usr/local/bin/

```
sudo make install
```

vii. 进入目录 /usr/local/bin中查看

```
cd /usr/local/bin
ls -all
redis-server redis服务器
**redis-cli redis命令行客户端
**redis-benchmark redis性能测试工具  redis-check-aof AOF文件修复工具  redis-check-rdb RDB文
```

viii. 配置文件,移动到/etc/目录下

```
配置文件目录为 /usr/local/redis/redis.conf
```

```
--sudo cp /usr/local/redis/redis.conf  /etc/redis/
```

1-配置: /etc/redis/redis.conf

a. 查看: sudo vi /etc/redis/redis.conf

b. 选项:

- 绑定ip:如果需要远程访问,可以注释掉或者绑定一个ip
- 端口: port 6379 # 默认
- 是否以守护进程:

  daemonize yes

  - 推介 yes ,不会在命令行阻塞,后台运行,类似服务
- 数据库,默认16个 : database 16
- 主从复制,类似双机备份: slaveof
- **数据类:
  - 数据文件: dbfilename dump.rdb
  - 数据文件存储路径: dir /var/lib/redis
  - 日志文件: logfile "/var/log/redis/redis-server.log"

2-服务器端-客户端命令

- **服务器端:** redis-server
  - ps aux | grep redis # 查看redis服务器进程
  - sudo kill -9 __(pid) # 杀死pid对应的redis 服务器
  - sudo service redis stop # 停止服务
  - **sudo redis-server /etc/redis/redis.conf** # 指定加载的配置文件,即按配置启动服务器
- 客户端: redis-cli
  - **redis-cli** --raw # 连接客户端 ,只有一个服务时候可以redis-cli
  - **-h ip地址 -p 端口** # 连接ip的p端口
  - --raw # 支持中文
  - ping # 运行测试
  - select __(num) # select 第num数据库, 默认16个,用0-15标识
- 帮助 redis-xxx --help

数据结构

- **redis是key-value的数据结构,每条都是一个键值对**
- 键类型是字符串
- 键不能重复
- 值类型:
  - **String字符串;--如记录用户名,存图片**
  - **hash哈希(类似字典);--如用户详细信息**

- list列表;--如发送邮件
- set集合;--如微博粉丝
- zset有序集合--如排行榜

- 数据操作行为:--保存---修改---获取---删除---

各个类型操作行为

	string	键命令
介绍	该类型可接受任何格式数据 场景:记录用户名,存图片信息	对键进行查看/修改操作
增加/修改	设置键值 set key value 设置键值及过期时间(s) setex key seconds value 多个键值 mset key1 value1 key2 value2 追加值 append key value	设置键的过期时间 expire key seconds 查看有效时间 ttl key
获取	根据键获取值,键不存返回 nil get key 根据多个获取多个 mget key1 key2...	查找键 keys pattern(规则,*所有) 判断键是否存在,在返回1,无返回0 exists key1 查看键类型 type key
删除		删除键及对应值(数据) del key1 key2...

	hash	list队列	set无序集
介绍	用于存储对象,对象结构为属性filed,值value 场景:用户详细信息	按照插入顺序排序 队列特性,先进先出	元素具有唯一 对于集合没有 场景:微博,粉
增加/修改	设置单个属性 hset key field value 设置多个属性 hmset key field1 value1 field2 value2	左侧插入数据 lpush key value1... 右侧插入 rpush key value1.. 在指定元素前后插入 linsert key before 现在 新加入 ...after... 设置指定索引位置元素值 lset key index value index 为下标	添加元素 sadd key m 没有修改操作
获取	获取指定键所有属性 hkeys key 获取一个属性的值	返回列表里指定范围内元素 lrange key start stop start,stop为下标索引	返回所有元素 smembers

	hget key field 获取多个属性的值 hmget key field1 field2... 获取所有属性的值 hvals key	-1 表示最后一个	
删除	删除整个,使用del 删除属性,对应值也会被删除 hdel key field1...	删除指定元素 lrem key count value 将列表中前count次出现为value元素删除 count>0从头 count<0从尾 count=0所有 截取修剪 ltrim key start stop [start,stop]区间外元素 全部删除	删除指定元素 srem key
总结:	一个键可以放一种数据结构,存储数据, 键-数据,而数据里面也有各种结构, 键-(数据键:值) 使用数据结构方法时,每个方法首字母都对应其名字首字母 获取值: 数据类型方法 key 删除: 数据类型首字母del key		

redis事务

注意Redis Cluster集群不支持事务

redis通过multi开启事务，相当于把命令添加到队列QUEUE中，最后exec执行

```
 # 进入redis-cli -p  6381
127.0.0.1:6381> multi
OK
127.0.0.1:6381> set a 100
QUEUED
127.0.0.1:6381> set b 200
QUEUED
127.0.0.1:6381> get a
QUEUED
127.0.0.1:6381> get b
QUEUED
127.0.0.1:6381> exec
1) OK
2) OK
3) "100"
4) "200"

但是期间如果出现问题，不会出现回滚现象，不会影响其他命令的执行。
127.0.0.1:6381> multi
```

```
OK
127.0.0.1:6381> set c 300
QUEUED
127.0.0.1:6381> hget a name
QUEUED
127.0.0.1:6381> set d 400
QUEUED
127.0.0.1:6381> set f 500
QUEUED
127.0.0.1:6381> exec
1) OK
2) (error) WRONGTYPE Operation against a key holding the wrong kind of value
3) OK
4) OK
```

watch监视命令

watch用来配合事务监视某个键是否变化，如果在事务没结束前键值就发生了变化，那么整个事务失效

```
27.0.0.1:6381> set stock 100
OK
127.0.0.1:6381> watch stock
OK
127.0.0.1:6381> multi
OK5
127.0.0.1:6381> incrby stock -2
QUEUED
127.0.0.1:6381> incrby sales 2
QUEUED
127.0.0.1:6381> exec
(nil)
 即 在执行exec之前，如果stock的值被修改了，则事务失败
```

Redis持久化

1）RDB快照持久化（默认开启）--保存数据

```
触发机制：
    定期触发：
```

```
        redis配置文件中：
            save 900 1          # 900秒中有1次及以上修改到redis中数据
            save 300 10         # 300秒中有10次及以上修改到redis中数据
            save 60 10000       # 60秒中有10000次及以上修改到redis中数据

    BGSAVE命令触发：
        执行BGSAVE命令，手动触发RDB持久化

    SHUTDOWN触发：
        SHUTDOWN命令，关闭redis时触发
```

2）AOF快照持久化（默认不开启）--保存语句

```
把执行的指令记录起在文件中

redis配置文件中：
    appendonly yes  # 是否开启AOF
    appendfilename "appendonly.aof"  # AOF文件

触发机制：
    # appendfsync always    # 执行每个命令都记录
    appendfsync everysec    # 每秒记录
    # appendfsync no    # 交给操作系统决定写到操作系统的时机

使用AOF机制的缺点是随着时间的流逝，AOF文件会变得很大。但redis可以压缩AOF文件。
```

Redis高可用

1)、redis主从配置

```
    客户端中通过命令 slaveof 主机地址 主机端口  来设置为从机

    也可以通过redis配置文件中：
    # slaveof <masterip> <masterport>
    来设置为从机

    客户端中通过 info Replication  查看该redis复制集信息
    客户端中通过 info 命令查看redis整机状态
```

Redis主从同步策略

主从刚刚连接的时候，进行全量同步；全同步结束后，进行增量同步。当然，如果有需要，slave 在任何时候都可以发起全量同步。redis 策略是，无论如何，首先会尝试进行增量同步，如不成功，要求从机进行全量同步。

2)、redis哨兵机制

哨兵其实本身也是一个可执行程序，像我们启动redis客户端一样，只是它是用来看护redis实例进程的。

哨兵提供以下几个功能：

- Monitoring 监控
- Notification 通知
- Automatic failover 自动故障转移 --主机宕机，从机自动切换补位，主机重启变从机
- Configuration provider 配置提供程序

```
在redis安装后，会自带sentinel哨兵程序，修改sentinel.conf配置文件
    bind 127.0.0.1
    port 26380
    daemonize yes
    # 哨兵日志文件
    logfile /var/log/redis-sentinel.log
    sentinel monitor mymaster 127.0.0.1 6380 2
    # 多久时间没有回应认为宕机
    sentinel down-after-milliseconds mymaster 30000
    sentinel parallel-syncs mymaster 1
    # 宕机后经过多少时间开启故障转移
    sentinel failover-timeout mymaster 18000

实际上，它就是个配置文件sentinel.conf，再加上一个启动命令：
    redis-sentinel sentinel.conf
```

sentinel monitor mymaster 127.0.0.1 6380 2 说明

- mymaster 为sentinel监护的redis主从集群起名
- 127.0.0.1 6300 为主从中任一台机器地址
- 2 表示有两台以的sentinel认为某一台redis宕机后，才会进行自动故障转移

3)、头条项目中的redis

```
        cluster -> 7000 7001 7002 7003 7004 7005      (缓存)
        master & slave -> 6380 6381    (持久化存储)
    哨兵sentinel -> 26380 26381 26382

    目录:  /etc/redis

    停止redis服务:
        redis终端中，shutdown
    启动redis服务:
        sudo /usr/local/bin/redis-server /etc/redis/6381.conf

```

4)、代码中使用redis哨兵客户端

```
REDIS_SENTINELS = [
    ('127.0.0.1', '26380'),
    ('127.0.0.1', '26381'),
    ('127.0.0.1', '26382'),
]
REDIS_SENTINEL_SERVICE_NAME = 'mymaster'

from redis.sentinel import Sentinel

_sentinel = Sentinel(REDIS_SENTINELS)   # 创建哨兵对象
redis_master = _sentinel.master_for(REDIS_SENTINEL_SERVICE_NAME)  # 连接主服务
redis_slave = _sentinel.slave_for(REDIS_SENTINEL_SERVICE_NAME)   # 连接从服务

redis_master.set("key","value")
redis_slave.get("key")
```

Redis集群

1)、创建集群的命令:

redis-trib.rb create --replicas 1 192.168.192.200:7000 192.168.192.200:7001 192.168.192.200:7002 192.168.192.200:7003 192.168.192.200:7004 192.168.192.200:7005

2)、终端中连接到redis集群

```
redis-cli -c -p 7000
```

3)、代码中连接到redis集群

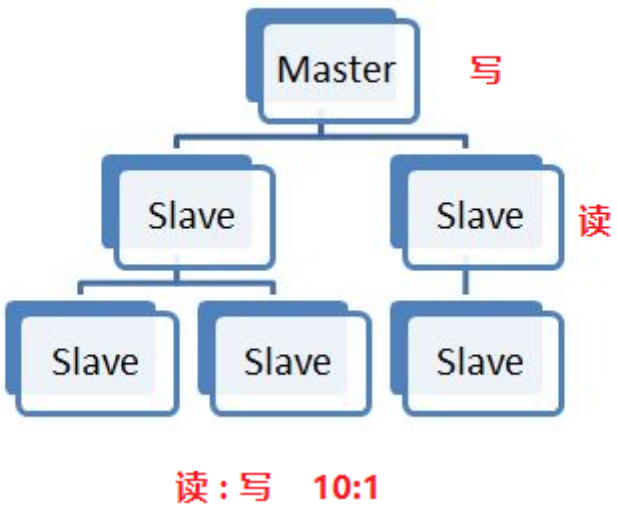
```
from rediscluster import StrictRedisCluster
# redis 集群
REDIS_CLUSTER = [
    {'host': '127.0.0.1', 'port': '7000'},
    {'host': '127.0.0.1', 'port': '7001'},
    {'host': '127.0.0.1', 'port': '7002'},
]

redis_cluster = StrictRedisCluster(startup_nodes=REDIS_CLUSTER,decode_responses=True)

# 可以将redis_cluster就当作普通的redis客户端使用
redis_cluster.delete(key)
```

搭建主从服务器

- 一个主机Master可以拥有多个slave,一个slave也可以拥有多个slave
- master 写数据 slave 读数据，统计局说 读与写的比率=10:1
- master 和slave 都是一个 redis实例(redis服务)



- **配置方法:**

  i. 配置主:

  - 查看当前主机ip--ifconfig
  - 修改 /etc/redis/redis.conf 文件
    - **sudo vi redis.conf**

```
bind ip地址   # linux: 末行/bind 快速寻找
```

- 重启redis

```
sudo service redis stop
sudo redis-server redis.conf
```

ii. 配置从slave:

- 复制 /etc/redis/redis.conf 文件

```
sudo cp redis.conf ./slave.conf
```

- 修改 redis/slave.conf 文件

```
sudo vi slave.conf
```

- 编辑内容

```
bind ip地址
slaveof ip地址 端口    # 空格隔开
port 自己端口
```

- redis服务

```
sudo redis-server slave.conf
```

iii. 查看主从关系

```
redis-cli -h ip地址 info Replication
```

- 数据操作
  - 进入master :redis-cli -h ip地址 -p 6379
  - 进入slave : redis-cli -h ip地址 -p 6378
  - 在master上写 ,在slave 上读数据

集群

- 概念
  - 一组相互独立,通过高速网络互联的计算机
- 为什么要有集群:
  - **一主可以多从,多服务器同时访问**
- **redis集群--至少6个,3主3从**
  - 参考阅读:
    - redis集群搭建 http://www.cnblogs.com/wuxl360/p/5920330.html

- [Python]搭建redis集群 http://blog.5ibc.net/p/51020.html

- 软件层面: 只有一台电脑,在这台电脑启动多个redis服务
- 硬件层面: 存在多台实体电脑,每台电脑都启动redis或者多个
- 配置集群步骤:
  1. 配置机器1
     a. 进入Desktop目录,创建conf目录,创建文件7000.conf,编辑
     - port 7000 #7001 7002
     - bind 当前ip1
     - daemonize yes # 后端
     - pidfile 7000.pid # 文件 7001.pid 7002.pid
     - cluster-enabled yes #允许集群
     - cluster-config-file nodes_7000.conf #集群配置,自动生成7000 7001 7002
     - cluster-node-timeout 15000 # 请求超时 默认15秒
     - appendonly yes # aof日志开启,每次写操作都记录一条日志

     b. 再进行上一步操作, 文件为7001 7002
     c. 使用配置文件启动redis服务

```
redis-server 7000.conf ...
```

  2. 配置机器2
     a. 重复机器1操作,使用机器2ip
     b. 编辑.conf文件
     - port pidfile cluseter-config-file 改为对应7003 7004 7005

     c. 启动redis服务 参考1.3.0

  3. 创建集群---redis-trib.rb
     a. 在机器1操作
     b. 安装ruby环境
     - 因为redis-trib.rb 是用ruby开发的

     c. 执行命令,添加redis-trib.rb 至环境变量中/usr/local/bin/

```
sudo cp /usr/share/doc/redis-tools/examples/redis-trib.rb /usr/local/bin/
```

     d. 运行创建

```
redis-trib.rb create --replicas 1 机器1,机器2的各个ip:端口
```

     i. 若是报错,需要设置gem源

```
-- 先查看自己的 gem 源是什么地址
gem source -l -- 如果是https://rubygems.org/ 就需要更换
-- 更换指令为
gem sources --add https://gems.ruby-china.com/ --remove https://rubygems.or
-- 通过 gem 安装 redis 的相关依赖
sudo gem install redis
-- 然后重新执行指令
```

     ii. 提示主从信息,yes

iii. 连接时,需要加参数-c 表示连接到集群

```
redis-cli -h 172.16.179.131 -c -p 7002
```

iv. 特性:

1. 在哪个服务器上写数据?

每个节点都是平等关系,set,get时候会采用hash slot(哈希槽)方式分配,所以表现出set,get跳转到其他端口

2. master会和对应salve之间进行数据同步,只有当一个master挂掉之后,才会启动一个对应的salve节点充当master,salve of one