MYsql 数据表在300-500万条数据后性能开始急速下降

8G运行32G内存下mysql大概支持2000-5000的并发???

**Redis最好使用单数节点**

**缓存穿透**

# 1、安装与集群搭建

**Redis安装**

安装gcc，yum install gcc(解析redis源码)

解压redis，tar-zxvf redis-3.0.0-rcz.tar.gz

进入到redis-3.0.0/src目录下进行安装，安装完成后验证src目录下是否生成了redis-server和redis-cli,然后make install

1 下载地址：http://redis.io/download

2 安装步骤：

3 # 安装gcc

4 yum install gcc

5

6 # 把下载好的redis‐5.0.3.tar.gz放在/usr/local文件夹下，并解压

7 wget http://download.redis.io/releases/redis‐5.0.3.tar.gz

8 tar xzf redis‐5.0.3.tar.gz

9 cd redis‐5.0.3

10

11 # 进入到解压好的redis‐5.0.3目录下，进行编译与安装

12 make

13

14 # 启动并指定配置文件

15 src/redis‐server redis.conf（注意要使用后台启动，所以修改redis.conf里的daemonize改为y

es)

16

17 # 验证启动是否成功

18 ps ‐ef | grep redis

19

20 # 进入redis客户端

21 src/redis‐cli

22

23 # 退出客户端

24 quit

25

26 # 退出redis服务：

27 （1）pkill redis‐server

28 （2）kill 进程号

29 （3）src/redis‐cli shutdown

**Redis的复制原理**

**全量同步**  
Redis全量复制一般发生在Slave初始化阶段，这时Slave需要将Master上的所有数据都复制一份。具体步骤如下：   
-  从服务器连接主服务器，发送SYNC命令；   
-  主服务器接收到SYNC命名后，开始执行BGSAVE命令生成RDB文件并使用缓冲区记录此后执行的所有写命令；   
-  主服务器BGSAVE执行完后，向所有从服务器发送快照文件，并在发送期间继续记录被执行的写命令；   
-  从服务器收到快照文件后丢弃所有旧数据，载入收到的快照；   
-  主服务器快照发送完毕后开始向从服务器发送缓冲区中的写命令；   
-  从服务器完成对快照的载入，开始接收命令请求，并执行来自主服务器缓冲区的写命令

**增量同步**  
Redis增量复制是指Slave初始化后开始正常工作时主服务器发生的写操作同步到从服务器的过程。   
增量复制的过程主要是主服务器每执行一个写命令就会向从服务器发送相同的写命令，从服务器接收并执行收到的写命令。

**主从复制**

Slaveof 127.0.0.1 6379命令行指定为谁的从节点，也可以在配置文件中配置（不建议），指定为从服务器后会删除原有数据，复制主服务器所有数据

Info replication 查看服务器信息

Slaveof no one清除主从命令（执行后复制的主服务器信息还在）

主服务器挂掉，从服务器变为status up原地待命状态，可读不可写

Shut down 关闭服务器，命令指定的从服务器宕机重启后变为master，需要命令行重新配置

**薪火相传**

Master控制一台从服务器，从服务器控制一台从服务器，一次向下传递，所有服务器中只有一台主服务器，薪火相传实现了去中心化，但串联操作中间节点挂掉后面节点都不能使用，故实际生产环境中不会使用

**哨兵模式**

Sentinel.conf文件内配置，vim sentinel.conf编辑该文件

Sentinel monitor host6379（mastre名随意写）127.0.0.1 6379 1（从服务器获得最低的升级为主服务器的票数）

**集群**

搭建集群需要安装ruby脚本

Yum install ruby

Yum install rubygems

Gem installredis-version 3.0.0(安装redis和ruby的接口)

搭建集群

# ./redis-trip.rb create --replicas 1(主节点数/从节点数) ip（所有节点ip，192.168.0.61：8001 。。。）

前面几台服务器为主节点，后面为从节点，第一个主节点对应最开始的几台从节点，集群的数据写入根据轮训来确定master，拿数据通过重定向拿，不论数据在哪个节点上

Ps -ef|grep redis 看进程，各个redis服务器

集群中的master挂掉重启后变为该小集群的从节点

**集群的分片？**

Cluster nodes查看节点状态

**加入新的服务器**

/user/local/redis-3.0.0/src/redis-trib.rb add-node 192.168.0.61:8007(新节点) 192.168.0.61.8001（集群内任意节点）

节点加入后分配hash槽，找到集群内任意主节点

/user/local/redis-3.0.0/src/redis-trip.rb reshard 主节点ip -->新节点分配多少槽位-->输入接受槽位的服务器ID

每个主节点分配一个范围的槽位给新的主节点，故新的主节点有数个范围的槽位

**给新的主节点加入从服务器**

在从服务器客户端 cluster replicate ID（新主节点的ID）

**删除从节点在**cluster：#/user/local/redis-3.0.0/src/redis-trip.rb del-node 192.168.0.61:8008 ID(从节点ID)

**删除主节点**，首先需要把数据转移到其他主节点上（同加入槽位步骤大体相同）

/user/local/redis-3.0.0/src/redis-trip.rb reshard 192.168.0.61:8007

输入移动槽位数-->移动到哪（集群内任意主节点ID）——>数据源（移动数据的主节点ID）-->done执行

# 2、原理

**Redis的单线程和高性能**

Redis 单线程为什么还能这么快？

因为它所有的数据都在内存中，所有的运算都是内存级别的运算，而且单线程避免了多线程的切换性能损耗问题。正因为 Redis 是单线程，所以要小心使用 Redis 指令，对于那些耗时的指令(比如keys)，一定要谨慎使用，一不小心就可能会导致 Redis 卡顿。

Redis 单线程如何处理那么多的并发客户端连接？

Redis的IO多路复用：redis利用epoll来实现IO多路复用，将连接信息和事件放到队列中，依次放到文件事件分派器，事件分派器将事件分发给事件处理器。

Nginx也是采用IO多路复用原理解决C10K问题