redis 简介

1.redis是什么？

开源非关系型数据库，是一个高性能的key-values数据库

a:1

b:2

数据可持续化，redis可将数据永久写入磁盘，实现数据持久化

数据不仅支持key-values类型的数据，还可以使用set list zset hash等数据结构

支持主从复制

2.redis的特点有哪些？

2.1 速度快，110000次/s读的速度，81000次/s写的速度

2.2 数据类型丰富

2.3 数据具有原子性

3.redis与其他非关系型数据库有什么不同？

3.1 数据拥有原子性

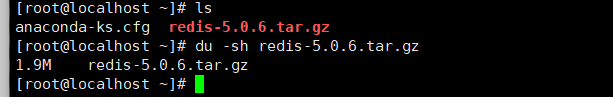
3.2 数据可持久化

redis安装过程

1.源码下载

# wget http://download.redis.io/releases/redis-4.0.11.tar.gz （官网）

# wget <http://download.redis.io/releases/redis-5.0.6.tar.gz> 最新



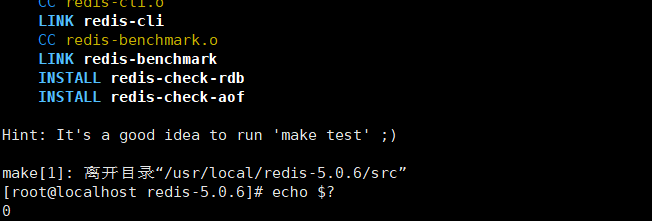
2.安装

# yum install -y gcc gcc-c++

# tar xf redis-4.0.11.tar.gz -C /usr/local/

# cd /usr/local/redis-4.0.11

# make



注意，如果出现下面的问题，请使用：make MALLOC=libc

致命错误：jemalloc/jemalloc.h：没有那个文件或目录

# mv /usr/local/redis-4.0.11/ /usr/local/redis

安装完成

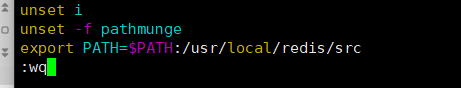
redis启动和基本使用

由于我们使用源码安装的，所以redis的启动命令是没有被添加进我们的系统变量的，需要手动配置

# export PATH=$PATH:/usr/local/redis/src 设置临时变量

# vim /etc/profile 加入长期环境变量

export PATH=$PATH:/usr/local/redis/src



# source /etc/profile

启动redis：

没有加入环境变量的：

# cd /usr/local/redis/src

# ./redis-server 如果以这种方式启动，为前台启动

# ./redis-server & 后台启动

加入环境变量的：

# redis-server 前台

# redis-server & 后台

或者，拷贝启动脚本到/etc/init.d/获取更便捷的启动方式

# cp /usr/local/redis/utils/redis\_init\_script /etc/init.d/redis

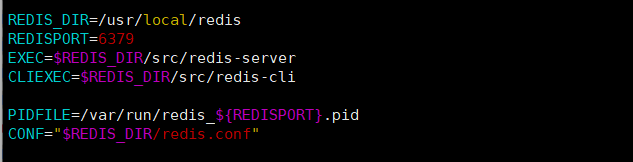
# vim /etc/init.d/redis

REDIS\_DIR=/usr/local/redis 指定redis安装目录

EXEC=$REDIS\_DIR/src/redis-server 指定redis启动命令路径

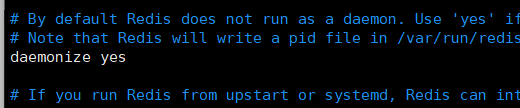
CLIEXEC=$REDIS\_DIR/src/redis-cli 指定redis客户端命令路径

CONF="$REDIS\_DIR/redis.conf" 指定redis配置文件路径



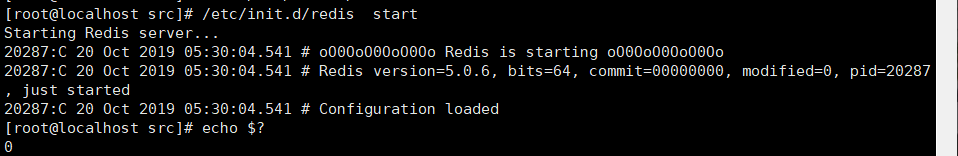
# vim /usr/local/redis/redis.conf

daemonize yes 将这里改成yes，否则你的redis每次启动都要加'&'才行



修改完成后：

# /etc/init.d/redis [start stop]



lsof -i:6379

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

如果加了密码后，停止redis时报错解决：

......

Waiting for Redis to shutdown ...

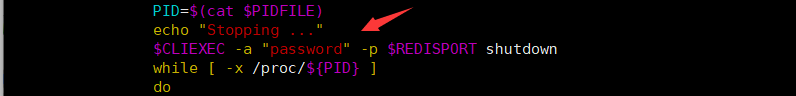
Waiting for Redis to shutdown ...

Waiting for Redis to shutdown ...

......

# vim /etc/init.d/redis

$CLIEXEC -a "password" -p $REDISPORT shutdown 修改这一行



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

登陆redis：

没有加入环境变量的：

# cd /usr/local/redis/src

# ./redis-cli

加入环境变量的：

redis-cli

127.0.0.1:6379> 只要看到这个提示符，表示登陆成功

127.0.0.1:6379>config get \*

==============================================================================

redis配置详解：

主要关心以下几个

daemonize no // 是否以守护进程启动redis（改成yes，否则每次启动redis都得加&）\*\*\*\*\*

pidfile /var/run/redis.pid // redis以守护进程启动后pid文件的路径

port 6379 // redis监听端口 \*\*\*\*\*

bind 127.0.0.1 // redis监听的主机

dir ./ // 指定redis数据目录路径 \*\*\*\*\*

timeout 300 // 指定redis客户端在多长时间后退出登录，如果是0则表示关闭该功能

logfile stdout // 日志记录方式，默认为标准输出，可自定义文件路径

save 900 1 // 该配置默认有三行，表示redis在900s内有一次更改，就将数据写入磁盘

save 300 10 // 300s内有10次便写入

save 60 10000 // 60s内有1w次便写入

masterauth // master设置的供slave连接时使用的密码

requirepass foobared // redis设置连接密码，设置后client登录后需要验证密码才能继续 \*\*\*\*

appendonly no // 是否开启数据同步写入，默认为异步写入

=====================================================================

redis基础操作命令

客户端操作命令

本地连接

# redis-cli 连接本地redis服务器

# redis-cli

127.0.0.1:6379> auth password 连接本地redis服务器后输入密码

# redis\_cli -a password 输入密码后登陆本地redis服务器

--------------------------------------------------------------------

远程连接

# redis-cli -h host -p port -a passwd 连接远程redis服务器

例如：

# redis-cli -h 3.3.3.8 -p 6379 -a 'test123'

==========================================================================

redis数据类型

string 字符串；该类型是redis中最基本的类型，一个key对应一个values，最大存储能力512M

字符串（string）操作命令

set key value 设置key的值

mset key1 value1 key2 value2 同时设置一个或多个key-value对

mset key1 value1 key2 value2 设置多个key-value,且只有当所有key都不存在时方可成功

get key 获取key的值

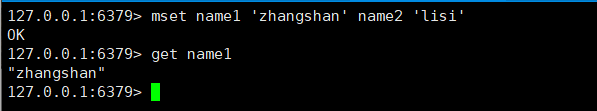
mget key1 key2 获取多个key的值

getset key value 修改key的值，并返回旧的值

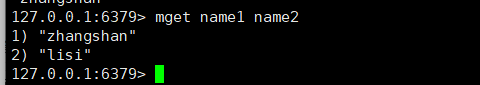
例如：

127.0.0.1:6379> mset name1 'zhangsan' name2 'lisi'

127.0.0.1:6379> get name1



127.0.0.1:6379> mget name1 name2



==========================================================================

hash 哈希；键值对集合，最常用与储存对象，最大存储能力2\*\*32-1个键值对

哈希（hash）操作命令

hset myhash key1 value1 key2 value2 创建hash

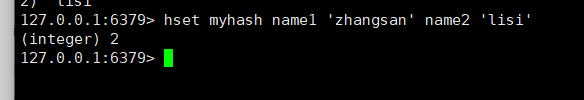
hget hashname keyname 获取哈希表中指定字段的值

hmget hashname keyname1 keyname2 批量获取哈希表中指定字段的值

hgetall hashname 获取哈希表中所有的键值对

例如：

127.0.0.1:6379> hset myhash name1 'zhangsan' name2 'lisi'



127.0.0.1:6379> hget myhash name1

'zhangsan'

127.0.0.1:6379> hget myhash name2

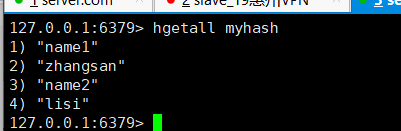
'lisi'

127.0.0.1:6379> hmget myhash name1 name2

1) 'zhangsan'

2) 'lisi'

127.0.0.1:6379> hgetall myhash



1) 'zhangsan'

2) 'lisi'

==========================================================================

list 列表；字符串列表，按照插入顺序排列，最大存储能力2\*\*32-1个元素

列表（list）操作命令

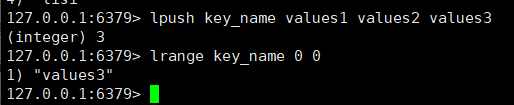
lpush list\_name value 创建列表

lrange list\_name n m 查看列表的n至m条数据

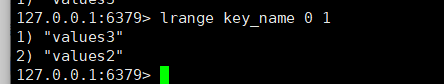
例如：

127.0.0.1:6379> lpush key\_name values1 values2 values3 ......

127.0.0.1:6379> lrange key\_name 0 0



127.0.0.1:6379> lrange key\_name 0 1



==========================================================================

set 集合（无序的）；字符串的无序集合，set是通过哈希表实现的，集合内的值不能重复提交，最大存储能力2\*\*32-1个成员

集合（set）操作命令

sadd key value 创建set

smembers key1 查看set

例如：

127.0.0.1:6379> sadd set\_name values1

(integer) 1

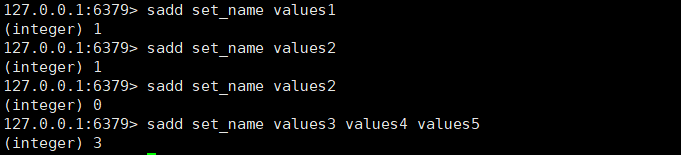
127.0.0.1:6379> sadd set\_name values2

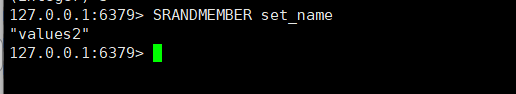
(integer) 1

127.0.0.1:6379> sadd set\_name values2

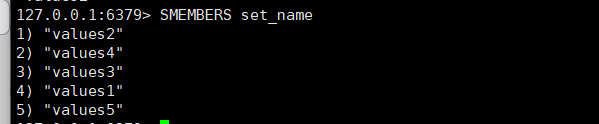
(integer) 0 重复提交数据返回0，且会直接被忽略掉

127.0.0.1:6379> sadd set\_name values3 values4 values5 一次设置多个值

127.0.0.1:6379> srandmember set\_name 随机返回一个set中的值



127.0.0.1:6379> smembers set\_name



127.0.0.1:6379> srem set\_name values1 删除set中的某一个或多个值

==========================================================================

zset 有序集合；和set类似，不过该类型的值是有序排列的

有序集合（zset）操作命令

zadd key value1 创建zset

zrange value1 n m 查看有序集合指定的n至m个成员

例如：

127.0.0.1:6379> zadd zset\_name 0 values1 成员前方的0表示分数，分数越低，排序时越靠前

(integer) 1

127.0.0.1:6379> zadd zset\_name 0 values2

(integer) 1

127.0.0.1:6379> zadd zset\_name 0 values2

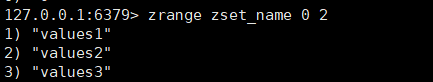
(integer) 0 和set一样，无法插入重复的数据

127.0.0.1:6379> zadd zset\_name 0 values3

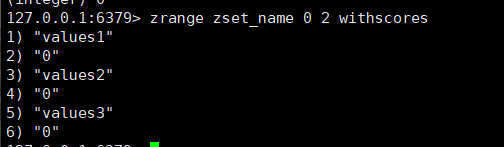
(integer) 1

127.0.0.1:6379> zadd zset\_name 0 values1 0 values2 0 values3 批量添加

127.0.0.1:6379> zrange zset\_name 0 2 从第一个成员开始，打印至第二个成员



127.0.0.1:6379> zrange zset\_name 0 2 withscores 显示指定范围的值，并显示每个值的分数



127.0.0.1:6379> zcard zset\_name 获取集合中的成员数（查看一共有多少个值）

127.0.0.1:6379> zscan zset\_name 0 获取集合中所有的值和对应的分数（0可以为任何数字）

==========================================================================

所有数据类型key的操作命令

del key 删除一个key

exists key 检查key是否存在

expire key n 给key设定过期时间n，单位s，到时间后redis将自动删除该key

pexpire key n 给key设定过期时间n，单位ms

pttl key 查看key的过期时间，单位ms

ttl key 查看key的过期时间，单位s

persist key 移除key的过期时间

rename key newkey 修改key名称，如果newkey已经存在，则key覆盖newkey的值

renamenx key newkey 修改key名称，如果newkey已经存在，则返回0，不修改

type key 查看key的类型

==========================================================================

Python客户端连接redis

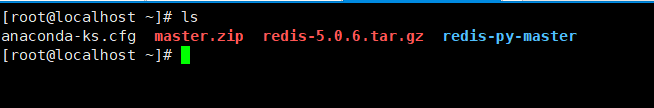
安装redis-py

1、下载：wget https://github.com/andymccurdy/redis-py/archive/master.zip

2、安装python程序

# yum install -y python

3、解压：unzip master.zip



4、安装：

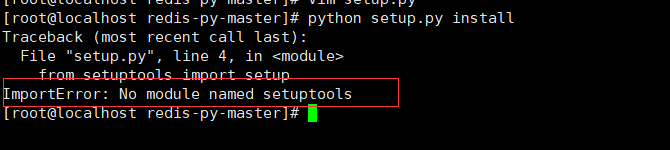
# cd redis-py-master

# python setup.py install

不报错即表示安装成功

如果出现以下错误：

ImportError: No module named setuptools



是少了个python扩展包：

# yum install -y python-setuptools

然后重新使用：# python setup.py install安装即可

测试使用redis-py

# python

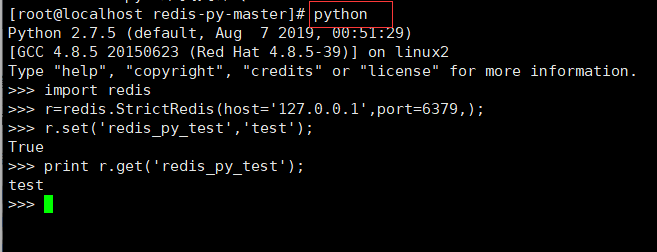
>>> import redis --调用redis模块

>>> r = redis.StrictRedis(host='127.0.0.1',port=6379,password='\*\*\*\*\*\*'); --设置连接redis数据库

>>> r.set('redis\_py\_test','test'); --设置键值对（string类型）

>>> print r.get('redis\_py\_test'); --获取key的值

test



hash

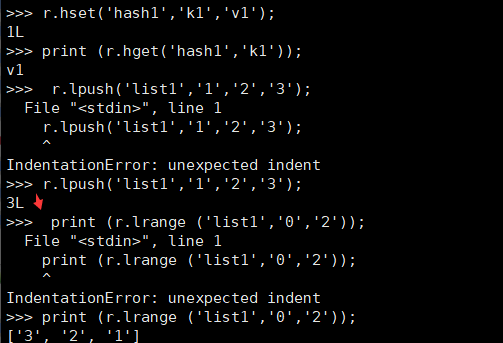
>>> r.hset('hash1','k1','v1');

>>> print (r.hget('hash1','k1'));

列表

>>> r.lpush('list1','1','2','3');

>>> print (r.lrange ('list1','0','2'));



其余类型自行测试。。。。。。

==========================================================================

redis数据持久化

数据持久化一共三种办法：

一、数据自动写入

1.appendfsync no

appendfsync为no，表示redis不会自动将内存中的临时数据写入磁盘中，全靠系统来进行数据保存，大多数linux系统默认为每30s保存一次缓冲池内的数据到磁盘

2.appendfsync everysec （redis默认数据写入级别）

该级别下，redis会每隔1秒进行一次数据写入磁盘操作，但如果某一次数据写入超过1秒，redis将会自动延迟一秒进行下一波写入操作，这一次写入不管花多长时间都会一直进行，但由于这时写入操作被占用，后面的写入操作将会被阻塞，导致部分数据无法正常写入；这种骚操作在大多数数据库中被称为“group commit”批量写入。

3.appednfsync always

该级别下，每一次写操作redis都会进行一次写入操作，这种级别下对数据的安全性是最好的，但由于写入次数过多，对性能会有一定影响

大佬们推介是直接使用默认级别就好

二、Bgrewriteaof 命令手动重写

命令格式：

127.0.0.1:6379> BGREWRITEAOF

三、快照

默认情况下，redis会开启自动快照功能，快照保存的文件为数据目录下的dump.rdb（可指定名称），自动快照促发条件可自行在配置文件中进行修改（save 900 1），或者可以执行 save 或者 bgsave 命令进行手动保存快照

快照文件将在子进程将快照文件中的内容全部写入磁盘后，重新创建一个新文件来替代老文件

==========================================================================

redis主从配置

主：master

# vim /usr/local/redis/redis.conf

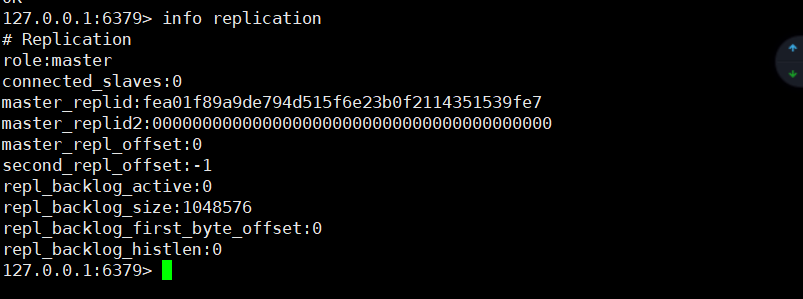
requirepass 123 将该行注释打开，并修改好自己的redis登陆密码

# bind 127.0.0.1 找到该行，并将其注释，开启允许远程访问

masterauth 123 设置master连接密码

127.0.0.1:6379> auth 123

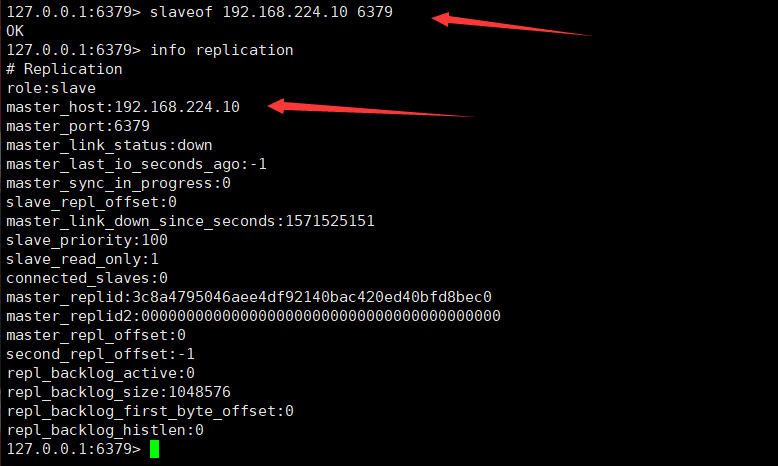
127.0.0.1:6379> info replication



从：slave

127.0.0.1:6379> info replication 查看当前redis主从属性

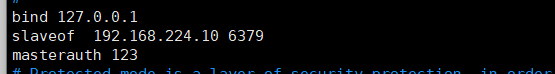
127.0.0.1:6379> slaveof host port 临时配置主从关系，重启后失效



# vim /usr/local/redis/redis.conf

slaveof host port 将host和port修改为对应的IP和端口，重启redis即可永久开启主从关系

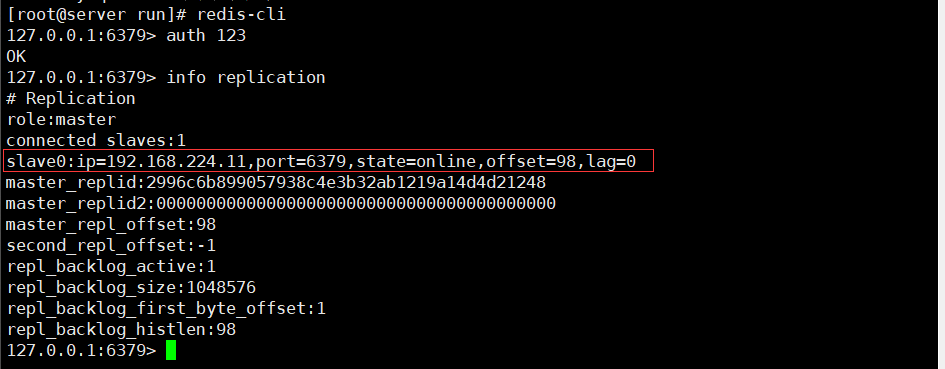
masterauth 123 如果master有开启密码验证，需打开这行注释，并在后方添加上master指定的密码



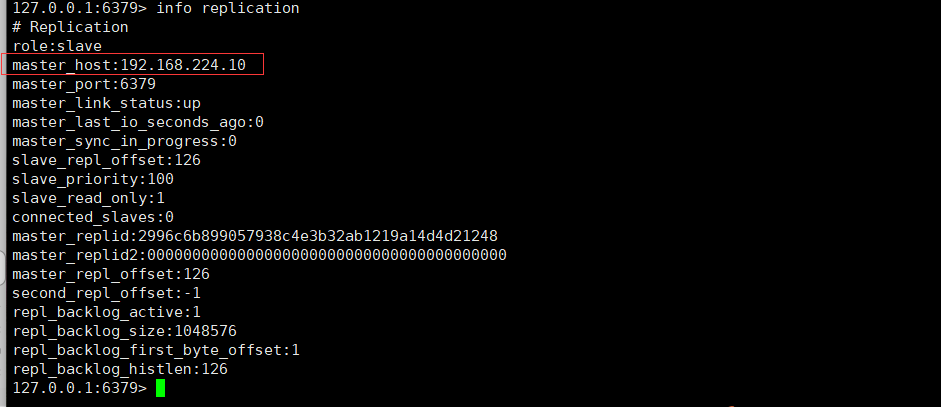
# /etc/init.d/redis start

redis读写分离介绍

redis中，只要实现了主从复制关系的，就已经满足了读写分离，其中master可读写，slave只能读

主:

从:

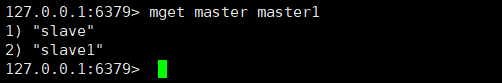


在主输入信息，

127.0.0.1:6379> mset master 'slave' master1 'slave1'

在从查看：

127.0.0.1:6379> mget master master1



end