**redis运维手册**

****

**亚信科技（南京）有限公司**

**2015年10月**

文档信息

|  |  |  |  |
| --- | --- | --- | --- |
| 项目名称 |  | | |
| 项目经理 |  | 文档版本 | 1.0 |
| 质量评审方法 |  |  |  |
| 文档编写人 |  | 编写日期 |  |
| 文档评审人 |  | 评审日期 |  |

分发

| From | Date | Phone/Fax |
| --- | --- | --- |
|  |  |  |
|  |  |  |

| To | Action\* | Due Date | Phone/Fax |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\* Action Types: Approve, Review, Inform, File, Action Required, Attend Meeting, Other (please specify)

文档变更历史

| 版本号 | 版本日期 | 修改人 | 变更描述 |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

版权说明

|  |
| --- |
| All rights reserved. Neither the whole nor any parts of this document may be reproduced, stored in any retrieval system or transmitted, in any form or by any means, without the prior written consent of the copyright owner. |

Copyright © 2015 by

All rights reserved.

# 安装

## 下载redis安装包

1、redis-stable.tar.gz下载

<http://www.redis.cn/download.html>

## 机器准备

本次采用分布式模式分别安装在以下3台机器上：

|  |
| --- |
| 133.37.135.219 |
| 133.37.135.220 |
| 133.37.135.221 |

## 创建用户

useradd -d /home/redis redis

passwd redis

## 创建安装目录

mkdir /usr/lib/redis

chown -R redis: redis /usr/lib/redis

## 安装解压包

将下载的安装包拷贝到创建的redis安装目录下,依次执行如下命令

tar -zxvf redis-3.0.5.tar.gz

cd redis-3.0.5

make

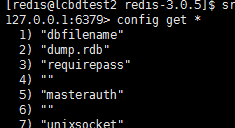
## 配置和优化

### Redis配置文件

redis可以在没有配置文件的情况下通过内置的配置来启动，但是这种启动方式只适用于开发和测试。

合理的配置Redis的方式是提供一个Redis配置文件，这个文件通常叫做redis.conf。

可以通过config get \* 获得所有的配置项的key及相应的值



### redis配置参数

配置参数可参考<http://www.runoob.com/redis/redis-conf.html>

1.bind

指定redis可接受的IP地址，不设置将处理所有地址发来的请求.

2.requirepass

设置客户端连接后进行任何其他指令前需要使用的密码

3.port

端口 默认为6379

4.timeout

客户端连接的超时时间,单位为秒,超时后会关闭连接,0永不超时

5.loglevel

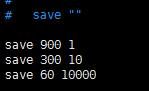
日志级别，默认为verbose，总共支持四个级别：debug、verbose、notice、warning

6.日志记录方式



7.redis数据持久化

7.1rdb方式



 Redis默认配置，分别表示900秒（15分钟）更改的键大于1或300秒（5分钟）内更改的键大于10或60秒内更改的键大于10000的时候就将数据同步到数据文件。

也可以手动执行SAVE或BGSAVE命令主动进行快照操作。

* **SAVE命令：**当执行SAVE命令时，Redis同步进行快照操作，期间会阻塞所有来自客户端的请求，所以放数据库数据较多时，应该避免使用该命令；
* **BGSAVE命令：** 从命令名字就能看出来，这个命令与SAVE命令的区别就在于该命令的快照操作是在后台异步进行的，进行快照操作的同时还能处理来自客户端的请求。执行BGSAVE命令后Redis会马上返回OK表示开始进行快照操作，如果想知道快照操作是否已经完成，可以使用LASTSAVE命令返回最近一次成功执行快照的时间，返回结果是一个Unix时间戳。
* Redis默认会将快照文件存储在Redis当前进程的工作目录的dump.rdb文件中，可以通过配置文件中的dir和dbfilename两个参数分别指定快照文件的存储路径和文件名，例如：

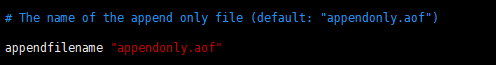




7.2 AOF方式



设置appendonly 为yes,开启之后，Redis每执行一条写命令就会将该命令写入硬盘中的AOF文件。AOF文件保存路径和RDB文件路径是一致的，都是通过dir参数配置，默认文件名是：appendonly.aof，可以通过配置appendonlyfilename参数修改，例如：



### 集群配置

1、redis分片方式

redis分片方式主要分有三种:客户端实现数据分片、服务端实现数据分片、通过代理服务器实现数据分片。

**客户端数据分片：**

即客户端自己计算数据的key应该在哪个机器上存储和查找，此方法的好处是降低了服务器集群的复杂度，客户端实现数据分片时，服务器是独立的，服务器之前没有任何关联。多数redis客户端库实现了此功能，也叫sharding,这种方式的缺点是客户端需要实时知道当前集群节点的联系信息，同时，当添加一个新的节点时，客户端要支持动态sharding.，多数客户端实现不支持此功能，需要重启redis

**服务端数据分片：**

其理论是，客户端随意与集群中的任何节点通信，服务器端负责计算某个key在哪个机器上，当客户端访问某台机器时，服务器计算对应的key应该存储在哪个机器，然后把结果返回给客户端，客户端再去对应的节点操作key，是一个重定向的过程，此方式是redis3.0正在实现，目前处于beta版本， Redis 3.0的集群同时支持HA功能，某个主节点挂了后，其从节点会自动接管。常用的有redis-cluster,这里我也是采用的redis-cluster配置的.

**代理服务器数据分片：**

此方式是借助一个代理服务器实现数据分片，客户端直接与proxy联系，proxy计算集群节点信息，并把请求发送到对应的集群节点降低了客户端的复杂度，需要proxy收集集群节点信息。常用的有Twemproxy,codis等。

2、redis-cluster配置

**要让集群正常运作至少需要三个主节点**， 不过在刚开始试用集群功能时， 强烈建议使用六个节点： 其中三个为主节点， 而其余三个则是各个主节点的从节点。

**首先配置文件的参数需要进行相应的修改，**

port 7000

cluster-enabled yes

cluster-config-file nodes.conf

cluster-node-timeout 5000

appendonly yes

文件中的 cluster-enabled 选项用于开实例的集群模式， 而 cluster-conf-file 选项则设定了保存节点配置文件的路径， 默认值为nodes.conf 。

刚开始安装的时候，我在219、220、221三台机器上各装了redis，这3个当作主节点。然后各拷贝一份redis.conf文件，创建一个子目录，修改文件中的端口号（默认为6379.修改为7000）.然后启动6个redis实例

例:在220机器上



有两个文件夹，node文件夹存放的是一个redis.conf文件，其中把端口号改成了7000

Redis-3.0.5是安装的redis.启动的时候命令如下:

Redis-3.0.5/src/redis-server redis-3.0.5/redis.conf

Redis-3.0.5/src/redis-server node/redis.conf

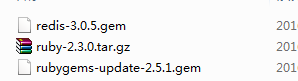
这样就启动了两个redis实例。

类似的，现在启动3个机器里的6个redis实例。

然后执行如下命令:

src/redis-trib.rb create --replicas 1 133.37.135.219:6379 133.37.135.220:6379 133.37.135.221:6379 133.37.135.219:7000 133.37.135.220:7000 133.37.135.221:7001

注:这里要注意的是redis-trib.rb是一个ruby程序,所以需先安装ruby程序、rubygems和ruby与redis接口程序.



这样redis-trib.rb才能正确执行.(本来用的是linux自带的ruby，但是由于版本过低，运行出现错误，所以重新装个高版本的ruby)

**Ruby安装命令如下：**

tar -zxvf ruby-2.3.0.tar.gz

cd ruby-2.3.0

./configure

make

sudo make install

**rubygems安装命令如下：**

gem install rubygems-update-2.5.1.gem

**redis与ruby接口程序安装命令**

gem install redis-3.0.5.gem

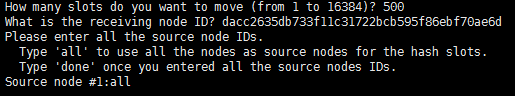
### 重新分片

执行如下命令



只需要指定集群中其中一个节点的地址， redis-trib 就会自动找到集群中的其他节点，

提示你移动哈希槽的数量，比如我们移动500个，输入500，再输入节点的id



然后redis会从其他主节点各取一部分哈希槽，凑够500个，移动到目标节点上。

### 添加删除节点

a.添加节点

1、添加主节点

首先需再启动个redis实例，比如启动个133.37.135.219 7001实例，再执行redis-trib.rb add-node 133.37.135.219:7001 133.37.135.219:6379 命令即可，其中133.37.135.219:7001为新增节点，133.37.135.219:6379可为集群任一个旧节点

2、添加从节点

redis-trib.rb add-node --slave --master-id dacc2635db733f11c31722bcb595f86ebf70ae6d 133.37.135.219:7001 133.37.135.219:6379

--slave，表示添加的是从节点

--master-id dacc2635db733f11c31722bcb595f86ebf70ae6d,主节点的node id

133.37.135.219:7001,新节点

133.37.135.219:6379集群任一个旧节点

b.删除节点

1、删除从节点

redis-trib.rb del-node  133.37.135.219:7001  9c240333476469e8e2c8e80b089c48f389827265

2、删除主节点

如果主节点有从节点，将从节点转移到其他主节点

如果主节点有slot，去掉分配的slot，然后在删除主节点

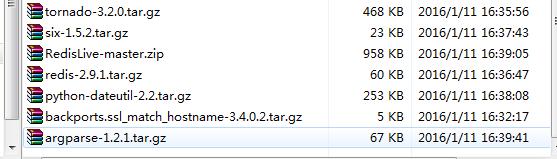
### 集群监控

一.redislive监控

RedisLive是一款用Python编写的Redis图形监控工具，RedisLive以可视化的方式展示了Redis实例中的数据，分析查询模式和峰值.

1. 安装环境

首先，需安装python,最好为2.7.3版本，我用linux默认版本安装的时候，出现错误.其次，还需安装如下几个安装包。

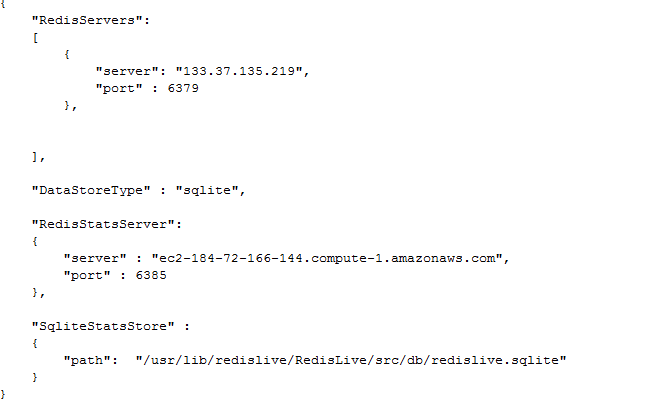


具体可参考<http://sofar.blog.51cto.com/353572/1369266>

这个页面提供了各个安装包的地址与安装配置，(其中argparse-1.2.1.tar.gz包的连接地址是谷歌，国内上不了，可以百度一下)

1. redislive配置

修改redis-live.conf文件



 在RedisServers中设置需要监控的redis-server，DataStoreType决定使用那种类型的数据存储，  
如果是redis，使用RedisStatsServer作为数据存储的目标，如果是sqlite，使用SqliteStatsStore作为数据存储的目标。

开启监控脚本

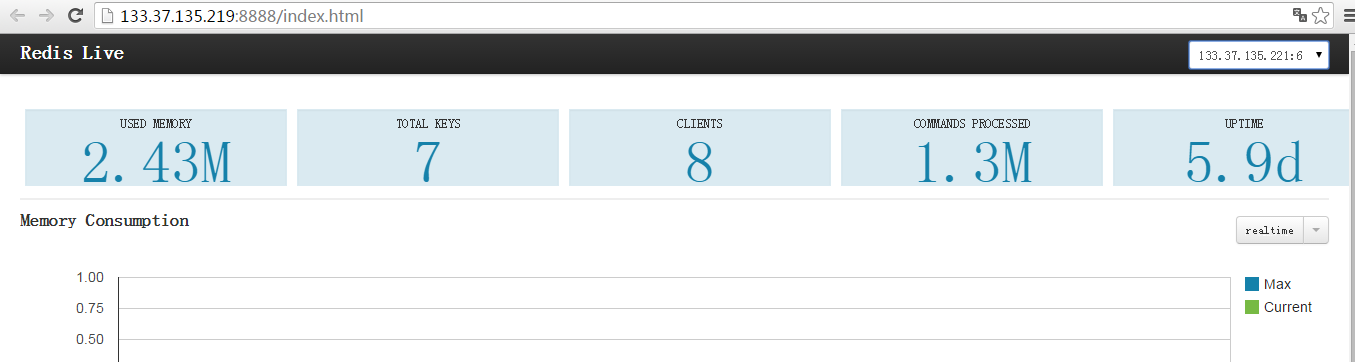
redis-monitor.py --duration 120 &

开启webserver

redis-live.py &

在浏览器中输入地址测试，因为我是在219机器上装的redislive,redislive默认端口为8888

所以地址输出133.37.135.219:8888/index.html



二、命令行监控



结果会返回 Server、Clients、Memory、Persistence、Stats、Replication、CPU、Keyspace 8个部分。从info大返回结果中提取相关信息，就可以达到有效监控的目的。各个参数的含义可参考<http://www.cnblogs.com/kaituorensheng/p/3979298.html>

# 2.启动及测试

## 启动

redis服务端的启动命令

[redis@lcbdtest2 redis-3.0.5]$ src/redis-server redis.conf

使用的话，启动redis客户端，命令参考如下:

[redis@lcbdtest2 redis-3.0.5]$ src/redis-cli -c –p 6379

## 停止

命令参考如下:

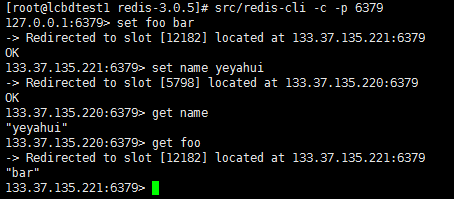
[redis@lcbdtest2 redis-3.0.5]$ src/redis-cli –p 6379 shutdown

## 测试

1.命令行

a. redis-cli

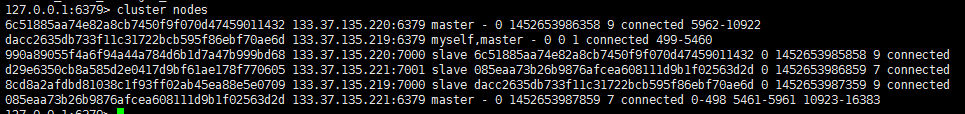
测试 Redis 集群比较简单的办法就是使用redis-cli连接服务端



redis-cli 对集群的支持是非常基本的， 所以它总是依靠 Redis 集群节点来将它转向（redirect）至正确的节点。

b.redis-cluster

redis-cluster一些命令可以对集群进行操作、查看,比如cluster nodes就是查看集群的节点信息



其他命令可参考<http://blog.51yip.com/nosql/1726.html/comment-page-1>

2.API

2.1导包

Java连接redis，首先需引入两个jar包

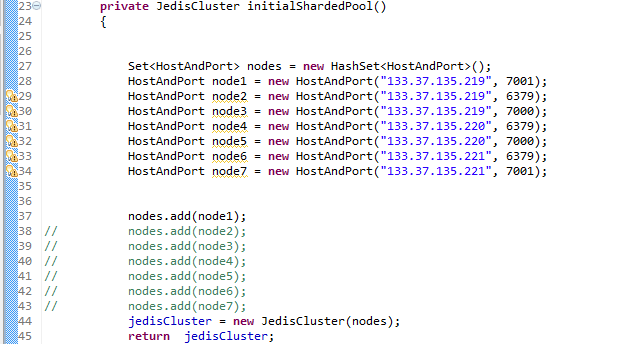


2.2类的使用

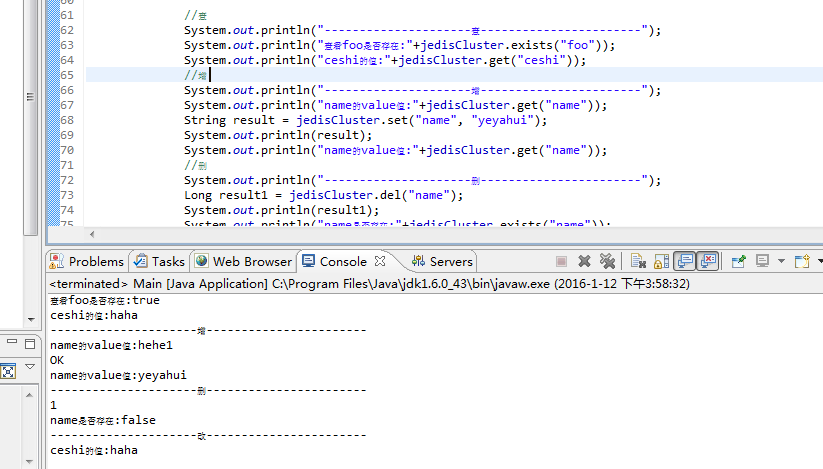
Java连接redis集群的类为

JedisCluster类

然后配置连接信息



jedisCluster只要添加集群中任意一个节点，就可以连接到redis集群中，



jedisCluster相关API目前只有英文的，而且没有详细说明,里面具体的方法还需要多用用，多了解了解。<http://javadox.com/redis.clients/jedis/2.5.1/redis/clients/jedis/JedisCluster.html>

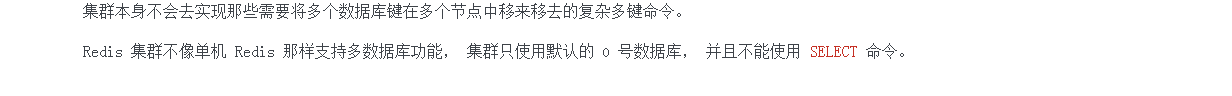
# 3.Redis介绍

## Redis分库

Redis默认配置了16个库，分库命令如下:

表示选择第一个库

不过，现在redis集群并不支持分库,只是默认选择第一个库





## redis特点

Redis本质上是一个Key-Value类型的内存数据库, 整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据flush到硬盘上进行保存。因为是纯内存操作，Redis的性能非常出色，每秒可以处理超过 10万次读写操作，是已知性能最快的Key-Value DB。

Redis的出色之处不仅仅是性能，Redis最大的魅力是支持保存多种数据结构，此外单个value的最大限制是1GB，支持的数据类型主要有:string、list、set、sorted set、hash五种类型

REST使用帮助文档：<http://www.redis.cn/commands.html> <http://doc.redisfans.com/>

## Redis过期

Redis允许为每一个key设置不同的过期时间，当它们到期时将自动从[服务器](http://cpro.baidu.com/cpro/ui/uijs.php?adclass=0&app_id=0&c=news&cf=1001&ch=0&di=128&fv=20&is_app=0&jk=9a60020313d44961&k=%B7%FE%CE%F1%C6%F7&k0=%B7%FE%CE%F1%C6%F7&kdi0=0&luki=1&mcpm=0&n=10&p=baidu&q=78096078_cpr&rb=0&rs=1&seller_id=1&sid=6149d413302609a&ssp2=1&stid=9&t=tpclicked3_hc&td=2307574&tu=u2307574&u=http%3A%2F%2Fwww%2Eredis%2Ecn%2Fdocumentation%2Ehtml&urlid=0)上删除

命令参考如下:

127.0.0.1:6379> expire num 30

(integer) 1

返回1表示设置成功，返回0表示不能设置过期时间 30代表过期时间为30秒

## Redis事务

redis的事务是一组命令的集合。事务同命令一样都是redis的最小执行单元.

首先需要multi命令来开始事务，用exec命令来执行事务。

127.0.0.1:6379> set num 1

OK

127.0.0.1:6379> set num1 1

OK

127.0.0.1:6379> multi

OK

127.0.0.1:6379> incr num

QUEUED

127.0.0.1:6379> incr num1

QUEUED

127.0.0.1:6379> exec

1) (integer) 2

2) (integer) 2

multi代表事务的开始，返回ok表示成功；incr num,incr num1命令发出后并没执行而是被放到了队列中, 调用exec后俩个命令被连续的执行，最后返回的是两条命令执行后的结果.  
我们可以调用discard命令来取消一个事务，接上面的例子

127.0.0.1:6379> multi

OK

127.0.0.1:6379> incr num

QUEUED

127.0.0.1:6379> incr num1

QUEUED

127.0.0.1:6379> discard

OK

127.0.0.1:6379> get num

"2"

127.0.0.1:6379> get num1

"2"

可以发现这次incr num incr num1都没被执行。discard命令其实就是清空事务的命令队列并退出事务上下文。

如果事务执行过程中存在失败的情况下（某一个命令执行失败后其他命令会继续执行）

例:

127.0.0.1:6379> set num2 hehe

OK

127.0.0.1:6379> multi

OK

127.0.0.1:6379> incr num

QUEUED

127.0.0.1:6379> incr num1

QUEUED

127.0.0.1:6379> incr num2

QUEUED

127.0.0.1:6379> exec

1) (integer) 3

2) (integer) 3

3) (error) ERR value is not an integer or out of range

当前例子中num与num1都增加了1，而num2不是数字，所以会报错  
注意：redis不支持回滚操作，导致redis的错误异常需要开发人员处理。

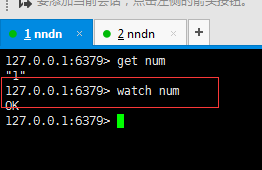
Wactch命令

比如说有两个client端，同时操作incr num这个命令，希望的结果是从1变成3， 但是很有可能两个client的get num，取到都是1，造成最终加两次结果却是2。主要问题我们没有对共享资源num的访问进行任何的同步。也就是说redis没提供任何的加锁机制来同步对num的访问。

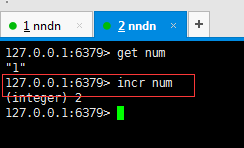
我们可以通过watch来实现加锁

例:

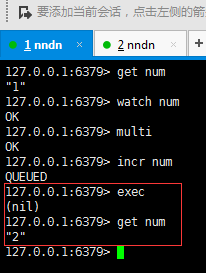
首先在A客户端watch num



再在B客户端对num执行+1操作



再在A客户端执行相应的事务操作



发现事务会失败，watch 命令会监视给定的key，当exec时候如果监视的key从调用watch后发生过变化，则整个事务会失败。

Redis事务特点:

1. 批量操作在发送 EXEC 命令前被放入队列缓存
2. 收到 EXEC 命令后进入事务执行，事务中任意命令执行失败，其余的命令依然被执行
3. 在事务执行过程，其他客户端提交的命令请求不会插入到事务执行命令序列中