# 前提概述

为什么要用？

因为Redis交换数据快，所以在服务器中常用来存储一些需要频繁调取的数据，这样可以大大节省系统直接读取磁盘来获得数据的I/O开销，更重要的是可以极大提升速度。

有什么用处？使用场景？

一：缓存——热数据

二：计数器 统计点击数等应用。由于单线程，可以避免并发问题，保证不会出错，而且100%毫秒级性能！

三：队列 相当于消息系统，ActiveMQ, 由于redis把数据添加到队列是返回添加元素在队列的第几位，所以可以做判断用户是第几个访问这种业务、个人觉得简单用一下还行

四：位操作（大数据处理）用于数据量上亿的场景下，例如几亿用户系统的签到，去重登录次数统计，某用户是否在线状态等等。

原理是：redis内构建一个足够长的数组，每个数组元素只能是0和1两个值，然后这个数组的下标index用来表示我们上面例子里面的用户id（必须是数字哈），那么很显然，这个几亿长的大数组就能通过下标和元素值（0和1）来构建一个记忆系统，上面我说的几个场景也就能够实现。用到的命令是：setbit、getbit、bitcount

五：分布式锁与单线程机制

验证前端的重复请求（可以自由扩展类似情况），可以通过redis进行过滤：每次请求将request Ip、参数、接口等hash作为key存储redis（幂等性请求），设置多长时间有效期，然后下次请求过来的时候先在redis中检索有没有这个key，进而验证是不是一定时间内过来的重复提交

秒杀系统，基于redis是单线程特征，防止出现数据库“爆破”

全局增量ID生成，类似“秒杀”

六：最新列表

例如新闻列表页面最新的新闻列表，如果总数量很大的情况下，尽量不要使用select a from A limit 10这种low货，尝试redis的 LPUSH命令构建List，一个个顺序都塞进去就可以啦。不过万一内存清掉了咋办？也简单，查询不到存储key的话，用mysql查询并且初始化一个List到redis中就好了。

七：排行榜 谁得分高谁排名往上。命令：ZADD（有续集，sorted set）

使用它的好处和弊端？

使用的是什么协议？

1、NoSQL (NoSQL= Not Only SQL),是一项全新的数据库理念，泛指非关系型的数据库

做大数据一定要知道NoSQL

2、互联网三高架构：高并发、高性能、高可用，简称三高（3H）

3、集群和分布式区别

**例1**：就比如新浪网，访问的人多了，他可以做一个群集，前面放一个响应服务器，后面几台服务器完成同一业务，如果有业务访问的时候，响应服务器看哪台服务器的负载不是很重，就将给哪一台去完成。

**例 2**如果任务由10个子任务组成，每个子任务单独执行需1小时，则在一台服务器上执行该任务10小时。

采用分布式方案：提供10台服务器，每台服务器只负责处理一个子任务，不考虑子任务间的依赖关系，执行完这个任务只需一个小时。(这种工作模式的一个典型代表就是Hadoop的Map/Reduce分布式计算模型）

而采用集群方案：同样提供10台服务器，每台服务器都能独立处理这个任务。假设有10个任务同时到达，10个服务器将同时工作，1小时后，10个任务同时完成，这样，整身来看，还是1小时内完成一个任务！

例 3小饭店原来只有一个厨师，切菜洗菜备料炒菜全干。后来客人多了，厨房一个厨师忙不过来，又请了个厨师，两个厨师都能炒一样的菜，这两个厨师的关系是集群。为了让厨师专心炒菜，把菜做到极致，又请了个配菜师负责切菜，备菜，备料，厨师和配菜师的关系是分布式，一个配菜师也忙不过来了，又请了个配菜师，两个配菜师关系是集群

集群一般被分为三种类型：高可用集群如RHCS、LifeKeeper等，负载均衡集群如LVS等、高性能运算集群;分布式应该是高性能运算集群范畴内。

分布式：不同的业务模块部署在不同的服务器上或者同一个业务模块分拆多个子业务，部署在不同的服务器上，解决高并发的问题

# redis结构

Redis使用c语言开发的一个开源高性能键值对(key-value)数据库

支持5种值数据类型：

1、字符串类型 string

2、散列类型：hash map相似

3、列表类型：list 可以重复

4、集合类型 set 不可重复 无序

5、有序集合类型 sortedset

运用场景

缓存(数据查询、短链接、新闻内容、商品内容等等)。最多用

聊天室的在线好友列表

任务队列，（秒杀、抢购、12306等等）

应用排行榜

网站访问统计

数据过期处理（可以精确到毫秒）

分布式集群架构中的session分离

# 安装和启动

## 安装

安装和启动redis 建议在linux使用，window版本不是官方出的，是微软出的（老师说的）

window

第一步 解压后有6个exe程序直接打开：redis-server服务（如mysql和oracle的服务）

第二步 不要关闭服务，打开redis-cli.exe进行使用命令

下面步骤都是在mac下操作的，linux（ubuntu）、coentos应该都是一样的

1）安装redis编译的c环境，yum install gcc-c++, （ubutun命令：sudo apt-get install gcc ）

2）下载redis 官网 https://redis.io/ 最新redis-5.0.3.tar.gz

3）解压 redis-5.0.3.tar.gz

4）进入解压后的目录，和nginx安装步骤不同的是此目录中本身就有Makefile目录了，

直接gcc编译命令：make进行编译

5）编译完后，指定安装目录，(这里要明白：naginx 是在编译makefile目录就已经指定了安装目录它的命令：直接：make install即可)

sudo make PREFIX=/usr/local/redis install 命令安装 redis到/usr/local/redis中

6）此时在/usr/local/redis下就有一个bin目录，bin目录下有以下文件6个文件

192:redis-5.0.3 wenwenliang$ ls -l /usr/local/redis/bin

total 8568

-rwxr-xr-x 1 root wheel 101564 3 4 11:46 redis-benchmark

-rwxr-xr-x 1 root wheel 1340076 3 4 11:46 redis-check-aof

-rwxr-xr-x 1 root wheel 1340076 3 4 11:46 redis-check-rdb

-rwxr-xr-x 1 root wheel 253064 3 4 11:46 redis-cli

lrwxr-xr-x 1 root wheel 12 3 4 11:46 redis-sentinel -> redis-server

-rwxr-xr-x 1 root wheel 1340076 3 4 11:46 redis-server

7）移动redis.conf（在第四步解压的文件目录下有）至bin目录下

命令：sudo cp redis.conf /usr/local/redis/bin/

8）修改redis.conf配置文件(注意权限问题，mac系统把文件移动usr/local变成root用户)

1、把文件权限改为：其它用户可以操作：sudo chmod 757 redis.conf

2、以下是redis.conf配置文件的修改内容

daemonize从no修改yes 表示后台启动

bind 127.0.0.1 进行注释：#bind 127.0.0.1 表示取消绑定本机才可以进行登录，允许所有ip远程登录

找到dir ./ 更改为：dir /Users/wenwenliang/redis/redis\_dbfiles 表示更改数据保存路径：快照dump.rdb(默认的保存名称，此rdb会定时保存数据到安装目录的根目录（默认值）)

找到 logfile “” 更改为 logfile /Users/wenwenliang/logs/myredis/redis\_log.log 表示更改log文件的路径，默认空，

找到protected-mode 把yes更改为no：表示可以远程进行修改插入等操作：远程操作：redis-cli -h 192.168.0.102 -p 6379

注意 只要重新指定目录必须存在，没有自己创建：dir和logfile两个要自己创建文件夹和文件名，root用户可以只改daemonize和bind即可

以上8步即完成安装

## 启动、关闭

前台启动和关闭

启动

运行命令：直接cd进入/usr/local/redis/bin/目录中运行redis-server

\_.\_

\_.-``\_\_ ''-.\_

\_.-`` `. `\_. ''-.\_ Redis 5.0.3 (00000000/0) 64 bit

.-`` .-```. ```\/ \_.,\_ ''-.\_

( ' , .-` | `, ) Running in standalone mode

|`-.\_`-...-` \_\_...-.``-.\_|'` \_.-'| Port: 6379

| `-.\_ `.\_ / \_.-' | PID: 2165

`-.\_ `-.\_ `-./ \_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' | http://redis.io

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' |

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

`-.\_ `-.\_\_.-' \_.-'

`-.\_ \_.-'

`-.\_\_.-'

启动后看到如上欢迎页面，但此窗口不能关闭，窗口关闭就认为redis也关闭了(类似Tomcat通过bin下的startup.bat的方式)这里不好，下面是后台启动

关闭

这里使用ctrl+c进行关闭，注意：关闭后会在目录中生成dump.rdb文件

如果出现以下内容关闭错误：因为redis关闭服务需要保存数据到本地(默认在本目录)/usr/local。。usr目录下普通用户是没有权限操作的

^C2197:signal-handler (1551679773) Received SIGINT scheduling shutdown...

2197:M 04 Mar 2019 14:09:33.875 # User requested shutdown... 用户请求关机…

2197:M 04 Mar 2019 14:09:33.875 \* Saving the final RDB snapshot before exiting. 出前保存最终RDB快照

2197:M 04 Mar 2019 14:09:33.875 # Failed opening the RDB file dump.rdb (in server root dir /usr/local/redis/bin) for saving: Permission denied打开RDB文件转储失败。用于保存的rdb(在服务器根目录/usr/local/redis/bin中):权限被拒绝

2197:M 04 Mar 2019 14:09:33.875 # Error trying to save the DB, can't exit. 试图保存数据库出错，无法退出。

2197:M 04 Mar 2019 14:09:33.875 # SIGTERM received but errors trying to shut down the server, check the logs for more information SIGTERM收到，但试图关闭服务器时出错，请检查日志以获取更多信息

只能强制性关闭：关闭当前终端窗口 或者开启另一个窗口进行kill -9 port进行杀死进程

后台启动和关闭

前提：安装步骤中的第7步和第8步有进行操作

192:bin wenwenliang$ redis-server redis.conf

2186:C 04 Mar 2019 14:00:29.744 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

2186:C 04 Mar 2019 14:00:29.744 # Redis version=5.0.3, bits=64, commit=00000000, modified=0, pid=2186, just started 刚刚启动

2186:C 04 Mar 2019 14:00:29.744 # Configuration loaded 配置已加载

出现以上信息证明启动成功

查看是否启动成功命令：ps aux|grep redis

关闭

第一种：通用强行终止Redis进程，但会导致redis持久化数据丢失

先查询pid 命令”ps aux | grep –I redis ” 然后杀死进程：kill -9 6799(查询到的pid)

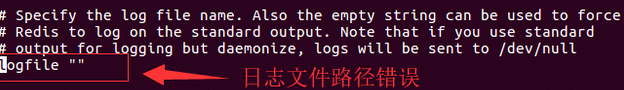
第二种：正确停止Redis的方式向Redis发送SHUTDOWN命令，方法为：（关闭默认端口）

进入安装的redis目录bin中使用命令: cd redis-cli shutdown

192:bin wenwenliang$ redis-cli shutdown #没有任何提示证明关闭成功

提示以下错误解决办法

1、(error) ERR Errors trying to SHUTDOWN. Check logs. 这个错误证明，配置文件redis.conf的logfile(配置日志文件保存路径)没有进行配置还是默认的空



解决办法：更改为logfile /Users/wenwenliang/logs/myredis/redis\_log.log；注意 下面图引用网上图片

../../../屏幕快照%202019-03-04%20下午2.45.31.png

配置好以后，还需要自己创建目录和文件，不然会报错：Can't open the log file: No such file or directory

mkdir -p /Users/wenwenliang/logs/myredis 使用参数-p递归创建目录

touch /Users/wenwenliang/logs/myredis/redis\_log.log新建文件

再进行启动即可，注意root用户是没有问题的

2、不使用root用户会出现Can't open the log file: Permission denied 此log文件没有权限，其实都是权限问题

1)sudo chmod 777 redis\_log.log redis\_log.log修改权限，改完这一步普通用户关闭的时候依然报错

2)其实：redis服务关闭后：会执行保存数据的操作(默认是保存根目录下dump.rdb)普通用户没有权限进行在usr/local/…目录下保存操作

解决办法：通过redis.conf配置文件修改dump.rdb的保存路径 ：找到dir ./ 更改为：dir /Users/wenwenliang/redis/……，目录不存在要自己创建，不然还是一样报错

## 登录

客户端本机登录

远程使用客户端：./redis-cli

192:bin wenwenliang$ redis-server redis.conf

192:bin wenwenliang$ redis-cli

127.0.0.1:6379> ping

PONG

127.0.0.1:6379> get a

"1"

127.0.0.1:6379> shutdown

not connected> exit

客户端远程登录 -a 后面是密码，有设置过密码才需要此参数 –h：host ；–p port

远程使用客户端：./redis-cli -h 192.168.0.102 -p 6379 -a Passw0rd

192:bin wenwenliang$ redis-cli -h 192.168.0.102 -p 6379

192.168.0.102:6379> get a

"1"

192.168.0.102:6379> ping

PONG

# aof 和Rdb

持久化方案两种：aof 和Rdb

Redis的所有数据都是保存到内存中的

Rdb：快照形式，定期把内存中当前时刻的数据保存到磁盘。Redis默认支持的持久化方案

在/usr/local/redis/bin中的redis-config文件中rdb搜索就会有：默认保存磁盘的方式：（不能关闭）

15分钟也就是900内只有1个key变化才会进行保存磁盘的操作

5分钟内也就是300秒中有10个key变化会进行保存磁盘

1分钟如果有10000个key变化会进行保存磁盘

aof形式：append only file 把所有对redis数据库操作的命令，增删改操作的命令，保存到文件中，数据库恢复时把所有的命令执行一边即可，默认不开启：频繁操作磁盘，性能下降

在/usr/local/redis/bin中的redis-config文件中aof

开启：appendonly no改为yes 即可 变为appendonly.aof文件作为保存数据文件，默认16个数据库，可以配置

# Redis命令

## 存储字符串

存储字符串string

字符串类型是Redis中最为基础的数据存储类型，它在Redis中是二进制安全的，这 便意味着该类型可以接受任何格式的数据，如JPEG图像数据或Json对象描述信息等。 在Redis中字符串类型的Value最多可以容纳的数据长度是512M

添加

set key value：设定key持有指定的字符串value，如果该key存在则进行覆盖操作。总是返回”OK”

获取

get key：获取key的value。如果与该key关联的value不是String类型，redis将返回错误信息，因为get命令只能用于获取String value；如果该key不存在，返回null。

修改

getset key value：先获取该key的值，然后在设置该key的值。

## 存储hash

添加 注意 值里面就可以重复添加了 field1 2 field 2 2

hset demo field 1 添加了：key为demo，值中的key为field，value为 1

获取

hget demo filed 获取单一 ，hkeys demo 获取所有demo中的key；hvals demo 获取所有demo的值；hgetall demo 获取所有值和键

删除

hdel demo field 删除 值中的key为field hk

## 存储list

存储list 先加进去在前面，后加进去在后面，可以重复

lpush demo 1 2 3 4 5 6 这个是从右开始添加，取值就变成了：654321

rpush demo abcdef 这个是从左边往右添加，取值就变成：abcdef

最后上面两个都在同一个变量中执行了取值就变成了：654321abcdef

查看

lrange demo 0 -1 查看全部

取值 和查有区别的

lpop demo 取左边第一个元素：6 ，注意取值完就没了，查看全部操作命令：就不会显示取完值的6

rpop demo取右边第一个元素：f，一样取完值就没了，往下执行查看全部命令

## 存储set

set 没有顺序，不能重复

增加

sadd demo a b c d e f

删除

srem demo a 删除 a

查看

smembers demo

例子：取出两个set中不相同的值

sdiff demo1 demo2 取出demo1中的值和demo2中不相同值

sunion demo1 demo2 取出demo1和demo2 中不相同的值，合并在一起显示

交集：sinter demo1 demo2 取出demo1和demo2相同的值

## 通用

删除

del key

加减

incr key加1 decr key减1

查看所有的key

keys \*

设置过期时间

expire key 10 10秒后删除此键值

查看元素的持久化

ttl key 值为：正数证明是有时间限制，为-1：永远存在，为-2：不存在的元素

例子 把持久化改为有时间限制再改为持久化

set key hello 改为60秒后删除 expire key 60 再改为持久化：Persist key

# java操作

## 连接

1、对象简单连接

第一步 导包 commons-pool2-2.3.jar和jedis-2.7.0.jar，两个包

第二步.获得连接对象

注意：如果是远程：

linux系统需要开放端口：命令：第一开放端口：/sbin/iptables -I INPUT -p tcp --dport 6379 -j ACCEPT，

第二保存命令：/etc/rc.d/init.d/iptables save

这两个命令执行了，才能连接到linux数据库

Jedis jedis = new Jedis("127.0.0.1",6379); //参数： ip,端口

第三步、获取数据

String userName=jedis.get("userName");

System.out.println(userName);

第四步、存储

jedis.set("addr", "北京");

//编码问题 存进去的是二进制，存进什么取什么

System.out.println(jedis.get("addr"));

2、通过连接池pool获得jedis连接对象

//0、创建池子的配置对象

JedisPoolConfig pc = new JedisPoolConfig();

pc.setMaxIdle(30); //最大闲置个数，多了30个关掉

pc.setMinIdle(10); //最小闲置个数，少了10个增加

pc.setMaxTotal(50); //最大连接数量

//1、创建一个redis的连接池

JedisPool pool = new JedisPool(pc,"127.0.0.1",6379);

//2、从池子中获取redis的连接资源

Jedis jedis=pool.getResource();

//3、操作数据库 对象 转换成json存

jedis.set("xx","yyyy");

//4、关闭资源

jedis.close();

pool.close();//真正开发不关闭池子

# 整合spring操作

## 自己整合

第一步 导包后xml配置：

<!-- 连接redis单机版 使用连接池-->

<bean id="jedisPool" class="redis.clients.jedis.JedisPool">

<constructor-arg name="host" value="192.168.25.153"/>

<constructor-arg name="port" value="6379"/>

</bean>

<bean id="jedisClientPool" class="com.taotao.jedis.JedisClientPool"/>

<!-- 连接redis集群版 -->

<!-- <bean id="jedisCluster" class="redis.clients.jedis.JedisCluster">

<constructor-arg>

<set>

<bean class="redis.clients.jedis.HostAndPort">

<constructor-arg name="host" value="192.168.25.153"/>

<constructor-arg name="port" value="7001"/>

</bean>

<bean class="redis.clients.jedis.HostAndPort">

<constructor-arg name="host" value="192.168.25.153"/>

<constructor-arg name="port" value="7002"/>

</bean>

<bean class="redis.clients.jedis.HostAndPort">

<constructor-arg name="host" value="192.168.25.153"/>

<constructor-arg name="port" value="7003"/>

</bean>

</bean>

</set>

</constructor-arg>

</bean>

<bean id="jedisClientCluster" class="com.taotao.jedis.JedisClientCluster"/> -->

第二步：封装工具类：一个接口类，两个实现类：单机版的redis，和集群版的redis

接口类 public interface JedisInterface

private String set(String key, String value); //添加String数据类型

单机实现类 需要关闭流操作

public String set(String key, String value) {

Jedis jedis = jedisPool.getResource();

String result = jedis.set(key, value);

jedis.close();

return result;

}

集群实现类 不用进行关闭流

public String set(String key, String value) {

return jedisCluster.set(key, value);

}

第三步 代码中的操作：查询：获取数据保存到redis，增删改：清除缓存

查询数据的时候使用缓存：

1. 用try块中保证没有redis情况下不影响代码的运行
2. 判断此key有无缓存：可用redis 的string 类型来保存，判断获取数据是否为null即可

//查询缓存并判断：如有直接使用缓存

try {

String json = jedisClient.hget(CONTENT\_KEY, cid + "");

//判断json是否为空

if (StringUtils.isNotBlank(json)) {

//把json转换成list

List<TbContent> list = JsonUtils.jsonToList(json, TbContent.class);

return list;

}

} catch (Exception e) {

e.printStackTrace();

}

//根据cid查询内容列表

TbContentExample example = new TbContentExample();

//设置查询条件

Criteria criteria = example.createCriteria();

criteria.andCategoryIdEqualTo(cid);

//执行查询

List<TbContent> list = contentMapper.selectByExample(example);

//向缓存中添加数据

try {

jedisClient.hset(CONTENT\_KEY, cid + "", JsonUtils.objectToJson(list));

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

缓存同步 只要在增删改中删除此数据即可

public E3Result addContent(TbContent content) {

//补全属性

content.setCreated(new Date());

content.setUpdated(new Date());

//插入数据

contentMapper.insert(content);

//缓存同步

jedisClient.hdel(CONTENT\_KEY, content.getCategoryId().toString());

return E3Result.ok();

## springDataRedis框架

第一步 导包

<!-- 集中定义依赖版本号 -->

<properties>

<spring.version>4.2.4.RELEASE</spring.version>

</properties>

<dependencies>

<!-- Spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jms</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

</dependency>

<!-- 缓存：springdataredis框架和redis -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.8.1</version>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-redis</artifactId>

<version>1.7.2.RELEASE</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- java编译插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.2</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

第二步 配置文件的配置

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:cache="http://www.springframework.org/schema/cache"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/cache

http://www.springframework.org/schema/cache/spring-cache.xsd">

<context:property-placeholder location="classpath\*:properties/\*.properties" />

<!--第一步 redis相关配置连接池 -->

<bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">

<property name="maxIdle" value="${redis.maxIdle}" /> <!-- 最大空闲数 -->

<property name="maxWaitMillis" value="${redis.maxWait}" /> <!-- 连接时的最大等待毫秒数 -->

<property name="testOnBorrow" value="${redis.testOnBorrow}" /> <!-- 在提取一个jedis实例时，是否提前进行验证操作：如果为true，则得到的jedis实例均是可用的 -->

</bean>

<!-- 第二步 spring data redis 包提供的 host-name：主机，port：端口，password：密码，引用 第一步的连接池-->

<bean id="JedisConnectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"

p:host-name="${redis.host}" p:port="${redis.port}" p:password="${redis.pass}" p:pool-config-ref="poolConfig"/>

<!-- 第三步 spring提供的模板(类中要注入) 引用 第二步 -->

<bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">

<property name="connectionFactory" ref="JedisConnectionFactory" />

</bean>

</beans>

redis-config.properties的配置

redis.host=106.12.106.21

redis.port=6379

redis.pass=w@123456

redis.database=0 #redis的第几个数据库，redis共16个

# 控制一个pool最多有多少个状态为idle(空闲的)的jedis实例

redis.maxIdle=300

# 表示当borrow(引入)一个jedis实例时，最大的等待时间，如果超过等待时间(毫秒)，则直接抛出JedisConnectionException；

redis.maxWait=3000

# 在borrow一个jedis实例时，是否提前进行validate操作；如果为true，则得到的jedis实例均是可用的

redis.testOnBorrow=true

第三步 类中的操作

第0步 测试用的两个注解：需 spring-test的包和junit

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration(locations="classpath:spring/applicationContext-redis.xml")

public class TestValue {

//第一步 注入配置文件的redisTemplate；

@Autowired//有泛型

private RedisTemplate redisTemplate;

//第二步 使用类型string

@Test//添加

public void setValue() {

redisTemplate.boundValueOps("name").set("使用springDataRedis存进去了");

}

@Test//获取

public void getValue() {

String str=(String) redisTemplate.boundValueOps("name").get();

System.out.println(str);

}

@Test//删除

public void deleteValue() {

redisTemplate.delete("name");

}

//第三步 使用类型set:没有顺序，不可重复

@Test

public void setSetValue() {

redisTemplate.boundSetOps("nameset").add("曹操");

redisTemplate.boundSetOps("nameset").add("刘备");

redisTemplate.boundSetOps("nameset").add("孙权");

}

@Test//获取

public void getSetValue() {

Set set=redisTemplate.boundSetOps("nameset").members();

System.out.println(set);

}

@Test//删除

public void deleteSetValue() {

//删除其中一个或多个，参数是可变参数

//redisTemplate.boundSetOps("nameset").remove("孙权");

//删除全部

redisTemplate.delete("nameset");

}

//第四步 使用类型list:有顺序,可重复，有下标

@Test//存储两个方式：右压栈和左压栈

public void setListValue() {

//右压栈：后添加的元素排在后边

redisTemplate.boundListOps("nameList1").rightPush("刘备");

redisTemplate.boundListOps("nameList1").rightPush("关羽");

redisTemplate.boundListOps("nameList1").rightPush("张飞");

//显示效果：[刘备, 关羽, 张飞]

//左压栈：后添加的元素排在前边

redisTemplate.boundListOps("nameList2").leftPush("刘备");

redisTemplate.boundListOps("nameList2").leftPush("关羽");

redisTemplate.boundListOps("nameList2").leftPush("张飞");

}

@Test//获取

public void getListValue() {

//索引：从哪里开始到结束，要查全部结束写大一点

List str= redisTemplate.boundListOps("nameList1").range(0, 10);

System.out.println(str);

//查询第几个元素:获取第二个元素

String str1=(String) redisTemplate.boundListOps("nameList1").index(1);

System.out.println(str1);

}

@Test//删除

public void deleteListValue() {

//删除某一个,第一个参数：1，要删除的个数(同样的值是否有多个)，第二个：要删除的值

redisTemplate.boundListOps("nameList1").remove(1, "张飞");

//删除全部

}

//第四步 使用类型hash:

@Test

public void setHashValue() {

redisTemplate.boundHashOps("nameHash").put("a", "唐僧");

redisTemplate.boundHashOps("nameHash").put("b", "八戒");

redisTemplate.boundHashOps("nameHash").put("c", "猴子");

redisTemplate.boundHashOps("nameHash").put("d", "沙僧");

}

@Test//获取

public void getHashValue() {

//获取所有的keys

Set keys=redisTemplate.boundHashOps("nameHash").keys();

System.out.println(keys);

//获取所有的values

List values=redisTemplate.boundHashOps("nameHash").values();

System.out.println(values);

//根据keys获取值

String str= (String) redisTemplate.boundHashOps("nameHash").get("a");

System.out.println(str);

}

@Test//删除

public void deleteHashValue() {

//删除其中一个小key或多个，参数是可变参数

redisTemplate.boundHashOps("nameHash").delete("a");

//删除全部：

}

}

# redis集群操作

redis集群原理

redis-cluster就叫集群名字

redis-cluste架构细节:

(1)所有的redis节点彼此互联(PING-PONG机制),内部使用二进制协议优化传输速度和带宽.

(2)节点的fail是通过集群中超过半数的节点检测失效时才生效.（票制）

(3)客户端与redis节点直连,不需要中间proxy层.客户端不需要连接集群所有节点,连接集群中任何一个可用节点即可

(4)redis-cluster把所有的物理节点映射到[0-16383]slot上,cluster 负责维护，最多16383

搭建集群服务器（至少3个主机还有3个备用）6个服务器，每一个主机从0-16383分配

## 搭建

条件：

Redis集群中至少应该有三个节点（服务器）。要保证集群的高可用，需要每个节点有一个备份机。

Redis集群至少需要6台服务器(3台服务器，每一台匹配备用机)

为什么要3台呢？因为节点判断是否损坏是投票机制，如果两台投票就是1比1)

搭建伪分布式步骤：可以使用一台虚拟机运行6个redis实例。需要修改redis的端口号7001-7006

第一步 找到安装并编译好bin目录(其实就是redis服务器)：进行复制6份：位置/usr/local/redis/ bin

随意找位置：改名为：redis01、redis02、redis03、redis04、redis05 、redis06

注意：要搭建集群每一个节点都必须是空的，目录中如有appendonly.aof和dump.rdb都需要删除掉

第二步 进入redis.conf更改配置：6个节点的端口号port和cluster

port 7001

cluster-enabled yes 默认是注释掉的，把#去掉即可

第三步 启动每一个redis实例

建议写一个可执行文件，这样就不用一次一次的去启动了 后缀名 .sh,

还需要更改权限：chomd u+x 文件名.sh

启动完查看：ps aux |grep redis

第四步 连接每一个节点，需要使用ruby脚本搭建集群。需要ruby的运行环境。

1安装ruby环境

yum install ruby

yum install rubygems ruby包管理器

2还需要安装ruby脚本运行使用的第三方库。就像java的第三方java

安装命令：gem install redis-3.0.0.gem #redis-3.0.0.gem是库的名字，老师提供了

3安装完后还需要参数命令：使用ruby脚本搭建集群

进入源代码redis中的src中，找后缀为rb

命令：cd redis-3.0.0/src

命令ll \*.rb

-rwxrwxr-x. 1 root root 48141 Apr 1 2015 redis-trib.rb 找到此文件移动到6个节点目录中

./redis-trib.rb create --replicas 1 192.168.25.153:7001 192.168.25.153:7002 192.168.25.153:7003 192.168.25.153:7004 192.168.25.153:7005 192.168.25.153:7006 #./redis-trib.rb脚本文件的名称 1是指每一个节点有一个备用机，还有ip和端口

运行完命令后一大堆信息，最后让你确认：输入 yes

注意 如果是6台服务器操作的要留意linux防火墙需要关闭

## 测试

使用redis-cli连接集群

命令：随便一个集群（备用的也可以）进入：./redis-cli –p 7004 -c必须有参数 –p和端口号 -c

进行保存操作：set a 123 就会出现：提示

Redurected to slot [15495] located at 192.168.25.156:7003 #此值保存在ip192.168.25.156的端口7003的redis服务器上

## java集群连接和操作

第一步：使用JedisCluster对象。需要一个Set<HostAndPort>参数。Redis节点的列表。

第二步：直接使用JedisCluster对象操作redis。在系统中单例存在。

第三步：打印结果

第四步：系统关闭前，关闭JedisCluster对象。

第一步：使用JedisCluster对象。需要一个Set<HostAndPort>参数。Redis节点的列表。

Set<HostAndPort> nodes = new HashSet<>();

nodes.add(new HostAndPort("192.168.25.153", 7001));

nodes.add(new HostAndPort("192.168.25.153", 7002));

nodes.add(new HostAndPort("192.168.25.153", 7003));

nodes.add(new HostAndPort("192.168.25.153", 7004));

nodes.add(new HostAndPort("192.168.25.153", 7005));

nodes.add(new HostAndPort("192.168.25.153", 7006));

JedisCluster jedisCluster = new JedisCluster(nodes);

// 第二步：直接使用JedisCluster对象操作redis。在系统中单例存在。

jedisCluster.set("hello", "100");

String result = jedisCluster.get("hello");

// 第三步：打印结果

System.out.println(result);

// 第四步：系统关闭前，关闭JedisCluster对象。

jedisCluster.close();

# 注意问题

注意 防止木马漏洞

1. 禁止Redis服务对公网开放，可通过修改redis.conf配置文件中的"#bind 127.0.0.1" ，去掉前面的"#"即可（Redis本来就是作为内存数据库，只要监听在本机即可）；

2. 设置密码访问认证，可通过修改redis.conf配置文件中的"requirepass" 设置复杂密码 （需要重启Redis服务才能生效）；

3. 对访问源IP进行访问控制，可在防火墙限定指定源ip才可以连接Redis服务器；

4. 修改Redis默认端口，将默认的6379端口修改为其他端口；

5. 禁用config指令避免恶意操作，在Redis配置文件redis.conf中配置rename-command项"RENAME\_CONFIG"，这样即使存在未授权访问，也能够给攻击者使用config 指令加大难度；

6. Redis使用普通用户权限，禁止使用 root 权限启动Redis 服务，这样可以保证在存在漏洞的情况下攻击者也只能获取到普通用户权限，无法获取root权限；

客户端登录 -a 后面是密码，有设置过密码才需要此参数

远程使用客户端：./redis-cli -h 127.0.0.1 -p 6379 -a Passw0rd