### redis学习参考

1. Redis远程客户端命令行操作

### <http://www.cnblogs.com/stephen-liu74/archive/2012/03/14/2349815.html>

1. Api

<http://www.yiibai.com/redis/>

1. Redis命令及对应jedis操作(String)

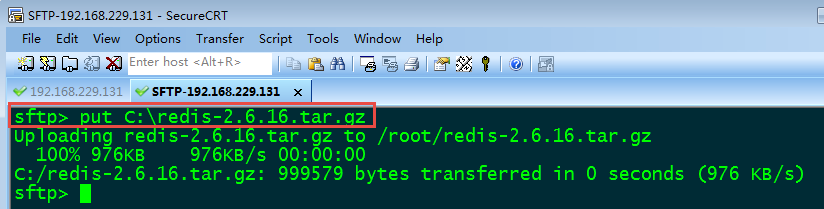
<http://blog.csdn.net/wtyvhreal/article/details/40980167>

### Linux系统简介与安装.avi

要点:

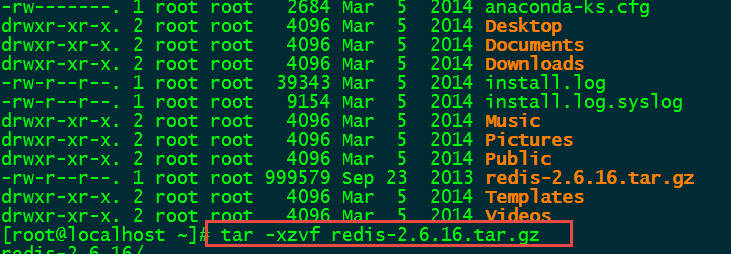
1. 上传redis的安装包到linux服务器

按atl + p 打开ftp窗口健入以下命令:



put F:\开发软件\虚拟机\linux软件\redis\redis-2.6.16.tar.gz

1. 解压安装包



1. 编译



1. 安装



1. 启动服务器

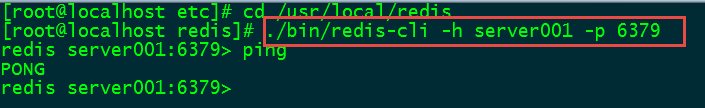
拷贝配置文件



启动



1. 客户端测试



### Redis数据类型string的命令行操作.avi

String数据类型的操作

**SET key value**

**设置key的值，如果该key已存在则覆盖原来的值.**

**返回值: 总是OK**

**GET key**

**获取key的值。**

**返回值: 对应key的value , 如果key不存在返回nil。**

**MSET key value、key value ….**

**原子性的执行多次set**

**返回值：总是OK**

**MGET key key …**

**执行多次get**

**返回值: 对应key的value, 某个key不存在返回nil**

**SETNX key value**

**执行set操作，如果key不存在设置key和value，如果key存在不会覆盖。**

**返回值: key存在返回0 , key不存在返回1**

**MSETNX key value , key value…**

**原子性执行多次MSETNX**

**返回值:所有MSETNX 成功返回1 ，只要有一个MSETNX失败返回0**

**Append key value**

**追加value到key的value后面,如果key不存在则吧value做为key的value.**

**返回值: 追加后的value的总字符数**

**Decr key**

**递减，如果key不存在.默认为0**

**返回值: 递减后的值，**

**Incr key**

**递增， 如果key不存在默认为0执行操作后为1.**

**返回值：递增后的值,如果value不为数值则返回错误信息**

**Decrby key decrement**

**递减指定的值，如果key不存在.默认为decrement**

**返回值: 递减后的值。**

**Incrby key increment**

**递增指定的值，如果key不存在默认为0执行操作后为increment**

**返回值：递增后的值,如果value不为数值则返回错误信息**

**Getset key**

**原子性的执行get 和set。**

**返回值: key原来的value, 不存在返回nil**

**Strlen key**

**返回key的value的字符个数**

**返回值: key的value字符的个数,不存在key返回0**

**Getrange key start end**

**截取字符串, 包括start 和end 。**

**返回值: 返回截取的字符串, key 不存在返回“”**

**Setrange key offset value**

**设置字符串**

**返回值: 返回设置后的字符个数.**

**Setbit key bit bitvlaue**

**设置二进制位的值**

**返回:设置后的结果**

**Getbit key bit**

**获取二进制位的值**

**返回: 二进制位的值**

### Redis客户端Jedis的使用.avi

创建工程

添加jar包

|  |
| --- |
| **package** jedistest;  **import** redis.clients.jedis.Jedis;  **public** **class** JedisDemo {  **public** **static** **void** main(String[] args) {  Jedis jedis = **new** Jedis("192.168.2.199");  // 测试客户端与redis服务器的联通性  String ping = jedis.ping();  System.**out**.println(ping);  }  } |

### Redis客户端Jedis操作String类型数据.avi

|  |
| --- |
| **package** jedistest;  **import** org.junit.Before;  **import** org.junit.Test;  **import** redis.clients.jedis.Jedis;  **public** **class** JedisDemo {  Jedis jedis = **null**;  @Before  **public** **void** init() {  jedis = **new** Jedis("192.168.2.199");  }  @Test  **public** **void** testString() {  // 插入一条string类型的数据  String res = jedis.set("jedis-s-key-01", "itcast ia the greatest it school");  System.**out**.println(res);  // 获取一条string类型的数据  String value = jedis.get("jedis-s-key-01");  System.**out**.println(value);  // 子字符串的获取  String value2 = jedis.getrange("jedis-s-key-01", 0, 5);  System.**out**.println(value2);  // 子字符串的替换  jedis.setrange("jedis-s-key-01", 0, "itheima");  String value3 = jedis.get("jedis-s-key-01");  System.**out**.println(value3);  jedis.setrange("jedis-s-key-01", 26, "trainningschool");  String value4 = jedis.get("jedis-s-key-01");  System.**out**.println(value4);  //如果偏移量超出字符串长度，则会自动补\0x00  jedis.setrange("jedis-s-key-01", 50, "very good");  String value5 = jedis.get("jedis-s-key-01");  System.**out**.println(value5);  //setnx会判断指定的key是否存在，如果已存在，则不会插入数据  **for** (**int** i = 0; i < 5; i++) {  jedis.setnx("jedis-s-key-0" + i, "000-" + i);  }  }  **public** **static** **void** main(String[] args) {  Jedis jedis = **new** Jedis("192.168.2.199");  // 测试客户端与redis服务器的联通性  String ping = jedis.ping();  System.**out**.println(ping);  }  } |

### Redis数据结构List的命令行操作.avi

Lpush key value

从List中插入数据

Lrange key start end

从List中获取数据

Lpushx key value

从List中插入数据如果key不存在不做任何操作

Lpop

从头弹出一个元素.

Llen

求list的长度

Lrem key count value

删除n个指定的value

Lset key index value

设置指定索引位置的值

Lindex key index

获取index位置的值

Ltrim key start end

截断list

Linsert key before | alter value insertValue

要某个值的之前之后插入一个值

Rpush key value ..

在List的尾部插入新元素

Rpush key value ..

在List的尾部插入新元素, 如查key不存在不执行任何操作.

Rpop key

从list尾部弹出一个元素.

RpopLpush source destination

原子性的从source尾部弹出一个元素插入到destination的头部

### Redis客户端Jedis操作List数据结构.avi

|  |
| --- |
| @Test  **public** **void** testList(){    //在链表的头部插入元素  // Long count = jedis.lpush("jedis-l-key-01", "zhangsan","lisi","wangwu","zhaoliu");  // System.out.println("插入的元素个数为： " + count);    List<String> res = jedis.lrange("jedis-l-key-01", 0, -1);  System.**out**.println("创建的链表中的元素为：");  **for**(String s:res){    System.**out**.println(s);  }    //在链表的中间插入新元素  /\* jedis.linsert("jedis-l-key-01", LIST\_POSITION.BEFORE, "lisi", "田七");    System.out.println("插入新元素之后的链表元素为： ");    List<String> res2 = jedis.lrange("jedis-l-key-01", 0, -1);  for(String s:res2){    System.out.println(s);  }\*/    Long count = jedis.lrem("jedis-l-key-01", 3, "zhangsan");  System.**out**.println("成功删除了" + count + "个zhangsan");  List<String> res3 = jedis.lrange("jedis-l-key-01", 0, -1);  **for**(String s:res3){  System.**out**.println(s);  }  } |

### Redis数据结构List的应用示例--任务队列-1.avi

### Redis数据结构List的应用示例--任务队列-2.avi

|  |
| --- |
| **package** jedistest;  **import** java.util.Random;  **import** java.util.UUID;  **import** redis.clients.jedis.Jedis;  **public** **class** TaskScheduler {  /\*\*  \* 任务生产者线程逻辑  \*  \* **@author**  \*  \*/  **static** **class** TaskProducer **implements** Runnable {  **public** **void** run() {  Jedis jedis = **new** Jedis("192.168.2.199");  System.**out**.println("任务生产者启动.......");  jedis.del("task-list");  **while** (**true**) {  UUID newTaskId = UUID.randomUUID();  jedis.lpush("task-list", newTaskId.toString());  System.**out**.println("生产者插入了一个新任务： " + newTaskId);  **try** {  **int** nextInt = **new** Random().nextInt(2);  Thread.sleep(1000 + nextInt \* 1000);  } **catch** (InterruptedException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  }  }  /\*\*  \* 任务处理者线程逻辑  \*  \* **@author**  \*  \*/  **static** **class** TaskWorker **implements** Runnable {  **public** **void** run() {  Jedis jedis = **new** Jedis("192.168.2.199");  System.**out**.println("任务处理者启动.......");  jedis.del("status-list");  **while** (**true**) {  **try** {  Thread.sleep(1000);  // 从task-list弹出一个任务，并插入到status-list队列中  String taskId = jedis.rpoplpush("task-list", "status-list");  // 处理任务的逻辑......  **int** nextInt = **new** Random().nextInt(9);  // 模拟任务处理成功的情况  **if** (nextInt % 4 != 0) {  // 从status-list队列中弹出这个处理成功的任务  jedis.lpop("status-list");  System.**out**.println(taskId + ": 处理成功，并从任务调度系统中彻底删除");  } **else** {  // 模拟任务处理失败的情况  jedis.rpoplpush("status-list", "task-list");  System.**out**.println(taskId + ": 处理失败，并从status-list中弹回task-list");  }  } **catch** (InterruptedException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  }  }  **public** **static** **void** main(String[] args) **throws** Exception {  **new** Thread(**new** TaskProducer()).start();  Thread.sleep(200);  **new** Thread(**new** TaskWorker()).start();  }  } |

### 9.Redis数据结构Hash的命令行操作.avi

Hset key field value

Hget key field

Hmset key field value ….

Hmget key field field..

Hgetall

### 10.Redis客户端Jedis操作Hash数据结构.avi

|  |
| --- |
| @Test  **public** **void** testHash(){    jedis.hset("jedis-h-key-01", "name", "zhangsan");    String name = jedis.hget("jedis-h-key-01", "name");  System.**out**.println(name);    HashMap<String, String> fields = **new** HashMap<String, String>();  fields.put("password", "123");  fields.put("age", "18");  jedis.hmset("jedis-h-key-01", fields);      Set<String> keys = jedis.hkeys("jedis-h-key-01");  System.**out**.println("所有的key如下：");  **for**(String key :keys){  System.**out**.println(key);    }    List<String> hvals = jedis.hvals("jedis-h-key-01");  System.**out**.println("所有的value如下： ");  **for**(String val:hvals){    System.**out**.println(val);  }    System.**out**.println("一次性取出所有的key-value对：");  Map<String, String> kvs = jedis.hgetAll("jedis-h-key-01");  Set<Entry<String, String>> entrySet = kvs.entrySet();  **for**(Entry<String, String> ent :entrySet){  System.**out**.println(ent.getKey() + " : " + ent.getValue());    }  } |

### 11.Redis数据结构Set的命令行操作.avi

Sadd key m m…

向set中添加元素

Smembers key

获取set中的所有元素

Scard key

求 set长度

SisMember key m

判断是否是一个成员

Spop

随机弹出一个成员

Srandmember key

也是随机返回一个成员。但不会删除

Srem key m m

删除多个成员

Smove source distination member

移动成员

Sdiff key key

求差集

Sdiffstore distination key key

求差集并保存到另一个set

Sinter key key

求交集

Sinterstore distination key key

求交集并保存

Sunion key key

求并集，去掉重复

Sunionstore dist key key

求并集并保存到另一个set

### 12.Redis客户端Jedis操作Set数据结构.avi

|  |
| --- |
| @Test  **public** **void** testSet(){    //创建一条set类型的数据并插入一些members  Long sadd = jedis.sadd("jedis-set-key-01", "java","c","c++","js","ruby");  System.**out**.println("创建了一个set类型的数据，并且插入"+sadd+"个成员");    //获取一条set类型的数据的成员  Set<String> members = jedis.smembers("jedis-set-key-01");  System.**out**.println("获取到的成员为：");  **for**(String m:members){  System.**out**.println(m);  }    jedis.sadd("jedis-set-key-02", "c#","c",".net","python","ruby");      //求两个集合的差集  Set<String> set12 = jedis.sdiff("jedis-set-key-01","jedis-set-key-02");    System.**out**.println("set01 减去 set02 的差集结果为：");    **for**(String s:set12){  System.**out**.println(s);  }        Set<String> set21 = jedis.sdiff("jedis-set-key-02","jedis-set-key-01");    System.**out**.println("set02 减去 set01 的差集结果为：");    **for**(String s:set21){  System.**out**.println(s);  }    //求两个集合的并集  Set<String> union12 = jedis.sunion("jedis-set-key-01","jedis-set-key-02");  System.**out**.println("set01和set02的并集结果为:");  **for**(String s:union12){  System.**out**.println(s);  }  //求两个集合的交集  } |

### 13.Redis数据结构SortedSet的命令行操作.avi

Zadd key score member ….

添加成员

Zincrby key score member

添加指定分数

Zcard key

求长度

Zcount key min max 或者 zcount key (min(max

求分数在min max之间的成员数量

Zrange key start top withscores

求在指定名次之间的成员,按分数从低到高排

Zrevrange key start top withscores

求在指定定名次之间成员.按分数从高到低排

Zrangebyscore key min max [withdscores][limit offset count]

求在指定分数之间成员 , 按分数从小到大排

Zrevrangebyscore key min max [withdscores][limit offset count]

求在指定分数之间成员 , 按分数从大到小排

Zrank key member

取排名

Zrevrank key member

取排名

Zscore key member

取成员分数

Zrem key member..

删除成员

Zremrangebyrank key start stop

按排名删除成员

Zremrangebyscore key start stop

按分数删除成员

### 14.Redis客户端Jedis操作SortedSet数据结构.avi

|  |
| --- |
| @Test  **public** **void** testSortedSet(){  HashMap<String, Double> scoreMembers = **new** HashMap<String, Double>();  scoreMembers.put("zhangsan", 100.00);  scoreMembers.put("lisi", 90.00);  scoreMembers.put("wangwu", 80.00);  scoreMembers.put("zhaoliu", 70.00);  scoreMembers.put("tianqi", 60.00);    jedis.zadd("jedis-zset-key-01", scoreMembers);    //获取指定名次区间的所有成员，顺序为分数的由低到高  Set<String> allMembers = jedis.zrange("jedis-zset-key-01", 0, -1);  System.**out**.println("所有的成员为：");  **for**(String m: allMembers){    System.**out**.println(m);  }      System.**out**.println("给赵六加40分之后，成员的排名情况为：");  //给指定的成员增加分数  jedis.zincrby("jedis-zset-key-01", 40, "zhaoliu");    //获取指定名次区间的所有成员及其关联的分数，顺序为分数的由低到高  Set<Tuple> zrangeWithScores = jedis.zrangeWithScores("jedis-zset-key-01", 0, -1);  **for**(Tuple t:zrangeWithScores){    System.**out**.println(t.getElement() + " : " +t.getScore());  }      System.**out**.println("按照分数由高到低的顺序打印排行榜：");    //获取指定名次区间的所有成员及其关联的分数，顺序为分数的由高到低  Set<Tuple> zrevrangeWithScores = jedis.zrevrangeWithScores("jedis-zset-key-01", 0, -1);  **for**(Tuple t:zrevrangeWithScores){    System.**out**.println(t.getElement() + " : " +t.getScore());  }  //扩展练习其他方法  } |

### 15.Redis数据结构SortedSet的应用示例--英雄联盟排行榜.avi

|  |
| --- |
| **package** jedistest;  **import** java.util.Random;  **import** java.util.Set;  **import** redis.clients.jedis.Jedis;  **import** redis.clients.jedis.Tuple;  /\*\*  \* 模拟英雄联盟游戏盒子的英雄排行榜实现  \*  \* **@author**  \*  \*/  **public** **class** LegendTopRank {  // 数据采集线程，获取英雄（游戏角色）出场数据并插入到redis数据库中  **static** **class** HeroScoreGenerate **implements** Runnable {  **public** **void** run() {  String[] heros = **new** String[] { "盲僧", "盖伦", "石头", "亚索", "女警", "光辉", "剑圣", "放逐" };  Random random = **new** Random();  Jedis jedis = **new** Jedis("192.168.2.199");  // 模拟不断产生新的英雄出场数据  **while** (**true**) {  **try** {  Thread.sleep(1000 + random.nextInt(10) \* 100);  // 随机挑选了一个英雄，表示用这个挑选出来的英雄参与了一场游戏  **int** index = random.nextInt(8);  String hero = heros[index];  // 在redis数据库中给相应的英雄增加分数  jedis.zincrby("TopRankChuChang", 1, hero);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }  }  // 获取排行榜单的线程  **public** **static** **void** main(String[] args) {  **new** Thread(**new** HeroScoreGenerate()).start();  Jedis jedis = **new** Jedis("192.168.2.199");  **int** i = 0;  // 不断地去获取最新的排行榜  **while** (**true**) {  **try** {  Thread.sleep(3000);  Set<Tuple> heros = jedis.zrevrangeWithScores("TopRankChuChang", 0, -1);  System.**out**.printf("第%s次获取排行榜：", i);  System.**out**.println();  **for** (Tuple t : heros) {  System.**out**.println(t.getElement() + " : " + t.getScore());  }  i++;  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }  } |

### 16.Redis针对key的通用操作功能.avi

Key pattern

获取key

Del key

删除key

Exists key

判断key是否存在

Move key db

移动key到数据库

Select db

选择数据库,共16个库

Rename key newkey

Renamenx key newkey

Persist key

持久化可超时的key

Expire key seconds

设置超时， 超过时间自已删除， 如果在删除之间该key 有更新将废除超时

Expire key timestamp

设置超时， 超过时间自已删除， 如果在删除之间该key 有更新将废除超时

Ttl key

查看还有多久超时

Randomkey

随机返回一个key

Type key

返回key对应的value的类型

String list set hash zset ， key不存在返回none

### 17.Redis数据持久化机制介绍.avi

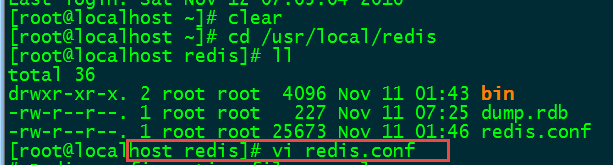
Rdb

aof

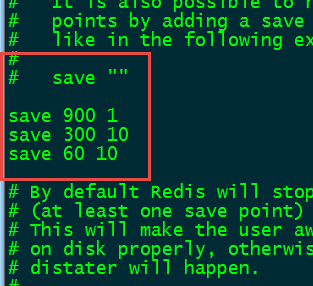
### 18.Redis数据持久化配置和测试.avi

默认是RDB

修改redis.conf配置rdb备份的频率



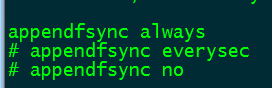
搜索 :/save



开启 AOF



配置AOF的备份频率

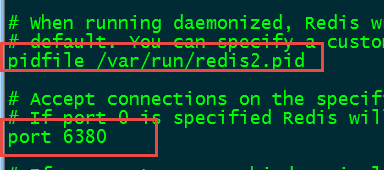


Rewrite

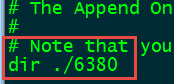
Bgrewriteaof

### 19.Redis集群之主从复制集群.avi

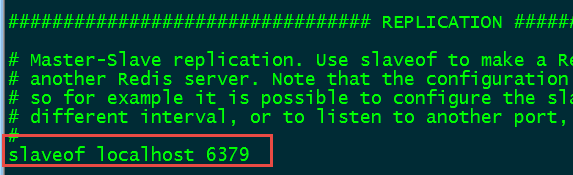
修改进程和端口



修改从服务器的快照存发目录



设置从服务器的主服务器ip 和端口



启动服务器





### 20.Redis集群之数据分片集群.avi

|  |
| --- |
| **package** jedistest;  **import** java.util.ArrayList;  **import** org.apache.commons.pool2.impl.GenericObjectPoolConfig;  **import** redis.clients.jedis.JedisShardInfo;  **import** redis.clients.jedis.ShardedJedis;  **import** redis.clients.jedis.ShardedJedisPool;  **public** **class** ShardClusterTest {    **public** **static** **void** main(String[] args) {    //poolConfig是连接池的配置参数  GenericObjectPoolConfig poolConfig = **new** GenericObjectPoolConfig();  //shards是分片集群中所有分片服务器信息列表,JedisShardInfo是分片服务器信息  ArrayList<JedisShardInfo> shards = **new** ArrayList<JedisShardInfo>();    //将集群中的两台shard服务器信息封装到两个JedisShardInfo对象中  JedisShardInfo shard1 = **new** JedisShardInfo("192.168.2.199", 6379);  JedisShardInfo shard2 = **new** JedisShardInfo("192.168.2.199", 6380);    //将分片服务器信息对象添加到分片服务器信息列表shards中  shards.add(shard1);  shards.add(shard2);    //创建一个带数据分片功能的jedis连接池  ShardedJedisPool shardedJedisPool = **new** ShardedJedisPool(poolConfig, shards);    //从连接池中获取一个(带数据分片功能的)jedis连接  ShardedJedis jedis = shardedJedisPool.getResource();    **for**(**int** i=0;i<1000;i++){  jedis.set("string-key-"+i, "1000" + i);  }  jedis.close();  shardedJedisPool.close();  }  } |