JedisPool 资源池优化及常见问题汇总

作者：zxniuniu
原文链接：https://ld246.com/article/1552815444506
来源网站：链滴

背景

合理的JedisPool资源池参数设置能为业务使用Redis保驾护航，本文将对JedisPool的使用、资源池参数进行详细说明，最后给出“最合理”配置。

一、使用方法

以官方的3.0.1为例子(Jedis Release)，Maven依赖如下：

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>3.0.1</version>
    <scope>compile</scope>
</dependency>
```

Jedis使用apache commons-pool2对Jedis资源池进行管理，所以在定义JedisPool时一个很重要的数就是资源池GenericObjectPoolConfig，使用方式如下，其中有很多资源管理和使用的参数(具体第二节)。注意：后面会提到建议用JedisPoolConfig代替GenericObjectPoolConfig

```
GenericObjectPoolConfig jedisPoolConfig = new GenericObjectPoolConfig();
jedisPoolConfig.setMaxTotal(..);
jedisPoolConfig.setMaxIdle(..);
jedisPoolConfig.setMinIdle(..);
jedisPoolConfig.setMaxWaitMillis(..);
```

JedisPool的初始化如下：

```
// redisHost和redisPort是实例的IP和端口，redisPassword是实例的密码
// timeout，这里既是连接超时又是读写超时，从Jedis 2.8开始有区分connectionTimeout和soTimeout的构造函数
JedisPool jedisPool = new JedisPool(jedisPoolConfig, redisHost, redisPort, timeout, redisPassword);

Jedis jedis = null;
try {
    jedis = jedisPool.getResource();
    //具体的命令
    jedis.executeCommand()
} catch (Exception e) {
    logger.error(e.getMessage(), e);
} finally {
    //注意这里不是关闭连接，在JedisPool模式下，Jedis会被归还给资源池。
    if (jedis != null)
        jedis.close();
}
```

二、参数说明

JedisPool保证资源在一个可控范围内，并且提供了线程安全，但是一个合理的GenericObjectPoolConfig配置能为应用使用Redis保驾护航，下面将对它的一些重要参数进行说明和建议：

在当前环境下，Jedis连接就是资源，JedisPool管理的就是Jedis连接。

1. 资源设置和使用

序号	参数名	含义	默认值	用建议
1	maxTotal	资源池中最大连接数	8	置建议见下节
2	maxIdle	资源池允许最大空闲的连接数	8	置建议见下节
3	minIdle	资源池确保最少空闲的连接数	0	置建议见下节
4	blockWhenExhausted	当资源池用尽后，调用者是否要	true	待。只有当为true时，下面的maxWaitMillis才会生效 议使用默认值
5	maxWaitMillis	当资源池连接用尽后，调用者的最大等待	-1：表示永不超时	间(单位为毫秒) 不建议使用默认值
6	testOnBorrow	向资源池借用连接时是否做连接有效性检测	false	ping)，无效连接会被移除 业务量很大时候建议设置为fa se(多一次ping的开销)。
7	testOnReturn	向资源池归还连接时是否做连接有效性检测	false	ping)，无效连接会被移除 业务量很大时候建议设置为fa se(多一次ping的开销)。
8	jmxEnabled	是否开启jmx监控，可用于监控	rue	建议开启，但应用本身也要开启

2.空闲资源监测

空闲Jedis对象检测，下面四个参数组合来完成，testWhileIdle是该功能的开关。

序号	参数名	含义	默认值	用建议
1	testWhileIdle	是否开启空闲资源监测	f	lse true
2	timeBetweenEvictionRunsMillis	空闲资源的检测周期(单	-1：不检测	为毫秒) 建议设置，周期自行选择，也可以默认 可以使用下面JedisPoolConfig中的配置
3	minEvictableIdleTimeMillis	资源池中资源最小空闲时间(	1000 * 60 * 30 = 30分钟	位为毫秒)，达到此值后空闲资源将被移除 根据自身业务决定，大部分默认值即可，也可以考虑使用下面JeidsPoolConfig中的配置
4	numTestsPerEvictionRun	做空闲资源检测时，每次的采	3	数 可根据自身应用连接数进行微调,如果设置为-1，就是对 有连接做空闲监测

为了方便使用，Jedis提供了JedisPoolConfig，它本身继承了GenericObjectPoolConfig设置了一些闲监测设置

```
public class JedisPoolConfig extends GenericObjectPoolConfig {
  public JedisPoolConfig() {
```

```
    // defaults to make your life with connection pool easier :)
    setTestWhileIdle(true);
    //
    setMinEvictableIdleTimeMillis(60000);
    //
    setTimeBetweenEvictionRunsMillis(30000);
    setNumTestsPerEvictionRun(-1);
     }
}
```

所有默认值可以从org.apache.commons.pool2.impl.BaseObjectPoolConfig中看到。

三、资源池大小(maxTotal)、空闲(maxIdle minIdle)设置建议

1.maxTotal：最大连接数

实际上这个是一个很难回答的问题，考虑的因素比较多：

- 业务希望Redis并发量
- 客户端执行命令时间
- Redis资源：例如 nodes(例如应用个数) * maxTotal 是不能超过redis的最大连接数。
- 资源开销：例如虽然希望控制空闲连接，但是不希望因为连接池的频繁释放创建连接造成不必靠开
。

以一个例子说明，假设:

- 一次命令时间（borrow|return resource + Jedis执行命令(含网络) ）的平均耗时约为1ms，一个接的QPS大约是1000
- 业务期望的QPS是50000

那么理论上需要的资源池大小是50000 / 1000 = 50个。但事实上这是个理论值，还要考虑到要比理值预留一些资源，通常来讲maxTotal可以比理论值大一些。

但这个值不是越大越好，一方面连接太多占用客户端和服务端资源，另一方面对于Redis这种高QPS服务器，一个大命令的阻塞即使设置再大资源池仍然会无济于事。

2. maxIdle minIdle

maxIdle实际上才是业务需要的最大连接数，maxTotal是为了给出余量，所以maxIdle不要设置过小否则会有new Jedis(新连接)开销，而minIdle是为了控制空闲资源监测。

连接池的最佳性能是maxTotal = maxIdle ,这样就避免连接池伸缩带来的性能干扰。但是如果并发量大或者maxTotal设置过高，会导致不必要的连接资源浪费。

可以根据实际总OPS和调用redis客户端的规模整体评估每个节点所使用的连接池。

3.监控

实际上最靠谱的值是通过监控来得到“最佳值”的，可以考虑通过一些手段(例如jmx)实现监控，找合理值。

四、常见问题

1.资源“不足"

```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the po
l
...
Caused by: java.util.NoSuchElementException: Timeout waiting for idle object
at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPool.java:
49)
```

或者

```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the po
l
...
Caused by: java.util.NoSuchElementException: Pool exhausted
at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPool.java:
64)
```

两种情况均属于无法从资源池获取到资源，但第一种是超时，第二种是因为blockWhenExhausted为f
lse根本就不等。

遇到此类异常，不要盲目的认为资源池不够大，第三节已经进行了分析。具体原因可以排查：网络、
源池参数设置、资源池监控(如果对jmx监控)、代码(例如没执行jedis.close())、慢查询、DNS等问题。

2. 预热JedisPool

由于一些原因(例如超时时间设置较小原因)，有的项目在启动成功后会出现超时。JedisPool定义最大
源数、最小空闲资源数时，不会真的把Jedis连接放到池子里，第一次使用时，池子没有资源使用，会
ew Jedis，使用后放到池子里，可能会有一定的时间开销，所以也可以考虑在JedisPool定义后，为Jed
sPool提前进行预热，例如以最小空闲数量为预热数量

```
List<Jedis> minIdleJedisList = new ArrayList<Jedis>(jedisPoolConfig.getMinIdle());
for (int i = 0; i < jedisPoolConfig.getMinIdle(); i++) {
   Jedis jedis = null;
   try {
      jedis = pool.getResource();
      minIdleJedisList.add(jedis);
      jedis.ping();
   } catch (Exception e) {
      logger.error(e.getMessage(), e);
   } finally {
      if (jedis != null)
         jedis.close();
   }
}

for (int i = 0; i < jedisPoolConfig.getMinIdle(); i++) {
   Jedis jedis = null;
   try {
      jedis = minIdleJedisList.get(i);
      jedis.close();
```

```
    } catch (Exception e) {
        logger.error(e.getMessage(), e);
    } finally {

        if (jedis != null)
            jedis.close();
    }
}
```

常见问题汇总

Jedis虽然使用起来比较简单，但是如果不能根据使用场景设置合理的参数(例如连接池参数)，不合理使用一些功能(例如Lua和事务)也会产生很多问题，本文对这些问题逐个说明：

详细目录：

ormal user.

一.无法从连接池获取到Jedis连接

1.异常堆栈

(1) 连接池参数blockWhenExhausted = true(默认)

如果连接池没有可用Jedis连接，会等待maxWaitMillis(毫秒)，依然没有获取到可用Jedis连接，会抛如下异常：

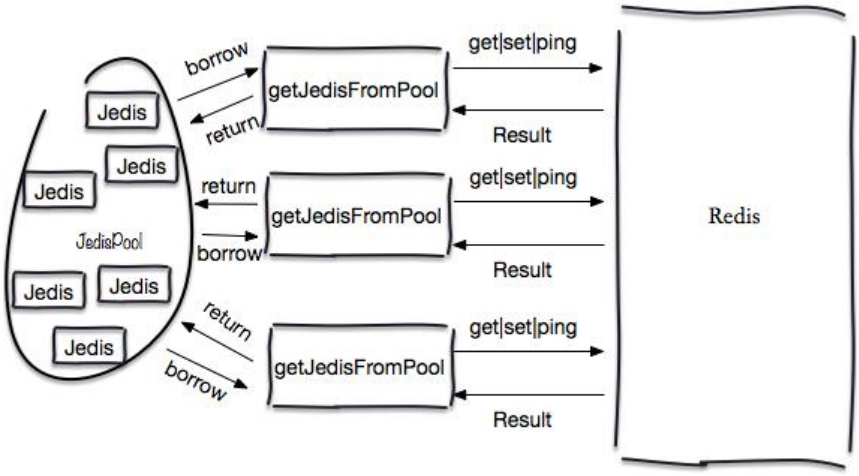
```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the po
l
    ...
Caused by: java.util.NoSuchElementException: Timeout waiting for idle object
    at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPool.ja
a:449)
```

(2) 连接池参数blockWhenExhausted = false

设置如果连接池没有可用Jedis连接，立即抛出异常：

```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the po
l
    ...
Caused by: java.util.NoSuchElementException: Pool exhausted
    at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPool.ja
a:464)
```

2.异常描述



1.从资源池借Jedis对象 2.Jedis执行命令 3.返回执行结果 4. 归还Jedis对象给连接池

上述异常是客户端没有从连接池(最大maxTotal个)拿到可用Jedis连接造成的，具体可能有如下原因：

(1) 连接泄露 (较为常见)

JedisPool默认的maxTotal=8，下面的代码从JedisPool中借了8次Jedis，但是没有归还，当第9次(jedsPool.getResource().ping())

```
GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
JedisPool jedisPool = new JedisPool(poolConfig, "127.0.0.1;

//向JedisPool借用8次连接，但是没有执行归还操作。
for (int i = 0; i < 8; i++) {
    Jedis jedis = null;
    try {
        jedis = jedisPool.getResource();
        jedis.ping();
    } catch (Exception e) {
         logger.error(e.getMessage(), e);
     }
 }
 jedisPool.getResource().ping();
```

所以推荐使用的代码规范是：

```
Jedis jedis = null;
try {
    jedis = jedisPool.getResource();
    //具体的命令
    jedis.executeCommand()
} catch (Exception e) {
    //如果命令有key最好把key也在错误日志打印出来，对于集群版来说通过key可以帮助定位到具体
点。
    logger.error(e.getMessage(), e);
 } finally {
     //注意这里不是关闭连接，在JedisPool模式下，Jedis会被归还给资源池。
     if (jedis != null)
         jedis.close();
 }
```

(2) 业务并发量大，maxTotal确实设置小了。

举个例子：

● 一次命令时间（borrow|return resource + Jedis执行命令(含网络) ）的平均耗时约为1ms，一个接的QPS大约是1000

● 业务期望的QPS是50000

那么理论上需要的资源池大小是50000 / 1000 = 50个，实际maxTotal可以根据理论值进行微调。

(3) Jedis连接还的太慢

例如Redis发生了阻塞(例如慢查询等原因)，所有连接在超时时间范围内等待，并发量较大时，会造成接池资源不足。

(4) 其他问题

例如丢包、DNS、客户端TCP参数配置。

3.解决方法：

可以看到这个问题稍微复杂一些，不要被异常的表象所迷惑，简单地认为连接池不够就盲目加大maxT

tal，要具体问题具体分析。

4.处理人

客户先确认，如解决不了，需要借助工单解决

还有一种情况是：从池子里拿连接，由于没有空闲连接，需要重新生成一个Jedis连接，但是连接被拒：

```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the po
l
    at redis.clients.util.Pool.getResource(Pool.java:50)
    at redis.clients.jedis.JedisPool.getResource(JedisPool.java:99)
    at TestAdmin.main(TestAdmin.java:14)
Caused by: redis.clients.jedis.exceptions.JedisConnectionException: java.net.ConnectException:
Connection refused
    at redis.clients.jedis.Connection.connect(Connection.java:164)
    at redis.clients.jedis.BinaryClient.connect(BinaryClient.java:80)
    at redis.clients.jedis.BinaryJedis.connect(BinaryJedis.java:1676)
    at redis.clients.jedis.JedisFactory.makeObject(JedisFactory.java:87)
     at org.apache.commons.pool2.impl.GenericObjectPool.create(GenericObjectPool.java:861)
     at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPool.ja
a:435)
     at org.apache.commons.pool2.impl.GenericObjectPool.borrowObject(GenericObjectPool.ja
a:363)
     at redis.clients.util.Pool.getResource(Pool.java:48)
     ... 2 more
 Caused by: java.net.ConnectException: Connection refused
     at java.net.PlainSocketImpl.socketConnect(Native Method)
     at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:339)
     at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:200)
     at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:182)
     at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
     at java.net.Socket.connect(Socket.java:579)
     at redis.clients.jedis.Connection.connect(Connection.java:158)
     ... 9 more
```

可以从at redis.clients.jedis.Connection.connect(Connection.java:158)看到实际是一个Socket连：

```
socket.setSoLinger(true, 0); // Control calls close () method,
      // the underlying socket is closed
      // immediately
      // <-@wjw_add
8:  socket.connect(new InetSocketAddress(host, port), connectionTimeout);
```

一般这种需要检查Redis的域名配置是否正确，排查该段时间网络是否正常

二、客户端缓冲区异常

1.异常堆栈

```
redis.clients.jedis.exceptions.JedisConnectionException: Unexpected end of stream.
```

```
    at redis.clients.util.RedisInputStream.ensureFill(RedisInputStream.java:199)
    at redis.clients.util.RedisInputStream.readByte(RedisInputStream.java:40)
    at redis.clients.jedis.Protocol.process(Protocol.java:151)
.....
```

2.异常描述：

这个异常是客户端缓冲区异常，产生这个问题可能有三个原因：

(1) 常见原因：多个线程使用一个Jedis连接，正常的情况是一个线程使用一个Jedis连接，可以使用JedisPool管理Jedis连接，实现线程安全，防止出现这种情况，例如下面代码中两个线程用了一个Jedis连：

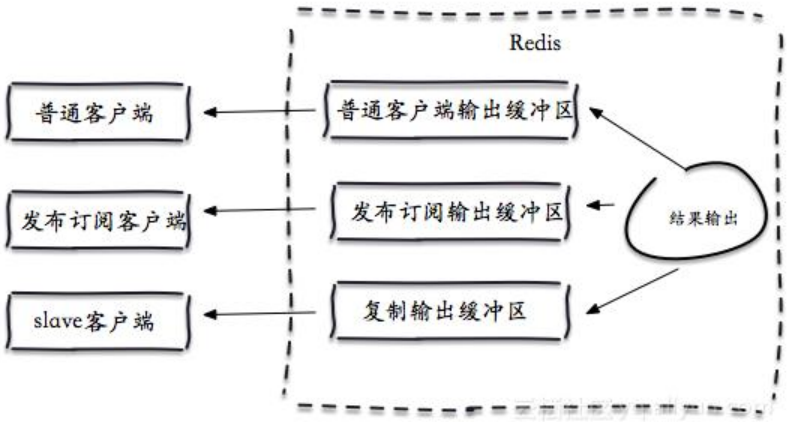
```
new Thread(new Runnable() {
    public  void  run() {
        for (int i = 0; i < 100; i++) {
            jedis.get("hello");
        }
    }
}).start();
 new Thread(new Runnable() {
     public  void  run() {
         for (int i = 0; i < 100; i++) {
             jedis.hget("haskey", "f");
         }
     }
 }).start();
```

(2) 客户端缓冲区满了

Redis有三种客户端缓冲区：

- 普通客户端缓冲区(normal)：用于接受普通的命令，例如get、set、mset、hgetall、zrange等
- slave客户端缓冲区(slave)：用于同步master节点的写命令，完成复制。
- 发布订阅缓冲区(pubsub)：pubsub不是普通的命令，因此有单独的缓冲区。

![客户端缓冲区]



Redis的客户端缓冲区配置具体格式是：

client-output-buffer-limit <class> <hard limit> <soft limit> <soft seconds>

(a) class: 客户端类型：(a) normal、(b) slave、(c) pubsub

(b) hard limit: 如果客户端使用的输出缓冲区大于hard limit，客户端会被立即关闭。

(c) soft limit和soft seconds: 如果客户端使用的输出缓冲区超过了soft limit并且持续了soft limit秒客户端会被立即关闭

例如下面是一份Redis缓冲区的配置，所以当条件满足时，客户端连接会被关闭，就会出现Unexpecte end of stream。

```
redis> config get client-output-buffer-limit
1) "client-output-buffer-limit"
2) "normal 524288000 0 0 slave 2147483648 536870912 480 pubsub 33554432 8388608 60"
```

(3) 长时间闲置连接被服务端主动断开，可以查询timeout配置的设置以及自身连接池配置是否需要做闲检测。

3.解决方法和处理人：

客户：排查自身代码是否使用JedisPool管理Jedis连接，是否存在并发操作Jedis的情况。

工单: 排查(2)(3)，阿里云Redis中timeout=0，也就是不会主动关闭空闲连接，缓冲区设置为0 0 0 也是不会对客户端缓冲区进行限制，一般不会有问题

三、非法客户端地址 （阿里云Redis提供客户端白名单功能）

1.异常堆栈

```
Caused by: redis.clients.jedis.exceptions.JedisDataException: ERR illegal address
    at redis.clients.jedis.Protocol.processError(Protocol.java:117)
    at redis.clients.jedis.Protocol.process(Protocol.java:151)
    at redis.clients.jedis.Protocol.read(Protocol.java:205)
    ......
```

2.异常描述：

Redis实例配置了白名单，但当前访问Redis的客户端(IP)不在白名单中。

3.解决方法：

添加该客户端(IP)的白名单

4.处理人

客户或者工单都可以

四、客户端连接数达到最大值

1.异常堆栈

redis.clients.jedis.exceptions.JedisDataException: ERR max number of clients reached

2.异常描述：

如果客户端连接数超过了Redis实例配置的最大maxclients

3.解决方法：

提工单帮助临时调大最大连接数，并让客户找到连接数暴涨的原因(因为上述调整只是临时调整)，

4.处理人

- 工单：临时调整最大连接数，协助定位问题
- 客户：定位自身问题(可以定位连接最多的客户端)，找到问题原因（例如连接池配置等）

五、客户端读写超时

1.异常堆栈

redis.clients.jedis.exceptions.JedisConnectionException: java.net.SocketTimeoutException: Rea
timed out

2.异常描述：

该问题原因可能有如下几种：
(1) 读写超时设置的过短。
(2) 有慢查询或者Redis发生阻塞。
(3) 网络不稳定。

3.解决方法：

客户提供读写超时时间，提交工单定位相关原因

4.处理人：

工单。

六、密码相关的异常

1.异常堆栈

Redis设置了密码，客户端请求没传密码：

```
Exception in thread "main" redis.clients.jedis.exceptions.JedisDataException: NOAUTH Authentication required.
    at redis.clients.jedis.Protocol.processError(Protocol.java:127)
    at redis.clients.jedis.Protocol.process(Protocol.java:161)
    at redis.clients.jedis.Protocol.read(Protocol.java:215)
```

Redis没有设置密码，客户端传了密码：

```
Exception in thread "main" redis.clients.jedis.exceptions.JedisDataException: ERR Client sent AUTH, but no password is set
    at redis.clients.jedis.Protocol.processError(Protocol.java:127)
    at redis.clients.jedis.Protocol.process(Protocol.java:161)
    at redis.clients.jedis.Protocol.read(Protocol.java:215)
```

客户端传了错误的密码：

```
redis.clients.jedis.exceptions.JedisDataException: ERR invalid password
    at redis.clients.jedis.Protocol.processError(Protocol.java:117)
    at redis.clients.jedis.Protocol.process(Protocol.java:151)
    at redis.clients.jedis.Protocol.read(Protocol.java:205)
```

2.解决方法：弄清楚到底有没有密码，密码是否正确。

七、事务异常

1.异常堆栈

```
redis.clients.jedis.exceptions.JedisDataException: EXECABORT Transaction discarded because of previous errors
```

2.异常描述：

这个是Redis的事务异常：事务中包含了错误的命令，例如如下sett是个不存在的命令。

```
127.0.0.1:6379> multi
OK
127.0.0.1:6379> sett key world
(error) ERR unknown command 'sett'
127.0.0.1:6379> incr counter
QUEUED
127.0.0.1:6379> exec
(error) EXECABORT Transaction discarded because of previous errors.
```

3.解决方法和处理人：

客户修复自身代码错误。

八、类转换错误

1.异常堆栈

```
java.lang.ClassCastException: java.lang.Long cannot be cast to java.util.List
        at redis.clients.jedis.Connection.getBinaryMultiBulkReply(Connection.java:199)
        at redis.clients.jedis.Jedis.hgetAll(Jedis.java:851)
        at redis.clients.jedis.ShardedJedis.hgetAll(ShardedJedis.java:198)

java.lang.ClassCastException: java.util.ArrayList cannot be cast to [B
        at redis.clients.jedis.Connection.getBinaryBulkReply(Connection.java:182)
        at redis.clients.jedis.Connection.getBulkReply(Connection.java:171)
        at redis.clients.jedis.Jedis.rpop(Jedis.java:1109)
        at redis.clients.jedis.ShardedJedis.rpop(ShardedJedis.java:258)
.......
```

2.异常描述：

Jedis正确的使用方法是：一个线程操作一个Jedis，通常来讲产生该错误是由于没有使用JedisPool造的，例如如下代码在两个线程并发使用了一个Jedis。(get、hgetAll返回类型也是不一样的)

```
new Thread(new Runnable() {
   public  void  run() {
      for (int i = 0; i < 100; i++) {
         jedis.set("hello", "world");
         jedis.get("hello");
      }
   }
}).start();
new Thread(new Runnable() {
    public  void  run() {
       for (int i = 0; i < 100; i++) {
          jedis.hset("hashkey", "f", "v");
          jedis.hgetAll("hashkey");
       }
    }
}).start();
```

3.解决方法和处理人：

客户排查自身代码是否存在上述问题

九、命令使用错误

1.异常堆栈

```
Exception in thread "main" redis.clients.jedis.exceptions.JedisDataException: WRONGTYPE Op
ration against a key holding the wrong kind of value
    at redis.clients.jedis.Protocol.processError(Protocol.java:127)
    at redis.clients.jedis.Protocol.process(Protocol.java:161)
    at redis.clients.jedis.Protocol.read(Protocol.java:215)
.....
```

2.异常描述：

例如key="hello"是字符串类型的键，而hgetAll是哈希类型的键，所以出现了错误。

```
jedis.set("hello","world");
jedis.hgetAll("hello");
```

3.解决方法和处理人：

请客户修改自身代码错误。

十、Redis使用的内存超过maxmemory配置

1.异常堆栈

redis.clients.jedis.exceptions.JedisDataException: OOM command not allowed when used memory > 'maxmemory'.

2.异常描述：

Redis节点(如果是集群，则是其中一个节点)使用大于该实例的内存规格(maxmemory配置)。

3.解决方法：

原因可能有以下几个：

- 业务数据正常增加
- 客户端缓冲区异常：例如使用了monitor、pub/sub使用不当等等
- 纯缓存使用场景，但是maxmemory-policy配置有误(例如没有过期键的业务配置volatile-lru)

紧急处理，可以临时提工单帮助临时调整maxmeory，后续咨询用户是否升配或者调整配置。

4.处理人

- 客户：找到内存增大的原因。
- 工单：协助临时调整maxmeomry，如果客户需要，可以协助解决

十一、Redis正在加载持久化文件

1.异常堆栈

redis.clients.jedis.exceptions.JedisDataException: LOADING Redis is loading the dataset in memory

2.异常描述：

Jedis调用Redis时，如果Redis正在加载持久化文件，无法进行正常的读写。

3.解决方法：

正常情况下，阿里云Redis不会出现这种情况，如果出现，则提交工单处理。

4.处理人：

工单。

十二、Lua脚本超时

1.异常堆栈

redis.clients.jedis.exceptions.JedisDataException: BUSY Redis is busy running a script. You can only call SCRIPT KILL or SHUTDOWN NOSAVE.

2.异常描述：

如果Redis当前正在执行Lua脚本，并且超过了lua-time-limit，此时Jedis调用Redis时，会收到下面异常

3.解决方法：

按照异常提示：You can only call SCRIPT KILL or SHUTDOWN NOSAVE. (使用script kill:kill掉Lu脚本)

4.处理人：

最好客户自己处理，如果解决不了，值班人员可以协助操作。

十三 连接超时

1.异常堆栈

redis.clients.jedis.exceptions.JedisConnectionException: java.net.SocketTimeoutException: connect timed out

2.异常描述：

可能产生的原因：

- 连接超时设置的过短。
- tcp-backlog满，造成新的连接失败。
- 客户端与服务端网络不正常。

3.解决方法：

客户提供连接超时时间，提交工单定位相关原因。

4.处理人：

工单。

十四 Lua脚本写超时

1.异常堆栈

```
(error) UNKILLABLE Sorry the script already executed write commands against the dataset. Yo
 can either wait the script termination or kill the server in a hard way using the SHUTDOWN
OSAVE command.
```

2.异常描述：

如果Redis当前正在执行Lua脚本，并且超过了lua-time-limit，并且已经执行过写命令，此时Jedis调Redis时，会收到上面的异常

3.解决方法：

提交工单做紧急处理，管理员要做重启或者切换Redis节点。

4.处理人：

工单。

十五、类加载错误

1.异常堆栈

例如找不到类和方法：

```
Exception in thread "commons-pool-EvictionTimer" java.lang.NoClassDefFoundError: redis/cli
nts/util/IOUtils
    at redis.clients.jedis.Connection.disconnect(Connection.java:226)
    at redis.clients.jedis.BinaryClient.disconnect(BinaryClient.java:941)
    at redis.clients.jedis.BinaryJedis.disconnect(BinaryJedis.java:1771)
    at redis.clients.jedis.JedisFactory.destroyObject(JedisFactory.java:91)
    at          org.apache.commons.pool2.impl.GenericObjectPool.destroy(GenericObjectPool.java
897)
    at org.apache.commons.pool2.impl.GenericObjectPool.evict(GenericObjectPool.java:793)
    at org.apache.commons.pool2.impl.BaseGenericObjectPool$Evictor.run(BaseGenericObject
ool.java:1036)
    at java.util.TimerThread.mainLoop(Timer.java:555)
     at java.util.TimerThread.run(Timer.java:505)
 Caused by: java.lang.ClassNotFoundException: redis.clients.util.IOUtils
 ......
```

2.异常描述：

运行时，Jedis执行命令，抛出异常：某个类找不到。一般此类问题都是由于加载多个jedis版本(例如jedis 2.9.0和jedis 2.6)，在编译期代码未出现问题，但类加载器在运行时加载了低版本的Jedis，造成运时找不到类。

3.解决方法：

通常此类问题，可以将重复的jedis排除掉，例如利用maven的依赖树，把无用的依赖去掉或者exclusin掉。

4.处理人

客户排查自身代码

十六、服务端命令不支持

1.异常堆栈

例如客户端执行了geoadd命令，但是服务端返回不支持此命令

redis.clients.jedis.exceptions.JedisDataException: ERR unknown command 'GEOADD'

2.异常描述：

该命令不能被Redis端识别，有可能有两个原因：

- 社区版的一些命令，阿里云Redis的不支持，或者只在某些小版本上支持(例如geoadd是Redis 3.2加的地理信息api)。
- 命令本身是错误的(不过对于Jedis来说还好，不支持直接组装命令，每个API都有固定的函数)。

3.解决方法：

咨询是否有Redis版本支持该命令，如支持可以让客户做小版本升级。

4.处理人

- 管理员：确认版本是否支持该命令
- 客户：确认后，做小版本升级

十七、pipeline错误使用

1.异常堆栈

redis.clients.jedis.exceptions.JedisDataException: Please close pipeline or multi block before calling this method.

2.异常描述：

在pipeline.sync()执行之前，通过response.get()获取值，在pipeline.sync()执行前，命令没有执行(以通过monitor做验证)，下面代码就会引起上述异常

```
Jedis jedis = new Jedis("127.0.0.1", 6379);
Pipeline pipeline = jedis.pipelined();
pipeline.set("hello", "world");
pipeline.set("java", "jedis");
Response<String> pipeString = pipeline.get("java");
//这个get必须在sync之后，如果是批量获取值建议直接用List<Object> objectList = pipeline.sync
ndReturnAll();
System.out.println(pipeString.get());
//命令此时真正执行
 pipeline.sync();
```

Jedis中Reponse中get()方法，有个判断:如果set=false就会报错，而response中的set初始化为false.

```
public T get() {
  //  if response has dependency response and dependency is  not built,
  // build it first and  no more!!
  if (dependency != null && dependency.set && !dependency.built) {
    dependency.build();
  }
  if (!set) {
    throw new JedisDataException(
        "Please close pipeline or multi block before calling this method.");
  }
  if (!built) {
    build();
  }
  if (exception != null) {
    throw exception;
  }
  return response;
 }
```

pipeline.sync()会每个结果设置set=true。

```
public  void  sync() {
  if (getPipelinedResponseLength() > 0) {
    List<Object> unformatted = client.getAll();
    for (Object o : unformatted) {
      generateResponse(o);
    }
  }
}
```

其中generateResponse(o):

```
protected Response<?> generateResponse(Object data) {
  Response<?> response = pipelinedResponses.poll();
  if (response != null) {
    response.set(data);
```

```
  }
  return response;
}
```

其中response.set(data);

```
public  void  set(Object data) {
    this.data = data;
    set = true;
}
```

3.解决方法：

实际上对于批量结果的解析，建议使用pipeline.syncAndReturnAll()来实现，下面操作模拟了批量hgtAll

```
/**
* pipeline模拟批量hgetAll
* @param keyList
* @return
*/
public Map<String, Map<String, String>> mHgetAll(List<String> keyList) {
// 1.生成pipeline对象
Pipeline pipeline = jedis.pipelined();
 // 2.pipeline执行命令，注意此时命令并未真正执行
 for (String key : keyList) {
   pipeline.hgetAll(key);
 }

 // 3.执行命令 syncAndReturnAll()返回结果
 List<Object> objectList = pipeline.syncAndReturnAll();
 if (objectList == null || objectList.isEmpty()) {
   return Collections.emptyMap();
 }

// 4.解析结果
Map<String,Map<String, String>> resultMap = new HashMap<String, Map<String,String>>()

for (int i = 0; i < objectList.size(); i++) {
  Object object = objectList.get(i);
  Map<String, String> map = (Map<String, String>) object;
  String key = keyList.get(i);
  resultMap.put(key, map);
}
return resultMap;
}
```

4.处理人：

修改业务代码。

十八、管理员命令，普通用户不能执行

1.异常堆栈

命令role不能被普通用户执行，可以参考暂未开放的Redis命令

redis.clients.jedis.exceptions.JedisDataException: ERR command role not support for normal u
er

2.异常描述：

改命令尚未开放

3.解决方法：

不能使用该命令，如果有需求或者疑问可以联系值班人员。

4.处理人

从文档中确认该命令是否开放