2023.8.25电子秤无法传价格故障

故障报障时间：2023-08-25 08:11

故障实际发生时间：2023-08-25 07:56

故障开始处理时间：2023-08-25 08:13

故障恢复时间： 2023-08-25 08:18

故障影响范围：6家店商品价格/资料下秤失败报障,经排查下秤服务2个分片(总共40个分片，两个分片上共47家店任务等待执行)任务阻塞.

故障问题描述：门店商品价格/资料传秤失败

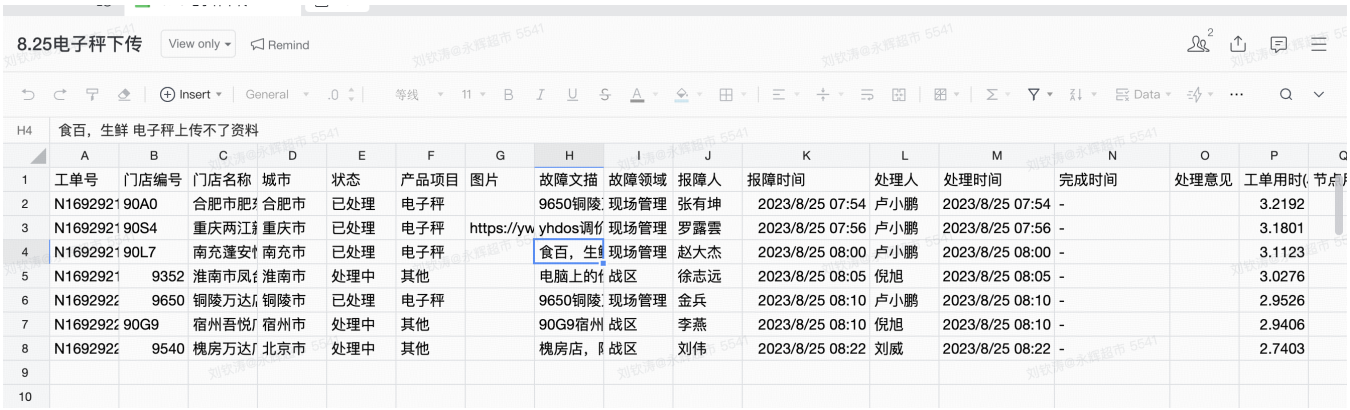
故障根因：redis连接卡死hang住，导致job卡死任务阻塞，从而导致商品价格/资料下秤失败。

故障处理与分析过程：

一、故障处理过程

1、08:11 —运维灭火群报账

后续跟运维要到的实际报障门店,共有6家门店报障(90S4,90L7,9352,9650,90G9,9540)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	工单号	门店编号	门店名称	城市	状态	产品项目	图片	故障文描	故障领域	报障人	报障时间	处理人	处理时间	完成时间	处理意见	工单用时(	节点
2	N1692921	90A0	合肥市肥	合肥市	已处理	电子秤		9650铜陵	现场管理	张有坤	2023/8/25 07:54	卢小鹏	2023/8/25 07:54	-		3.2192	
3	N1692921	90S4	重庆两江	重庆市	已处理	电子秤	https://yw	yhdos调仓	现场管理	罗露雲	2023/8/25 07:56	卢小鹏	2023/8/25 07:56	-		3.1801	
4	N1692921	90L7	南充蓬安	南充市	已处理	电子秤		食百，生	现场管理	赵大杰	2023/8/25 08:00	卢小鹏	2023/8/25 08:00	-		3.1123	
5	N1692921	9352	淮南市凤	淮南市	处理中	其他		电脑上的	战区	徐志远	2023/8/25 08:05	倪旭	2023/8/25 08:05	-		3.0276	
6	N1692922	9650	铜陵万达	铜陵市	已处理	电子秤		9650铜陵	现场管理	金兵	2023/8/25 08:10	卢小鹏	2023/8/25 08:10	-		2.9526	
7	N1692922	90G9	宿州吾悦	宿州市	处理中	其他		90G9宿州	战区	李燕	2023/8/25 08:10	倪旭	2023/8/25 08:10	-		2.9406	
8	N1692922	9540	槐房万达	北京市	处理中	其他		槐房店，	战区	刘伟	2023/8/25 08:22	刘威	2023/8/25 08:22	-		2.7403	
9																	
10																	

2、08:13 一接到通知进入灭火群进行处理

3、一按照之前的排障预案进行处理，排查ES JOB，发现两个分片长时间运行中，判断为该分片任务阻塞。

4、08:18重启出问题的两台机器

变更内容: 重启容器 yh-sod-scale-deviceadapter-e1576-694f485869-4tb4q
项目空间: 产品-计量中台
变更环境: 生产环境
变更应用: yh-sod-scale-deviceadapter
变更说明: 【负载过高】 重启
变更时间: 2023-08-25 08:18:17
操作人: 刘钦涛#81116320

发布通知 BOT

变更内容: 重启容器 yh-sod-scale-deviceadapter-e1576-694f485869-bwzc6
项目空间: 产品-计量中台
变更环境: 生产环境
变更应用: yh-sod-scale-deviceadapter
变更说明: 【负载过高】 重启
变更时间: 2023-08-25 08:18:34
操作人: 刘钦涛#81116320

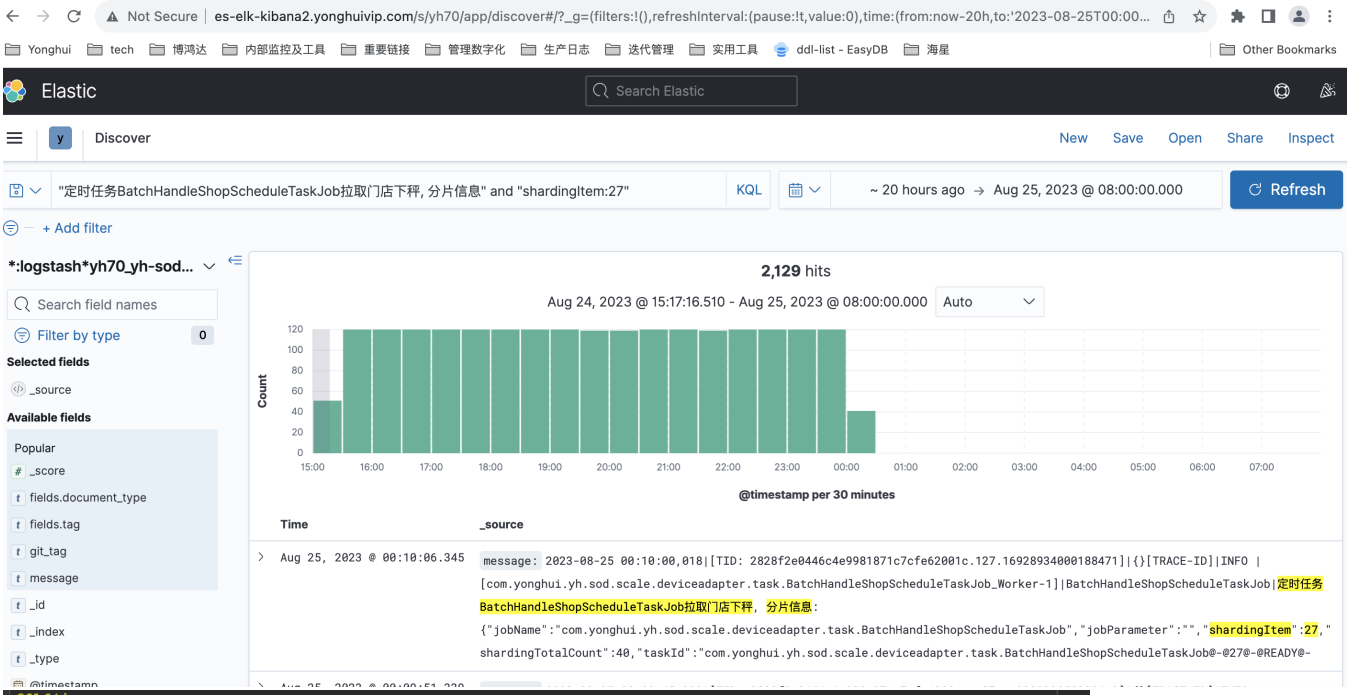
重启后同步灭火群，运维联系门店进行确认，研发看数据库任务已经正常处理

根因分析过程:

问题恢复后，研发进行问题排查，根据门店找到对应的分片之后，

1、发现最后一次调度任务是在00:10 最后一次调度卡住

http://es-elk-kibana2.yonghuivip.com/s/yh70/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:now-20h,to:'2023-08-25T00:00:00.000Z'))&_a=(columns:!(_source),filters:!(),index:'7af6f550-0770-11ed-bd60-4d58239e1af1',interval:auto,query:(language:kuery,query:'%22%E5%AE%9A%E6%97%B6%E4%BB%BB%E5%8A%A1BatchHandleShopScheduleTaskJob%E6%8B%89%E5%8F%96%E9%97%A8%E5%BA%97%E4%B8%8B%E7%A7%A4,%20%E5%88%86%E7%89%87%E4%BF%A1%E6%81%AF%22%20and%20%22shardingItem:27%22'),sort:!('@timestamp',desc))

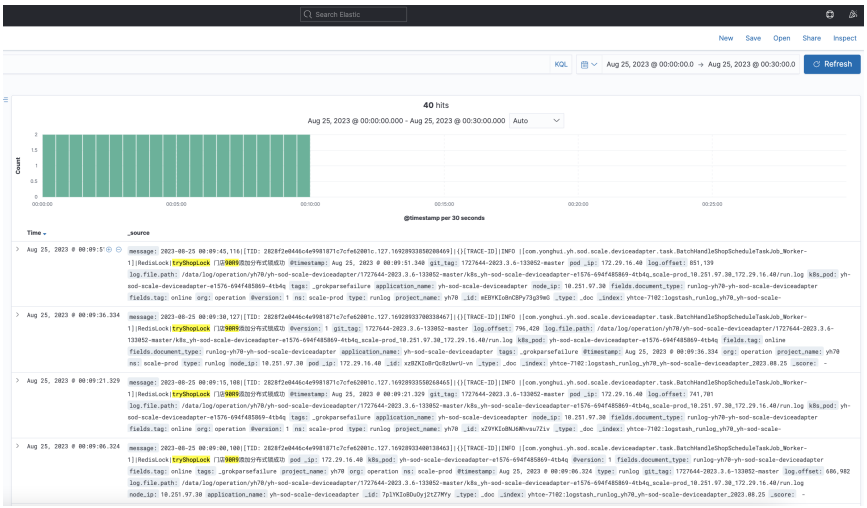


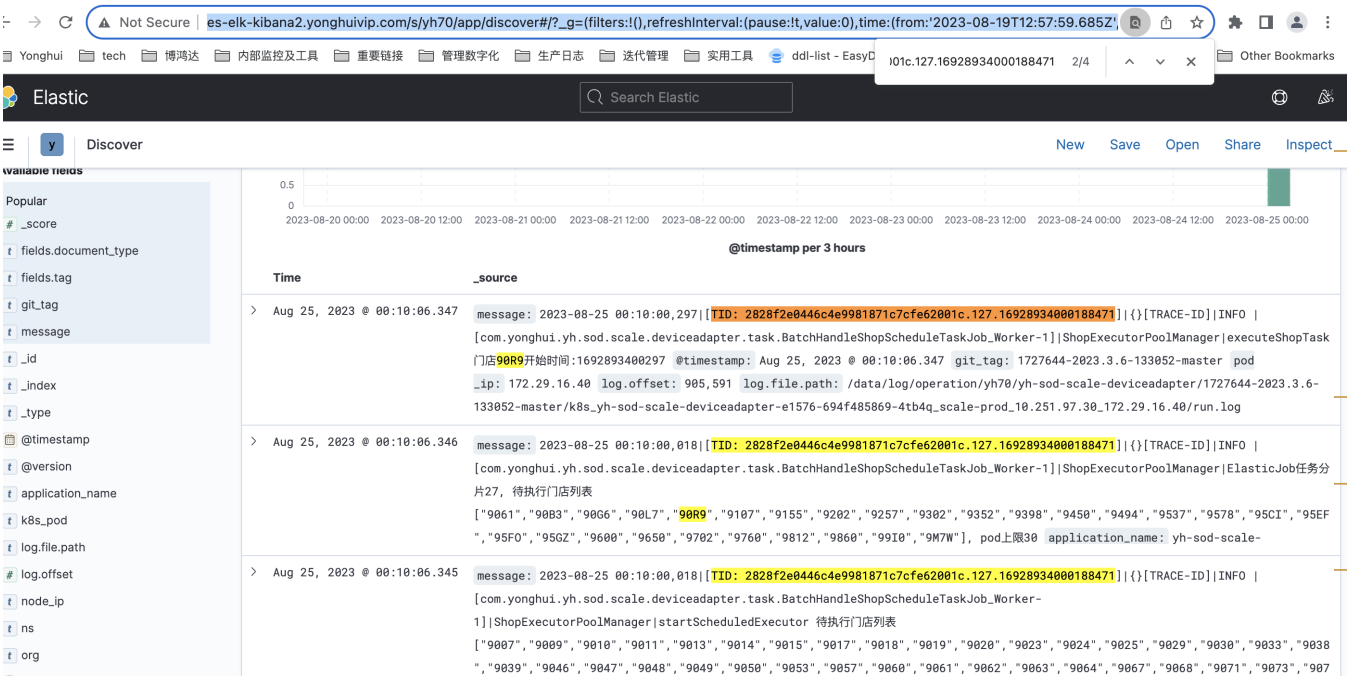

```java
@Slf4j
@Component
public class BatchHandleShopScheduleTaskJob implements SimpleJob {

    @Autowired
    private ShopExecutorPoolManager shopExecutorPoolManager;

    @Override
    public void execute(ShardingContext shardingContext) {
        try{
            log.info("定时任务BatchHandleShopScheduleTaskJob拉取门店下秤, 分片信息:{}", JSON.toJSONString(shardingContext));
            long startScheduledTime = System.currentTimeMillis();
            shopExecutorPoolManager.startScheduledExecutor(shardingContext.getShardingItem(), shardingContext.getShardingTot
            long endScheduledTime = System.currentTimeMillis();
            log.info("定时任务BatchHandleShopScheduleTaskJob任务分片{}, 调度开始时间:{}, 调度结束时间:{}, 耗时:{}", shardingContext
        } catch(Exception e){
            log.error("定时器BatchHandleShopScheduleTaskJob任务分片{}处理异常:{}", shardingContext.getShardingItem(), JSON.toJS
        }
    }
}
```

2、进一步发现90R9门店的处理流程卡住

可发现，没有获取锁成功的日志，只有开始执行的日志

http://es-elk-kibana2.yonghuivip.com/s/yh70/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:'2023-08-19T12:57:59.685Z',to:'2023-08-24T16:20:50.531Z'))&_a=(columns:!(_source),filters:!(),index:'7af6f550-0770-11ed-bd60-4d58239e1af1',interval:auto,query:(language:kuery,query:'%22TID:%202828f2e0446c4e9981871c7cfe62001c.127.16928934000188471%22%20and%20%2290R9%22'),sort:!('@timestamp',desc))





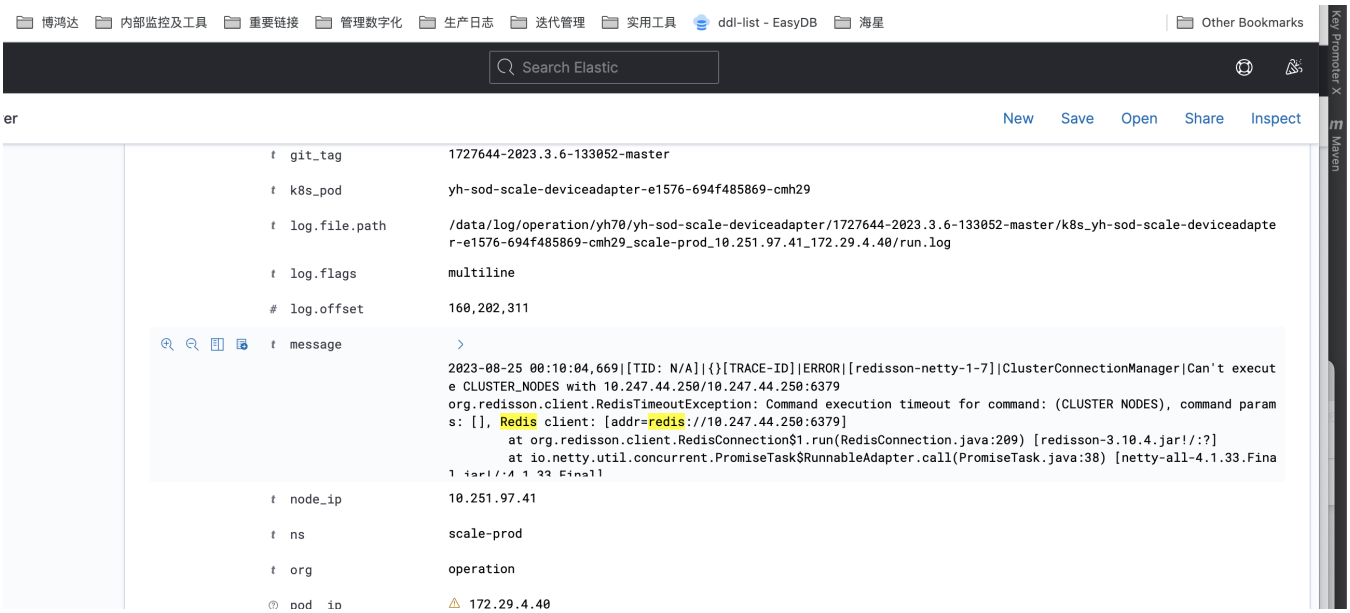
3、结合日志以及代码，可以看到与redis交互之后，至少会有一条与锁相关的日志，但是上面的日志中没有，判断为获取redis锁的地方卡住了

```
public void executeShopTask(List<String> shopIdList, Integer shopTaskLimit) {
    long startScheduledTime = System.currentTimeMillis();
    log.info("executeShopTask 批量执行门店开始时间:{}", startScheduledTime);
    // 查找当前门店下的秤任务
    for (String shopId : shopIdList) {
        long startTime = System.currentTimeMillis();
        log.info("executeShopTask 门店{}开始时间:{}", shopId, startTime);
        try {
            // 判断门店是否达到上限
            if (!validateShopNumsLimit(shopId, shopTaskLimit)) {
                continue;
            }
            // 门店分布式锁,加锁失败继续下个门店
            if (!redisLock.tryShopLock(shopId)) {
                continue;
            }
            // 查询门店任务
            List<String> scaleIpList = findScaleIpByParam(shopId);
            log.info("executeShopTask 门店{}可执行秤列表{}", shopId, JSONObject.toJSONString(scaleIpList));
            if (CollectionUtils.isEmpty(scaleIpList)) {
                log.info("executeShopTask 门店{}无可执行任务", shopId);
                // 释放门店锁
                redisLock.releaseShopLock(shopId);
                continue;
            }
```

4、redisson异常日志(发生在非阻塞的节点上)，发现客户端与redis连接异常，发现当时redis连接确实存在异常

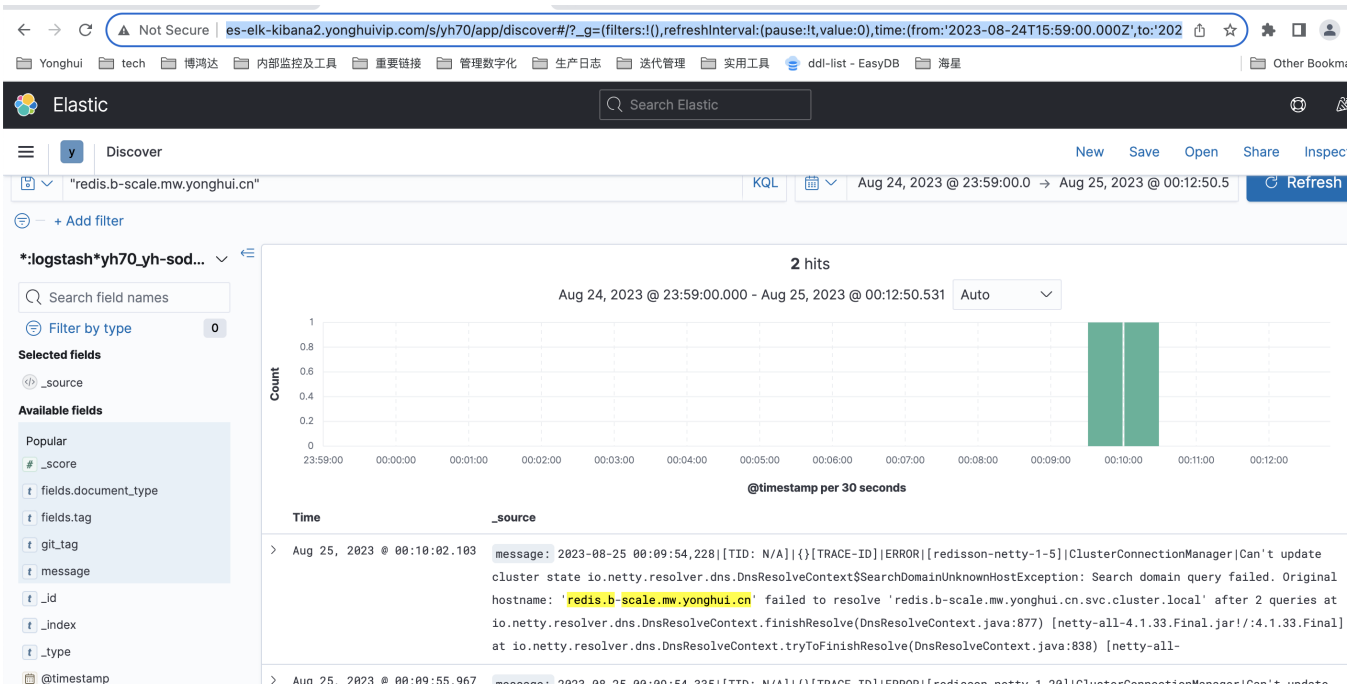
Can't update cluster state 异常

http://es-elk-kibana2.yonghuivip.com/s/yh70/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:'2023-08-19T12:57:59.685Z',to:'2023-08-24T16:20:50.531Z'))&_a=(columns:!(_source),filters:!(),index:'7af6f550-0770-11ed-bd60-4d58239e1af1',interval:auto,query:(language:kuery,query:%22redis%22),sort:!('@timestamp',desc))



域名异常

http://es-elk-kibana2.yonghuivip.com/s/yh70/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:'2023-08-24T15:59:00.000Z',to:'2023-08-24T16:12:50.531Z'))&_a=(columns:!(_source),filters:!(),index:'7af6f550-0770-11ed-bd60-4d58239e1af1',interval:auto,query:(language:kuery,query:%22redis.b-scale.mw.yonghui.cn%22),sort:!('@timestamp',desc))



5、分析redis交互的代码

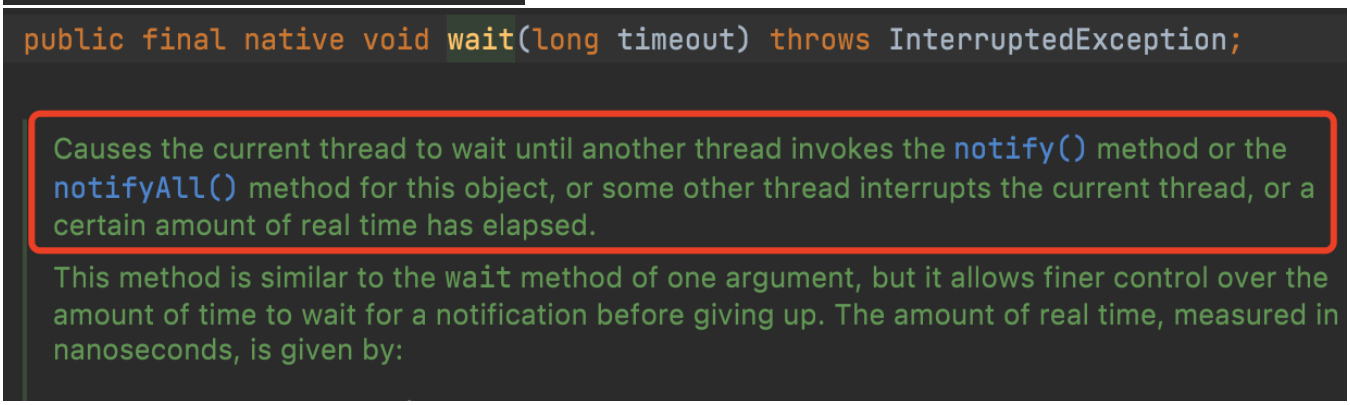
```
597
598     public <T, R> RFuture<R> writeAsync(byte[] key, Codec codec, RedisCommand<T> command, Object... params) {
599         RPromise<R> mainPromise = this.createPromise();
600         NodeSource source = this.getNodeSource(key);
601         this.async( readOnlyMode: false, source, codec, command, params, mainPromise,  attempt: 0,  ignoreRedirect: false);
602         return mainPromise;
603     }
```

这里返回mainPromise，它的await方法最终会被调用。

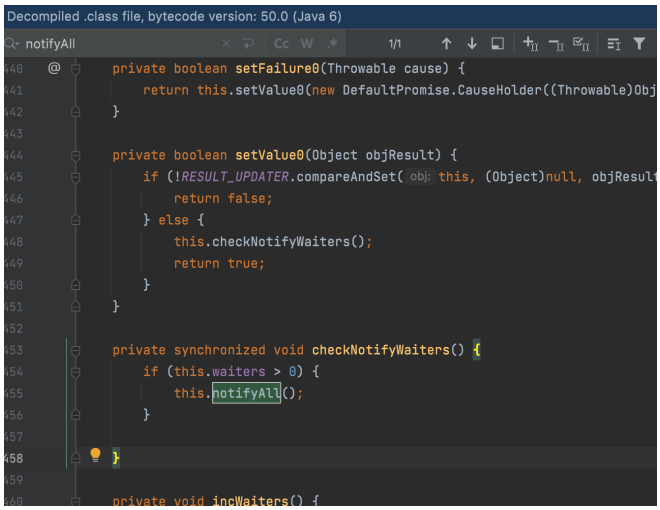
async中的代码：

DefaultPromise的await调用object.wait

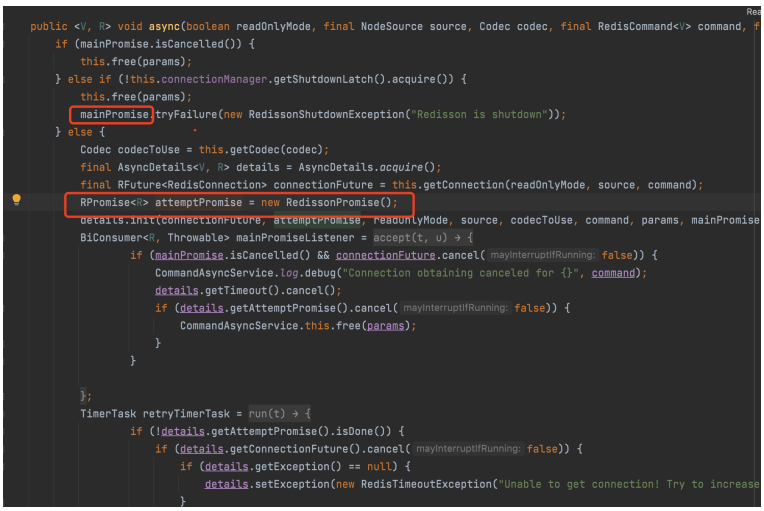
```
public final void wait() throws InterruptedException {
wait(0);
}
```



有wait()必定有notify()或者notifyAll()，而且因为wait()的对象是this，所以DefaultPromise里面必定有相应的方法来唤醒等待的线程。



当waiters》0时，会走notifyAll()，但是，



wait()和notifyAll()分别属于两个不同的对象，导致waiters=0，走不到notifyAll()，从而线程挂起。

与redis相关的类库：

yh-starter-redis-0.0.5-RELEASE.jar
redisson-spring-boot-starter-3.10.4.jar
spring-boot-starter-data-redis-2.1.3.RELEASE.jar
spring-data-redis-2.1.5.RELEASE.jar
redisson-3.10.4.jar
redisson-spring-data-21-3.10.4.jar

改进措施action：

1、增加POD维度的ESJob未触发消息及电话告警逻辑 - @王文斌 0825 已经添加

2、ES JOB 异步运行，保证JOB不阻塞-@王文斌，2023.09.05迭代上线

3、增加下秤门店数量/秤数量/任务数量拥堵、失败/成功率、线程数、最大/Tp99耗时 监控、消息和电话告警-@刘钦涛， 2023.09.05迭代上线

4、下秤上下游链路梳理及技改专项、redis sdk 从redisson改用jedis-@王文斌 2023.9.12

故障责任人：刘钦涛

故障等级：L5