

Hw_4 基于Kafka对项目的再优化-Fi&Shipping 物流管理系统系统设计报告

🕒 报告完成时间：2023-6-21 23:31

撰写人：-20231266 禹浩男

小组成员：-20271006 付柏逢 -20231107 潘明豪

仓库链接：<https://github.com/laven00/trans>

用户测试账号：pmh 密码：123

管理员测试账号：yhn 密码：123

#0 原理

Kafka 是一种高吞吐量、分布式、可扩展、可靠性的消息队列系统，它的核心原理主要包括以下几个方面：

第一是消息存储，Kafka 使用一个分布式的、可扩展的、持久化的消息存储系统来保存消息。每个 Topic 的消息被分为多个 Partition，每个 Partition 都被复制到多个 Broker 上，以保证可靠性。Kafka 的消息存储采用顺序写入的方式，保证了高吞吐量和低延迟。

其次是生产者和消费者业务角色，Kafka 的生产者将消息发送到指定 Topic 的 Partition 中，生产者可以选择同步或异步的方式发送消息。生产者可以配置消息的压缩、序列化方式和分区策略等。消费者则从指定的 Topic 和 Partition 中读取消息。每个消费者组可以有多个消费者实例，每个消费者实例只能读取一个 Partition 的消息。消费者可以配置消息的反序列化方式和消费的 Offset 策略等。

第三是offset，Kafka 使用 Offset 来标识消费者读取的消息位置。消费者可以通过手动提交 Offset 或者自动提交 Offset 来控制消息的消费位置。

最后是ZooKeeper，Kafka 使用 ZooKeeper 来进行 Broker 的选举、Topic 和 Partition 的元数据管理等。ZooKeeper 作为 Kafka 的协调器，保证了 Kafka 的可靠性和高可用性。

总之，Kafka 是一种高吞吐量、分布式、可扩展、可靠性的消息队列系统，它的核心原理包括消息存储、生产者、消费者、Offset 和 ZooKeeper 等。Kafka 的优点是高吞吐量、低延迟、可靠性和可扩展性等，被广泛应用于大数据、实时计算、日志处理等领域。

#1 应用

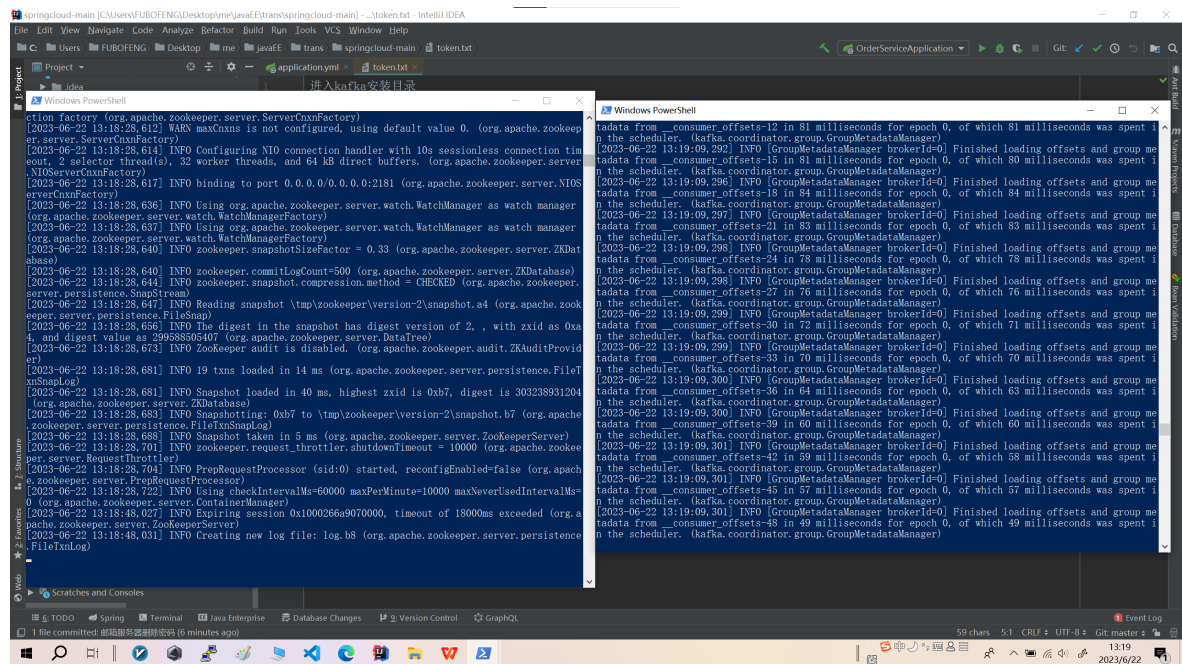
本项目主要应用了的Kafka 消息队列的功能来进行异步消息的传递和处理。

在第三次的项目中，我们通过服务调用来处理物流员分配运输资源的业务逻辑，即在调用orderservice服务的接口中再次调用transunit-service服务的接口去更新运输组的运输资源，我们发现这种方式在系统的并发请求过多时容易造成数据的不一致性，严重影响了业务逻辑的后续跟进和发展

所以在本次的项目重构中我们采用了kafka的消息队列来完成对运输资源的更新，在有对运输资源进行更新的请求时，我们将参数transunitid（运力资源id）和rest（运输资源量）作为消息传递给transunit-service服务集群，通过kafka缓冲队列来实现对运力资源的更新，这种方式不仅可以减小系统并发压力，保证了数据的一致性和安全性，并且我们将多个transunit-service分为一个组，在orderservice生产者端新建了多个partition分区，实现了对transunit-service服务的轮询消费，进一步减小了transunit-service的压力。

#2 实现

本地运行zookeeper及kafka



```
2023-06-22 13:18:28.612] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFactory)
[2023-06-22 13:18:28.614] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 2 selector thread(s), 32 worker threads, and 64 KB direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2023-06-22 13:18:28.617] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2023-06-22 13:18:28.636] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
[2023-06-22 13:18:28.637] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
[2023-06-22 13:18:28.640] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2023-06-22 13:18:28.644] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
[2023-06-22 13:18:28.647] INFO Reading snapshot \tmp\zookeeper\version-2\snapshot.a4 (org.apache.zookeeper.server.persistence.FileSnap)
[2023-06-22 13:18:28.656] INFO The digest in the snapshot has digest version of 2, with zxid as 0xa4 and digest value as 39658505407 (org.apache.zookeeper.server.DataTree)
[2023-06-22 13:18:28.676] INFO Zookeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2023-06-22 13:18:28.681] INFO 19 txns loaded in 14 ms. (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-06-22 13:18:28.681] INFO Snapshot loaded in 40 ms, highest zxid is 0xb7, digest is 303238931204 (org.apache.zookeeper.server.ZKDatabase)
[2023-06-22 13:18:28.683] INFO Snapshotting: 0xb7 to \tmp\zookeeper\version-2\snapshot.b7 (org.apache.zookeeper.server.persistence.FileSnap)
[2023-06-22 13:18:28.688] INFO Snapshot taken in 5 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2023-06-22 13:18:28.701] INFO zookeeper.request.throttler.shutdown.timeout = 10000 (org.apache.zookeeper.server.RequestThrottler)
[2023-06-22 13:18:28.704] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepareRequestProcessor)
[2023-06-22 13:18:28.722] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2023-06-22 13:18:48.023] INFO Expiring session 0x1000266a9070000, timeout of 18000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
[2023-06-22 13:18:48.031] INFO Creating new log file: log.b8 (org.apache.zookeeper.server.persistence.FileTxnLog)
```

关键配置：

生产者

```

spring:
  application:
    name: orderservice

  cloud:
    stream:
      kafka:
        binder:
          brokers: localhost:9092      #Kafka 的消息中间件服务器
          zk-nodes: localhost:2181     #Zookeeper 的节点, 如果集群, 后面加, 号分隔
          auto-create-topics: true     #如果设置为 false, 就不会自动创建 Topic 有可能你 Topic
还没创建就直接调用了。
          auto-add-partitions: true

        bindings:
          stream-demo:                #这里可以任意写, 消费者应与之一致
            destination: custom-message-topic  #这里可以任意写, 消费者应与之一致, 消息发往的目的
的地

```

```

      content-type: application/json  #消息发送的格式, 接收端不用指定格式, 但是发送端要。
文本则为 text/plain

      producer:
        # 分区数量 (默认为 1)
        partition-count: 3

```

消费者

```
kafka:
  group: lyq

spring:
  application:
    name: transunitservice

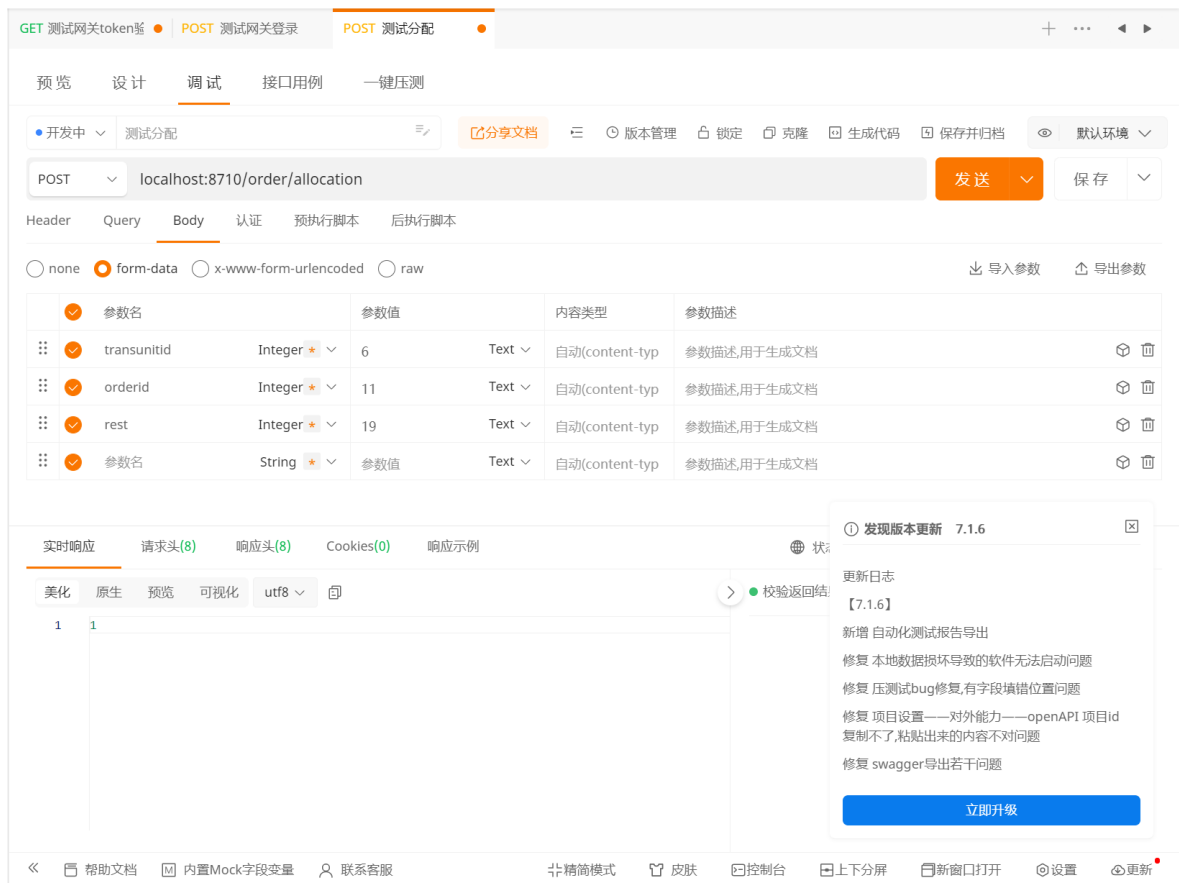
cloud:
  stream:
    kafka:
      binder:
        brokers: localhost:9092
        zk-nodes: localhost:2181
        auto-create-topics: true
      bindings:
        stream-demo:
          destination: custom-message-topic
          content-type: application/json
          group: ${kafka.group}
```

测试:

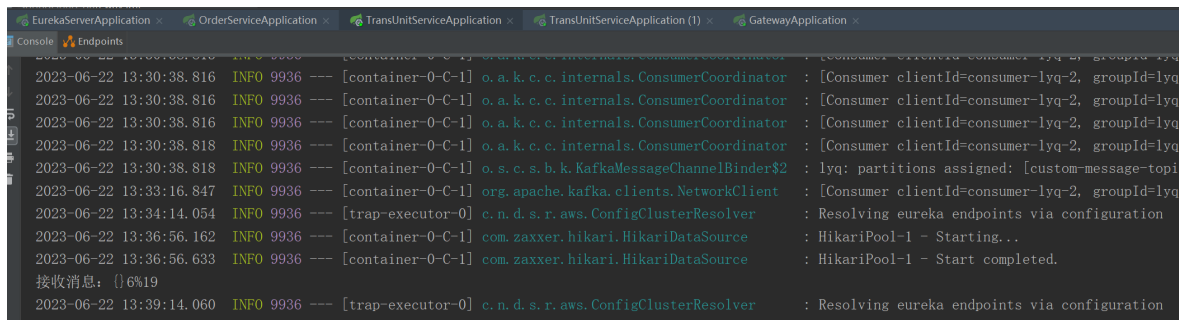
这里模拟了两个消费者transunitservice和transunitservice (1) 和一个生产者orderservice

当调用网关的orderservice服务接口进行订单分配时:

结果正常响应:

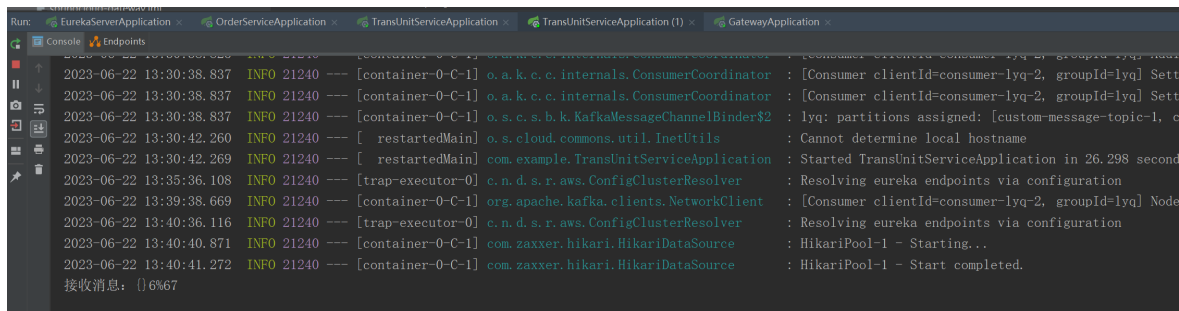


响应的微服务为transunitservice:



再次调用接口时:

响应的为另一个消费者transunitservice (1)



结果：

Order表

11	6	1	2023-06-22 1	河南省南阳辽宁省沈阳市沈公路	2023-06-2	运输中	19	18224426057	1363876812	程美玲	贾鹏飞	鞋子
12	6	2	2023-06-22 1	黑龙江省江苏省南京市秦空运	2023-07-0	运输中	67	18224426057	1371867523	张晓燕	谷建国	电脑

Transunit表

6	逸夫陆运1队	(Null)	(Null)	2	300	62
---	--------	--------	--------	---	-----	----

Records表：

11	已发货，运输中	2023-06-22 13:36:57
12	已发货，运输中	2023-06-22 13:40:41

业务逻辑正确