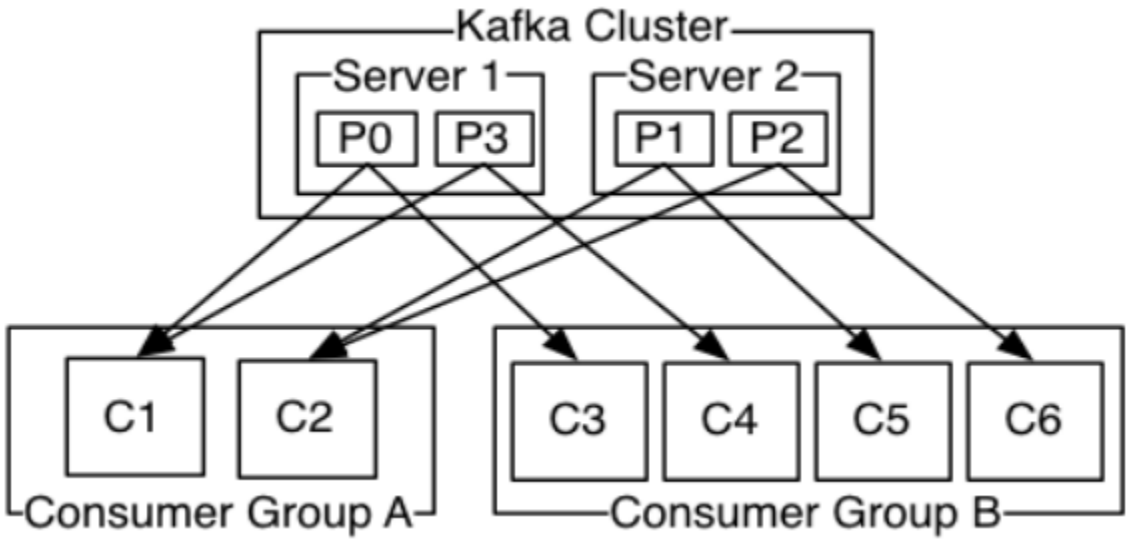
Kafka 培训课程：消费机制

目录

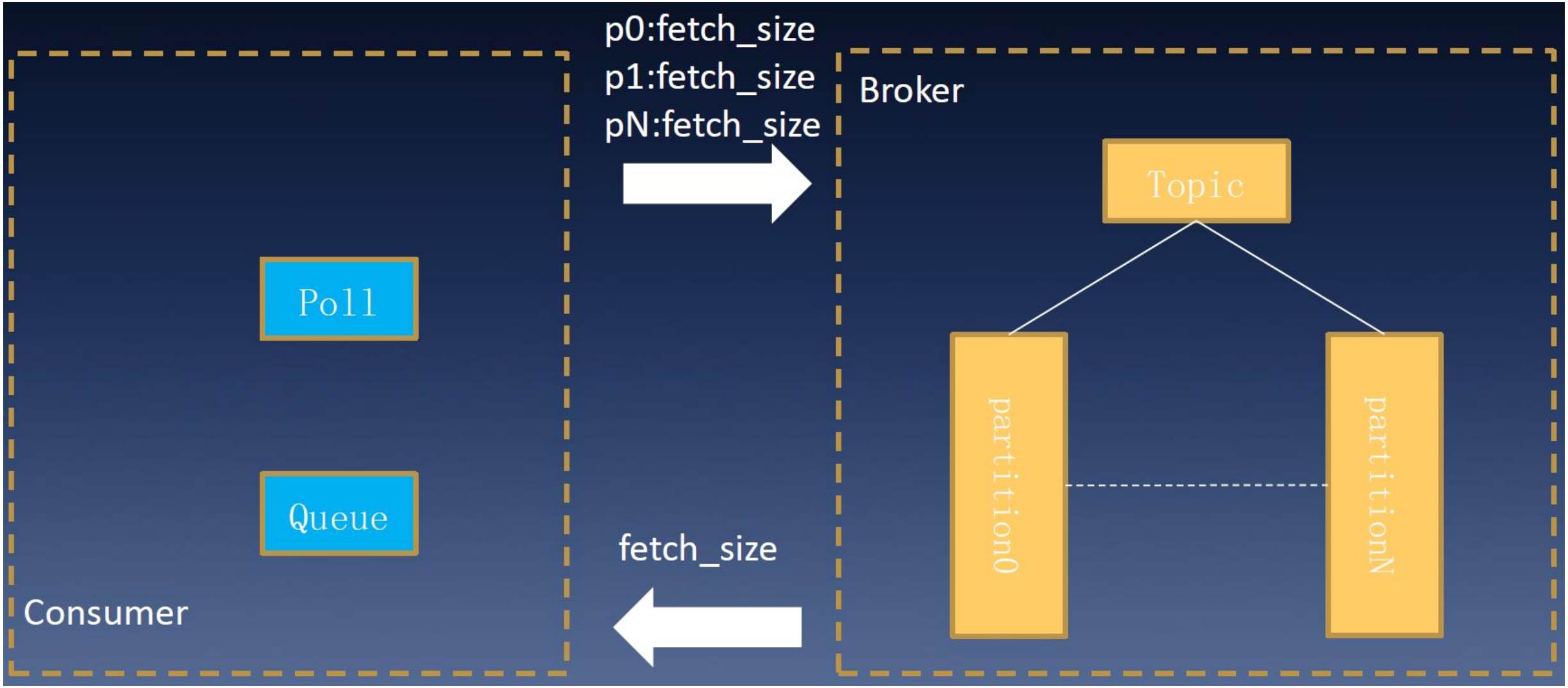
1.1 基本概念

1. **Consumer：kafka消费者负责拉取消息和确认消息，分new consumer和old consumer**
2. **Group：**每个消费者都属于一个消费组内，通过消费组概念可以实现Topic消息的广播（发给所有消费组）或者单播（组内消息均衡分担）。
3. **Rebalance：**组内的消费者以Topic分区个数进行均衡分配，所以组内消费者最多只能有分区个数的消费者。
4. **assign模式：**手工分配消费分区
5. **Subscribe模式：**自动分配消费分区

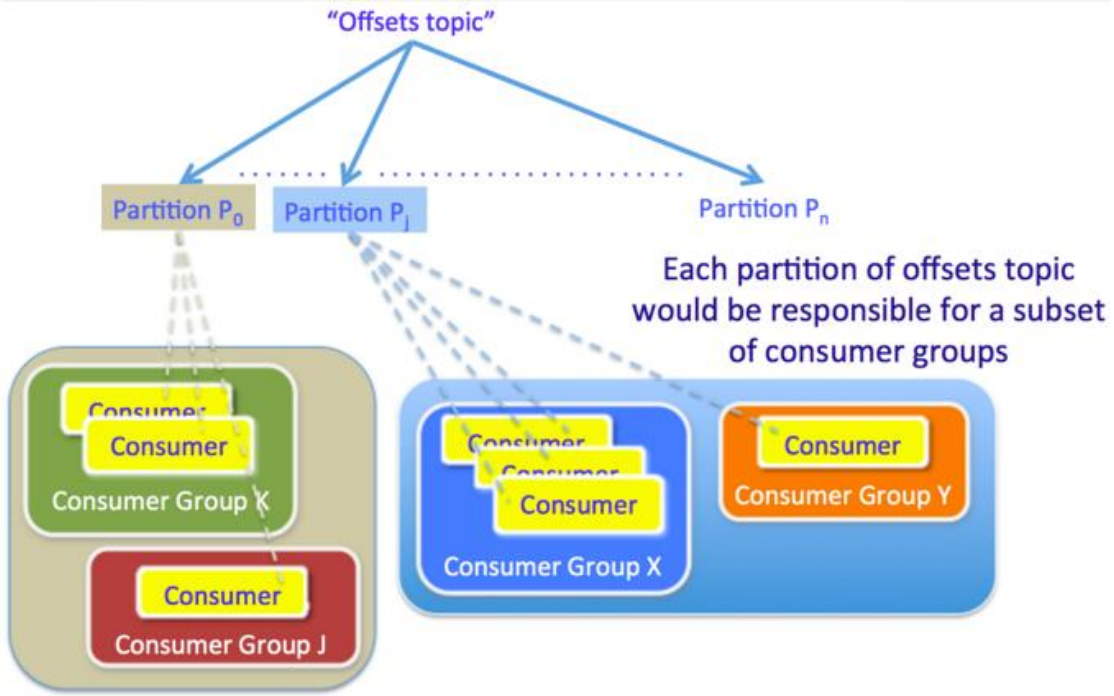


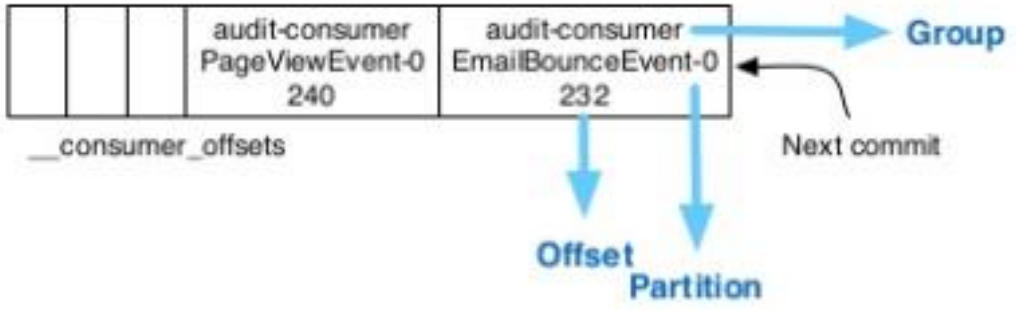
1.2 基本概念：消费模型

消费模型：消费者采用Pull模式进行消费，方便消费进度记录在客户端，服务端无状态。

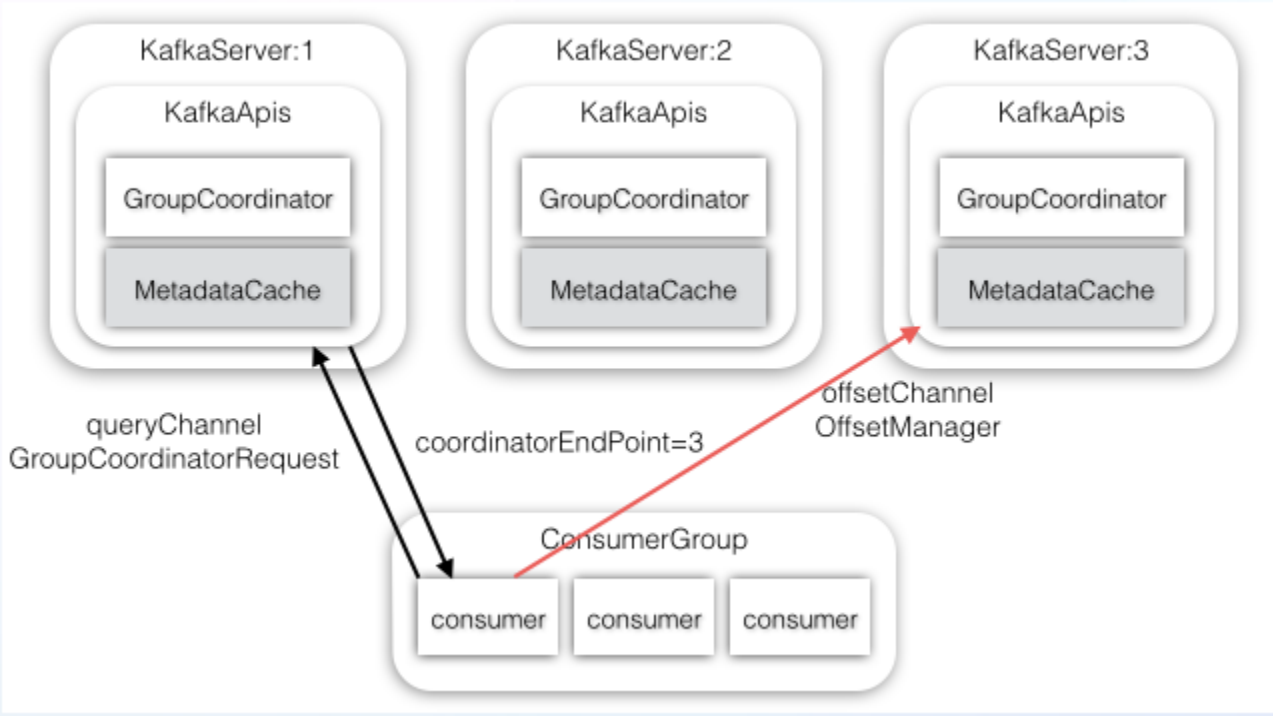


1.3 基本概念：消费进度管理机制

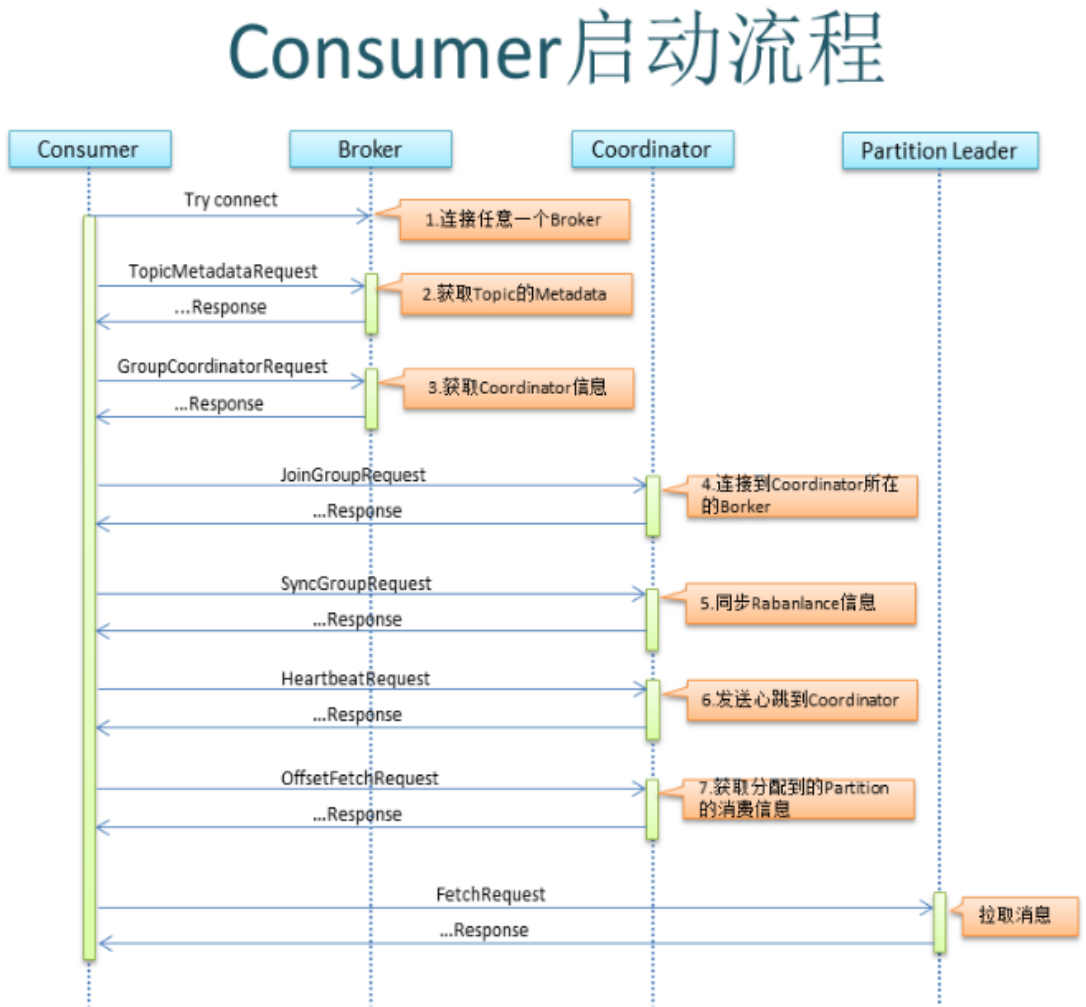




2 消费机制

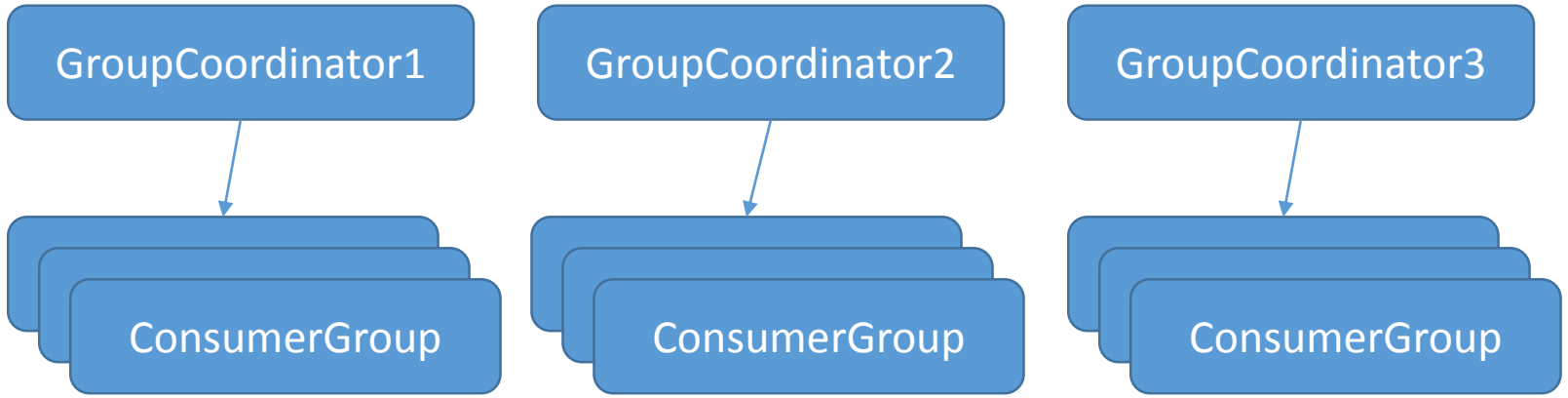


1. 每个消费组里面的消费者都需要先查找一个协调者
2. 消费者加入到这个组内，主要目的是为了对分区进行分配
3. 分配分区完毕后，再查找各自分区的Leader，进行消费



3 GroupCoordinator：职责

- 处理JoinGroupRequest、SyncGroupRequest完成partition分配
- 维护 _consumer_offset，管理消费进度
- 通过心跳检查消费者状态

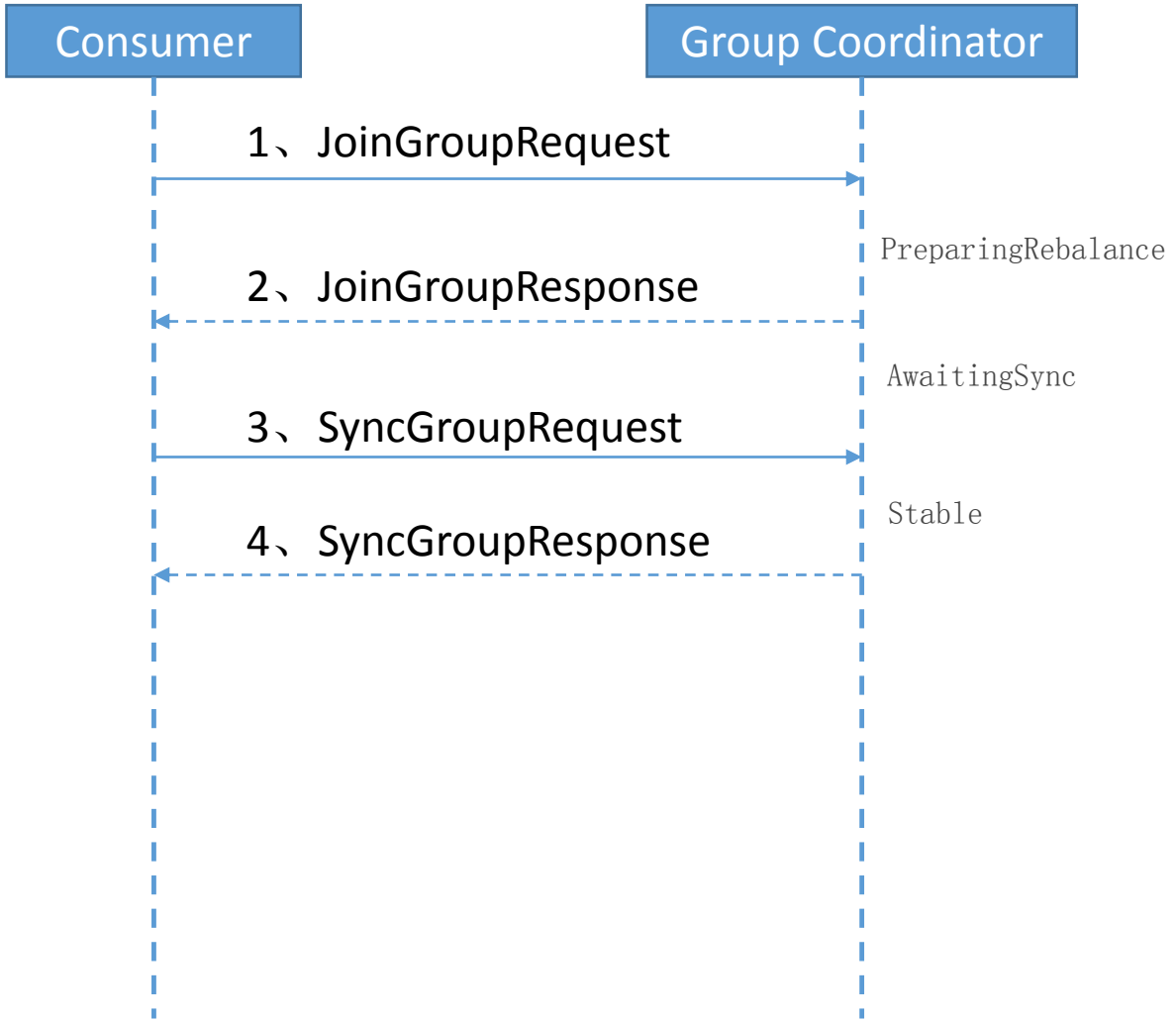


4.1 Group Rebalance：触发条件

- 新consumer加入group
- 有consumer退出：主动leave、宕机、网络故障等
- Topic分区数变化
- Consumer调用unsubscrible

4.2 Group Rebalance：rebalance流程

- Consumer 发送JoinGroupReuest请求，服务端Group进入PreparingRebalance状态
- Consumer 发送SyncGroupRequest请求，服务端Group进入Stable状态
  - 如果是group leader，计算分配分区，通过SyncGroupRequest发送至Coordinator
  - 非group leader在SyncGroupResponse中获取分配的分区



4.3 Group Coordinator计算

- 计算__consumers_offsets topic分区，计算公式：

  __consumers_offsets partition# = Math.abs(groupId.hashCode() % groupMetadataTopicPartitionCount) 注意：

  groupMetadataTopicPartitionCount由offsets.topic.num.partitions指定，默认是50个分区。

- 该分区leader所在的broker就是被选定的coordinator

5.1 参数配置

参数	默认值	推荐值	说明
max.partition.fetch.bytes	1048576	默认值	每次fetch请求中，针对每次fetch消息的最大字节数。这些字节将会督导用于每个partition的内存中，因此，此设置将会控制 consumer所使用的memory大小。这个fetch请求尺寸必须至少和server允许的最大消息尺寸相等，否则，producer可能发送的消 息尺寸大于consumer所能消耗的尺寸。
fetch.min.bytes	1	默认值	每次fetch请求时，server应该返回的最小字节数。如果没有足够的数据返回，请求会等待，直到足够的数据才会返回。
fetch.wait.max.ms	500	默认值	如果没有足够的数据能够满足fetch.min.bytes，则此项配置是指在应答fetch请求之前，server会阻塞的最大时间。

5.2 使用规范

1. consumer owner线程需确保不会异常退出，避免客户端没有发起消费请求，阻塞消费。
2. 确保处理完消息后再做消息commit，避免业务消息处理失败，无法重新拉取处理失败的消息。
3. consumer 不能频繁加入和退出group，会导致consumer 频繁做rebalance，阻塞消费。
4. consumer 数量不能超过topic分区数，否则会有consumer拉取不到消息。
5. consumer 需周期poll，维持和server的心跳，避免心跳超时，导致consumer频繁加入和退出，阻塞消费。
6. consumer 拉取的消息本地缓存应有大小限制，避免OOM
7. Kafka不能保证消费重复的消息，业务侧需保证消息处理的幂等性
8. 消费线程退出要调用consumer的close方法，避免同一个组的其他消费者阻塞sesstion.timeout.ms的时间

6 代码示例

```
public class KafkaConsumerDemo
{
    public static void main(String[] args)
    {
        if (args.length != 3)
        {
            throw new IllegalArgumentException("usage: dms.kafka.demo.KafkaProducerDemo bootstrap-servers topic-name group-name.");
        }

        Properties props = new Properties();
        props.put("bootstrap.servers", args[0]);
        props.put("group.id", args[2]);
        props.put("enable.auto.commit", "true");
        props.put("auto.offset.reset", "earliest");
        props.put("auto.commit.interval.ms", "1000");
        props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
        consumer.subscribe(Arrays.asList(args[1]));
        while (true)
        {
            ConsumerRecords<String, String> records = consumer.poll(200);
            for (ConsumerRecord<String, String> record : records)
                System.out.printf("offset = %d, key = %s, value = %s%n", record.offset(), record.key(), record.value());
        }
    }
}
```

实战演练

- 创建topic：topic-test01
- 编写代码生产消息
- 编写代码消费消息

Thank You