Kafka消费者不消费数据

背景：

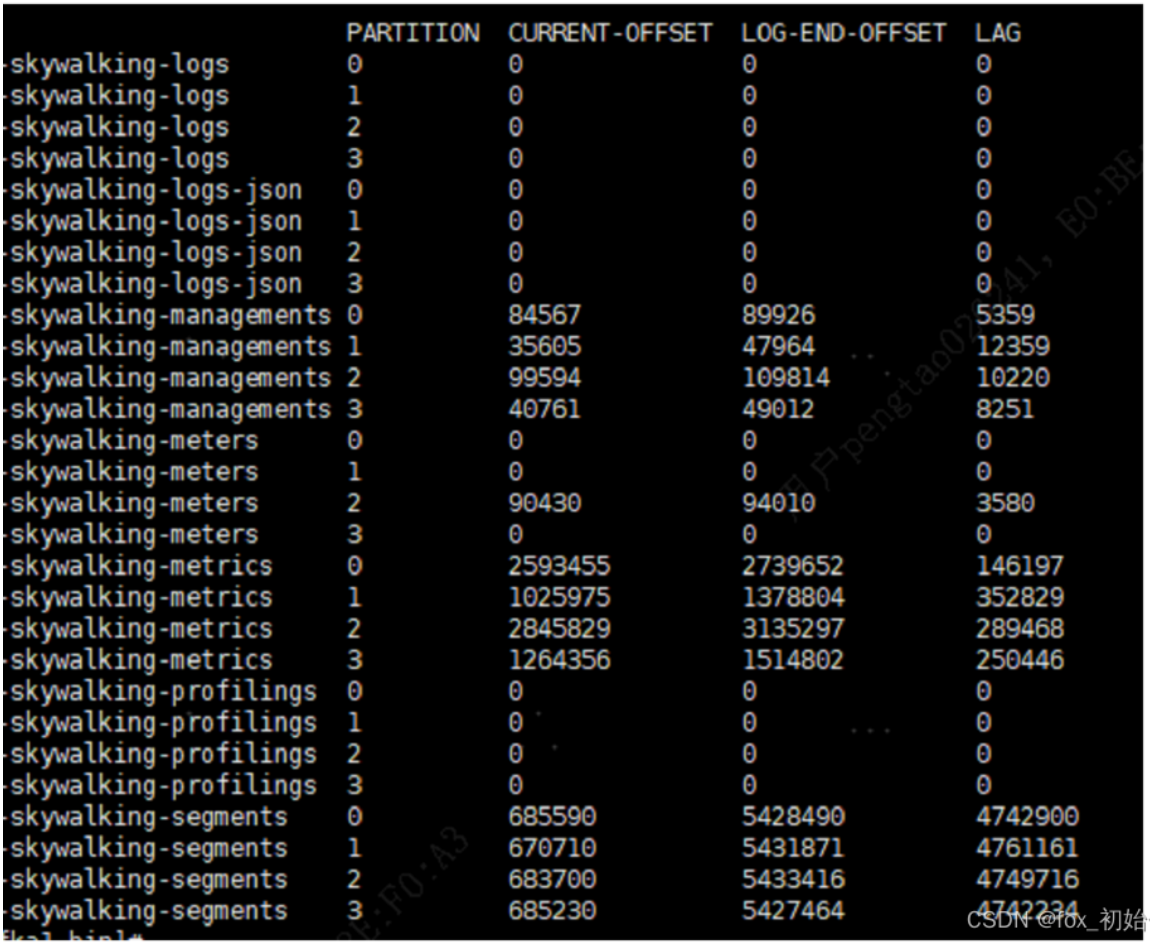
使用Skywalking agent+Kafka+ES进行应用监控。

现象：

公司使用Skywalking在开发测试环境中Kafka顺利消费数据，到了UAT环境一开始还正常，后面接入了更多的应用后出现了问题：Skywalking OAP服务正常但是ES里不再有数据。

排查：

通过查看消费者消费Kafka数据的情况可以看到，数据出现了积压。



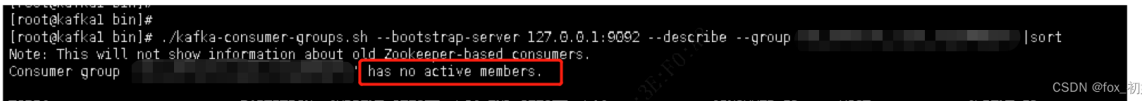
	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG
-skywalking-logs	0	0	0	0
-skywalking-logs	1	0	0	0
-skywalking-logs	2	0	0	0
-skywalking-logs	3	0	0	0
-skywalking-logs-json	0	0	0	0
-skywalking-logs-json	1	0	0	0
-skywalking-logs-json	2	0	0	0
-skywalking-logs-json	3	0	0	0
-skywalking-managements	0	84567	89926	5359
-skywalking-managements	1	35605	47964	12359
-skywalking-managements	2	99594	109814	10220
-skywalking-managements	3	40761	49012	8251
-skywalking-meters	0	0	0	0
-skywalking-meters	1	0	0	0
-skywalking-meters	2	90430	94010	3580
-skywalking-meters	3	0	0	0
-skywalking-metrics	0	2593455	2739652	146197
-skywalking-metrics	1	1025975	1378804	352829
-skywalking-metrics	2	2845829	3135297	289468
-skywalking-metrics	3	1264356	1514802	250446
-skywalking-profilings	0	0	0	0
-skywalking-profilings	1	0	0	0
-skywalking-profilings	2	0	0	0
-skywalking-profilings	3	0	0	0
-skywalking-segments	0	685590	5428490	4742900
-skywalking-segments	1	670710	5431871	4761161
-skywalking-segments	2	683700	5433416	4749716
-skywalking-segments	3	685230	5427464	4742234

编辑

由于没有设置消费者的参数，所以使用的是默认值max.poll.interval.ms是5分钟、 max.poll.records是500

编辑

目前积压数据远大于一次拉取消费的500，所以判断是因为消费者无法在等待时间内消费完数据，Consumer Group Coordination消费组判定当前消费者不在消费组内，所以查询消费者状态会出现消费者组不存在消费成员（如图符合判断）



编辑

目前解决方法：

max.poll.records改小或者 request.timeout.ms改大或者 request.timeout.ms改大，因为目前数据不稳定后续也只能通过数据量进行修改参数调优。重新开始消费，待后续观察。

结论：

由于数据量变大，消费者长时间不再请求数据，未向Group Coordinator发送心跳请求，所以kafka认为消费者已从消费组下线。所以不再进行消费。

学习：

之前知识浅浅了解了Rebalance，但没碰到过。所以借此好好学习一下。

一、什么是Rebalance

- Rebalance本质上是一种协议, 规定了一个Consumer Group下的所有consumer如何达成一致,来分配订阅Topic的每个分区。
- Rebalance发生时, 所有的Consumer Group都停止工作, 直到Rebalance 完成。

二、触发条件

① 组成员个数发生变化

- 新的消费者加入到消费组
- 消费者主动退出消费组
- 消费者被动下线。比如消费者长时间的GC, 网络延迟导致消费者长时间未向Group Coordinator发送心跳请求, 均会认为该消费者已经下线并踢出（本次问题出现的原因）

② 订阅的Topic的Consumer Group个数发生变化

③ Topic 的分区数发生变化

三、Rebalance的弊端

1. rebalance的时候消费组内的所有消费者都不能处理消息
2. 消费组内的消费者越多rebalance时间越长
3. Rebalance 效率不高。当前 Kafka 的设计机制决定了每次 Rebalance 时，Consumer Group 下的所有成员都要参与进来，而且通常不会考虑局部性原理，但局部性原理对提升系统性能是特别重要的。

四、如何避免Rebalance

从触发条件可以看到，① 、 ② 、③基本都是可以认为尽量避免也就是提前根据数据量规划好消费者数量，主要是① 中的第三个，需要靠kafka的参数去调整

```
# 心跳相关
session.timeout.ms = 6s
heartbeat.interval.ms = 2s
# 消费数量(默认500)
max.poll.records


# 消费时间(默认300000)
request.timeout.msmax.poll.interval.ms=300000
```

session.timeout.ms

用于检测工作程序故障的超时。工作人员定期发送心跳以向代理指示其活跃性。如果在此会话超时到期之前代理没有收到心跳，那么代理将从组中删除工作人员并启动重新平衡。请注意，该值必须在代理配置中由 `group.min.session.timeout.ms` 和配置的允许范围内 `group.max.session.timeout.ms` 。

类型： 整数
默认： 10000（10 秒）
有效值：
重要性： 高的

编辑

heartbeat.interval.ms

使用 Kafka 的组管理设施时，与消费者协调器之间的心跳之间的预期时间。心跳用于确保消费者的会话保持活跃，并在新消费者加入或离开组时促进重新平衡。该值必须设置为低于 `session.timeout.ms` ，但通常应设置为不高于该值的 1/3。它可以调整得更低，以控制正常重新平衡的预期时间。

类型： 整数
默认： 3000（3 秒）
有效值：
重要性： 高的

编辑

max.poll.interval.ms：使用消费者组管理时调用 poll() 之间的最大延迟。这为消费者在获取更多记录之前可以空闲的时间量设置了上限。如果在此超时到期之前未调用 poll()，则认为消费者失败，组将重新平衡，以便将分区重新分配给另一个成员。对于使用达到此超时的非 null 的消费者 `group.instance.id` ，不会立即重新分配分区。相反，消费者将停止发送心跳，并且分区将在 `session.timeout.ms` 。这反映了已关闭的静态消费者的行为。

类型：整数 – **默认值**：300000 – **有效值**：[1,...] – **重要性**：中等

编辑

request.timeout.ms：配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在必要时重新发送请求，或者如果重试次数用尽，则请求失败。

类型：整数 – **默认值**：30000 – **有效值**： – **重要性**：高 – **更新模式**：只读

编辑

max.poll.records：单次调用 poll() 时返回的最大记录数。

类型：整数 – **默认值**：500 – **有效值**：[1,...] – **重要性**：中等

partition.assignment.strategy：类名称或类类型的列表，按偏好排序，受支持的分配者负责分区分配策略，当使用组管理时，客户端将使用该策略在消费者实例之间分配分区所有权。实现该 `org.apache.kafka.clients.consumer.ConsumerPartitionAssignor` 接口允许您插入自定义分配策略。

类型：列表 – **默认值**：类 org.apache.kafka.clients.consumer.RangeAssignor – **有效值**：非空字符串 – **重要性**：中等

编辑

五、Rebalance过程

Coordinator服务

- Group Coordinator 是一个服务, 每个 Broker 在启动的时候都会启动一个该服务 Group Coordinator 的作用是用来存储 Group 的相关 Meta 信息, 并将对应 Partition 的 Offset 信息记录到 Kafka 内置 Topic(__consumer_offsets)中
- Kafka 在0.9之前是基于 Zookeeper 来存储Partition的 offset 信息 (consumers/{group}/offsets/{topic}/{partition}), 因为 Zookeeper 并不适用于频繁的写操作, 所以在0.9之后通过内置 Topic 的方式来记录对应 Partition 的 offset。

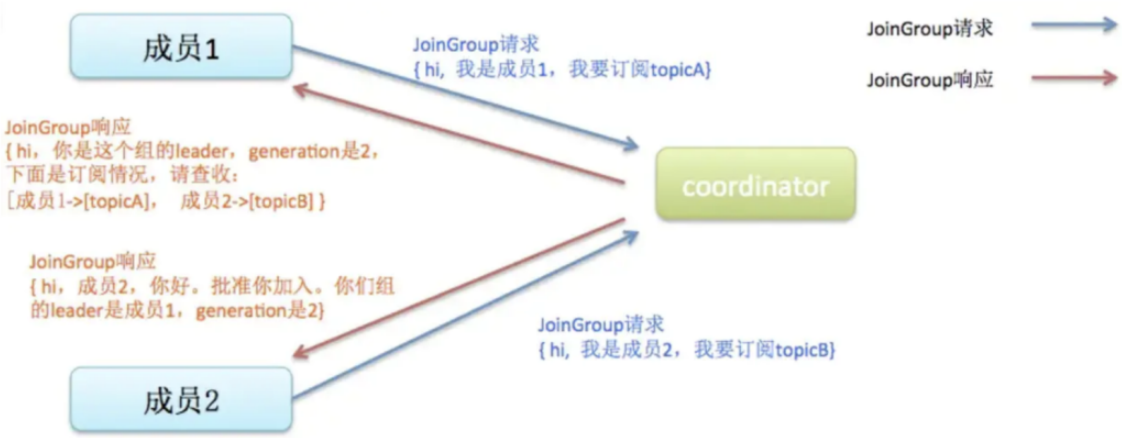
Rebalance过程分为两步：Join Group和 Sync Group。

Join Group

① 概述

- Join Group 顾名思义就是加入组。
- 这一步中, 所有成员都向 Coordinator 发送 JoinGroup 请求, 请求加入消费组。
- 一旦所有成员都发送了 JoinGroup 请求, Coordinator 会从中选择一个 Consumer 担任 Leader 的角色, 并把组成员信息以及订阅信息发给 Consumer Leader。
- 注意Consumer Leader 和 Coordinator不是一个概念。Consumer Leader负责消费分配方案的制定。

② 流程图



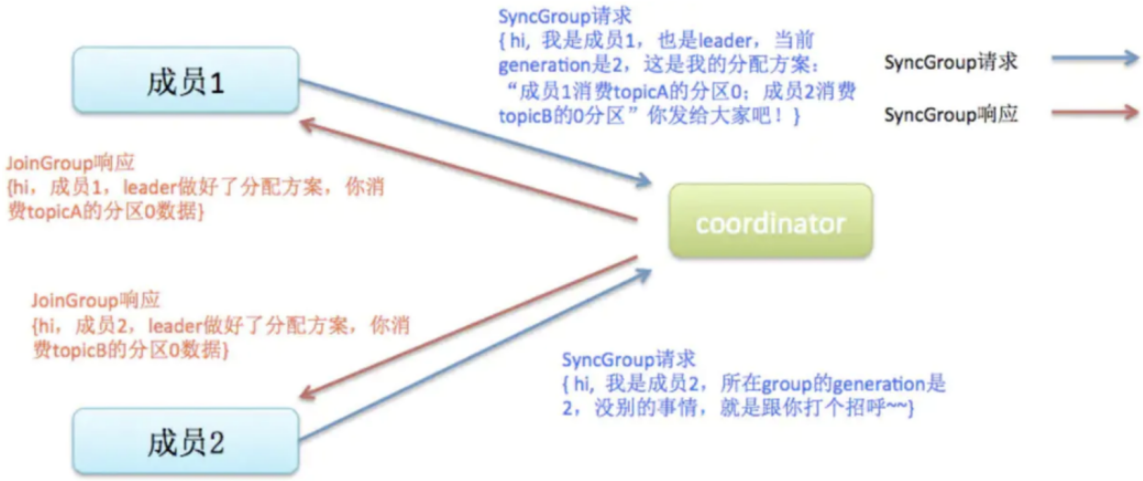
编辑

Sync Group

① 概述

- Consumer Leader 开始分配消费方案，即哪个 Consumer 负责消费哪些 Topic 的哪些 Partition。
- 一旦完成分配，Consumer Leader 会将这个方案封装进SyncGroup请求中发给 Coordinator。
- 非 Consumer Leader 也会发 SyncGroup 请求, 只是内容为空。
- Coordinator 接收到分配方案之后会把方案塞进SyncGroup的Response中发给各个Consumer。
- 这样组内的所有成员就都知道自己应该消费哪些分区了。

② 流程图



编辑

参考：Kafka学习笔记 NO.004 Kafka的Rebalance(重平衡) - 墨天轮