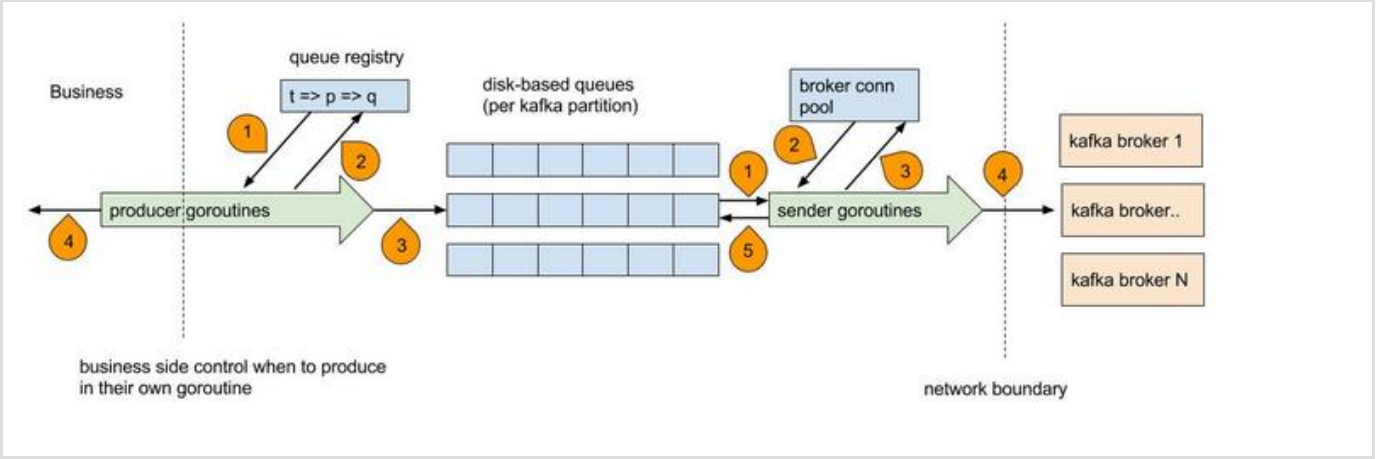
文 Kafka Agent 设计 - 可靠事件记录不是一件简单的事情

kafka **taowen** 3月9日发布



用 golang 编写，解决事件快速同时可靠入Kafka的问题。绿色表示goroutine（可能是不同的线程），蓝色表示共享的资源。所有对蓝色资源的访问需要加锁。磁盘队列使用内存映射文件实现ring buffer。kafka agent启动的时候有一个bootstrap的broker列表，同时发送的时候会与每个相关的broker维护一个到多个连接。但是kafka agent不读取也不监听zookeeper。

收益

- 业务方不需要等待kafka响应就可以继续干别的去（低延迟）
- 基于磁盘的队列（高可靠）
- 消息按partition排队后组大包并压缩发送（高吞吐）
- 因为发送到kafka之前业务侧已经拿到响应并返回，kafka的request/response可以简单地一去一回，不用基于corelation id回调通知业务侧（简单设计）

缺点是业务侧无法知道写入的kafka offset。

生产过程

业务方和 Kafka Agent 可以在同一个进程内，也可以通过本地unix domain socket沟通。如果是在同进程

内，事件直接在业务方的 goroutine 内产生并入磁盘队列。如果是在不同进程内，事件通过本地网络请求转发，由kafka agent代为入队列。入队列成功之后有返回，业务侧接到返回则认为事件已经可靠保存，Eventually会进入到Kafka中。

1. 产生事件，生成hash值。查询 queue registry 获得topic的partition数量。根据hash值计算落到parition号。根据topic和partition找到对应的磁盘队列。
2. 如果没有对应的磁盘队列，则新建，并保存回 queue registry。如果没有对应 topic 的 metadata 则需要查询并保存回 queue registry。
3. 事件写入磁盘队列，移动写入offset的指向位置。如果写入超过读取速度，则覆盖（丢弃掉旧的 event）。
4. 返回业务侧，告知已经可靠写入

异常流程

- 没有topic的metadata
- 没有对应的queue
- topic的partition数量可能增加，需要定时刷新
- 写入会超过读取的速度
- 写入超过读取速度的时候，读取可能正在进行中

发送过程

每个队列有一个对应的goroutine负责发送到kafka。

1. 定时唤醒发送goroutine，从自己负责的queue里读取一批事件（只会对应一个topic的一个partition，因为queue是对应topic和partition创建的）。
2. 当前goroutine里如果没有缓存对应的broker（parition的leader）的connection pool，则从全局连接池里拿一个pool。从pool里借一个conn。
3. 如果pool不存在则新建pool。
4. 发送消息到kafka broker，并同步等待其返回。然后归还connection到pool里。
5. 此时消息已经可靠写入kafka，移动读取的offset。

异常流程

- partition对应的broker还未知
- broker对应的pool还未创建
- 从broker里取得的connection已经损坏
- 写入时broker告知leader已经改变
- 移动offset的时候写入已经溢出

Reliable Event Logging

很多时候我们都希望可靠地记录事件，这些事件处理大致分为两类

日志监控

- 记录监控指标，用于告警
- 记录原始日志，用于定位问题
- 记录原始事件，用于离线统计分析

业务事件

- 离线计算发放奖励
- 触发一系列周边系统的动作

日志监控类的要求是迅速量大，可以丢个别的日志。业务事件类的要求是一条也不能丢，否则业务方要引入另外一个队列做离线对账（这队列不还是logging么，一般是业务自己的主数据库来承担）。业务事件的模式不能依靠上游来重试，因为主业务流程已经完成了，事件的写入和接下来的处理不应该来决定业务操作本身的成败。简单来说就是有一些业务希望自己业务本身成功了，事件必须可靠记录，同时基于事件的后续处理必须发生。

现在对kafka的使用方式，主要有两种

- kafka producer => kafka，实现方式是内存做buffer，然后批量写入
- log file => 日志采集 => kafka 用日志文件做缓冲

两种方式都不够完美，都无法满足使用方的要求。日志方式的问题是使用麻烦，而且性能开销大。直接用producer的方式几乎无持久化做缓冲，大部分时候就是靠内存buffer活着（除非能够忍受直接写远程kafka的延迟），根本没有持久化保证可靠性。

所以事实上目前基于kafka的应用架构里都无法把kafka做为一个可靠存储来用，不是因为kafka本身不可靠，而是做到不拖慢业务的同时可靠入库并不容易。或者讲究着把一个不可靠的data pipeline当作可靠的来用。

More on this: http://www.slideshare.net/JiangjieQin/no-data-loss-pipeline-with-apache-kafka-49753844

3月9日发布 更多

你可能感兴趣的文章

Structured Logging 需要更好的基础设施支持 2 收藏，987 浏览

docker运行kafka 1 收藏，664 浏览

mac安装kafka0.9.0.1 117 浏览

讨论区

使用评论询问更多信息或提出修改意见，请不要在评论里回答问题

提交评论

评论支持部分 Markdown 语法： `**bold**` `_italic_` `[link](http://example.com)` `> 引用` `` `code` `` `- 列表` 。
同时，被你 @ 的用户也会收到通知

本文隶属于专栏

taowen

I write code

taowen
作者

关注专栏

分享扩散：