Twitter 如何将 Kafka 当做一个存储系统

前言

当开发人员通过我们提供的 API 使用公开的 Twitter 数据时，他们需要可靠性、高效的性能以及稳定性。因此，在前一段时间，我们为 Account Activity API 启动了 Account Activity Replay API ，让开发人员将稳定性融入到他们的系统中。Account Activity Replay API 是一个数据恢复工具，它允许开发人员检索5天前的事件。并且提供了恢复由于各种原因而没有交付的事件，包括在实时交付期间服务器的宕机。

除了构建 API 来提供良好的开发者体验，我们还做了以下的优化：

- 提高我们工程师的生产力。
- 使系统易于维护。具体地说，我们尽量减少开发者、站点可靠性工程师和任何与系统交互的人的交互。

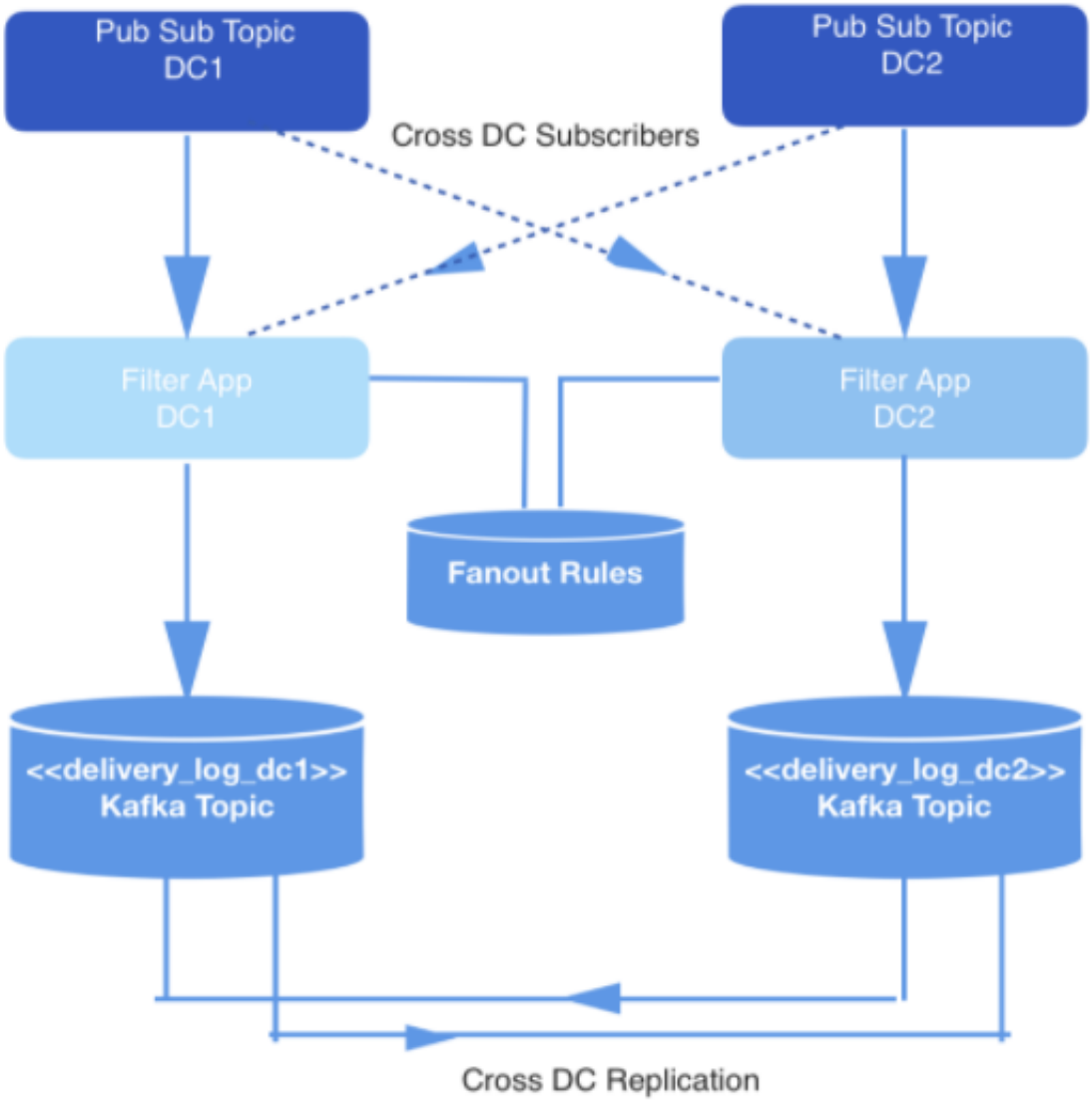
由于这些原因，在构建此 API 所依赖的回放（replay）系统时，我们利用了 Account Activity API 的实时系统的现有设计。这帮助我们重用已经存在的工作，并且最小化上下文切换和培训。

实时系统是利用了发布-订阅架构（publish-subscribe architecture）。因此，为了保持这一架构，在构建存储层的时候，我们需要重新思考传统的流技术——Apache Kafka。

背景

实时事件在两个数据中心（DCs）中产生，产生这些事件后，它们被写入 pub-sub 主题中，为了实现冗余目的，事件会写到两个数据中心。

并非所有事件都应该交付，因此内部应用程序将对这些主题中的事件进行筛选，根据键值存储中的一组规则检查每个事件，并决定是否应该通过我们的公共 API 将事件交付给给定的开发人员。事件是通过 webhook 方式交付，开发人员拥有的每个 webhook URL 都由一个唯一的 ID 标识。



如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：iteblog_hadoop

存储和分区

当构建一个需要存储功能的重放系统时，通常我们会使用基于 Hadoop 和 HDFS 的架构。而我们这里选择了 Apache Kafka，主要基于以下两个原因：

- 我们现有的实时系统采用的是发布-订阅架构；
- 重放系统存储的事件量不是 PB 级的，我们存储的数据不超过几天。此外，MapReduce 任务读取数据的效率也没有 Kafka 高，也更慢，这不能满足开发人员的期望。

我们利用实时管道（real-time pipeline）来构建回放管道（replay pipeline），首先要确保应该交付给每个开发者的事件都存储在 Kafka 上。我们把对应的 Kafka 主题称为 delivery_log，每个数据中心都有一个。然后，这些主题在两个数据中心交叉复制，以确保冗余，从而使得任一个数据中心都可以处理重放请求。

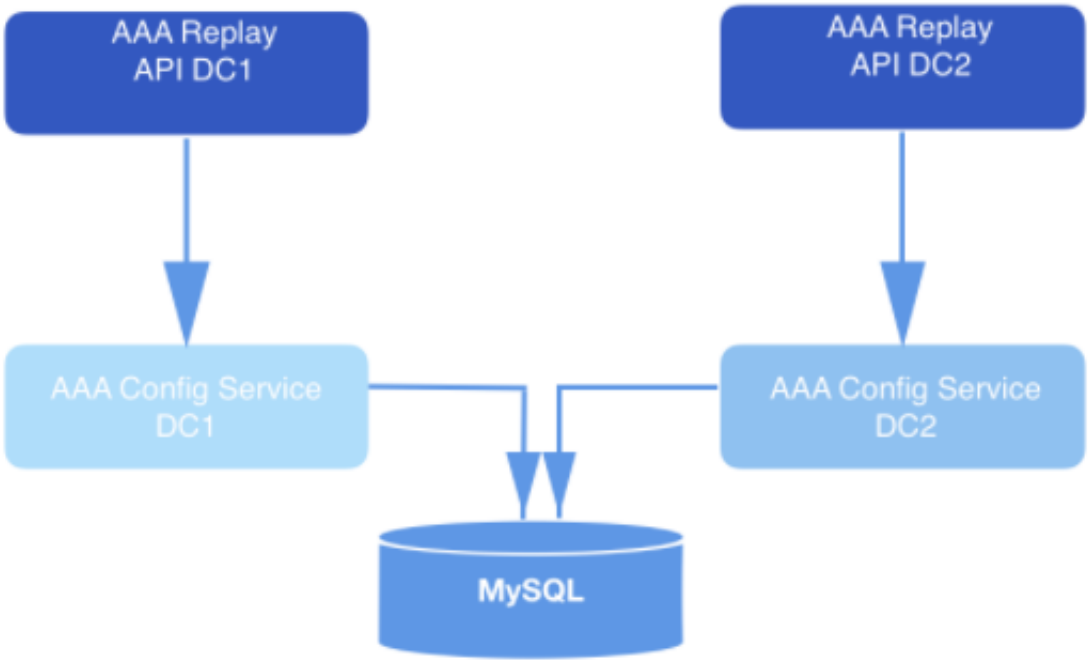
在这个 Kafka
主题中，我们使用默认的分区策略（也就是哈希分区）创建了多个分区。开发人员的 webhookId 哈希值对应分区 ID，它是每个记录的键。我们考虑过使用静态分区，但最终决定不使用它，因为如果一个开发人员生成的事件多于其他开发人员，那么一个分区的数据多于其他分区的风险就会增加。相反，我们选择固定数量的分区，使用默认分区策略来分散数据。有了这个，我们减轻了分区不平衡的风险，并且不需要读取 Kafka 主题上所有分区。相反，根据请求进入的 webhookId，重放服务确定要读取的特定分区，并为该分区启动一个新的 Kafka
消费者。主题上的分区数量不会改变，因为这会影响键的散列以及事件的分布方式。

根据每个时间段读取的事件数量，我们选择使用固态磁盘（SSD）。我们选择它而不是传统的机械硬盘（HDD），以获得更快的处理速度，并减少寻道和访问时间带来的开销。最好使用 ssd 以获得更快的读取速度，因为我们访问的数据很少重复使用，因此无法从页面缓存优化中获得好处。

为了减少存储空间，我们使用 snappy
作为压缩算法对这些数据进行压缩。因为我们知道大部分的事情都是在消费端进行的。我们选择 snappy 是因为它的解压速度比其他 kafka 支持的压缩算法 gzip 和 lz4 快。

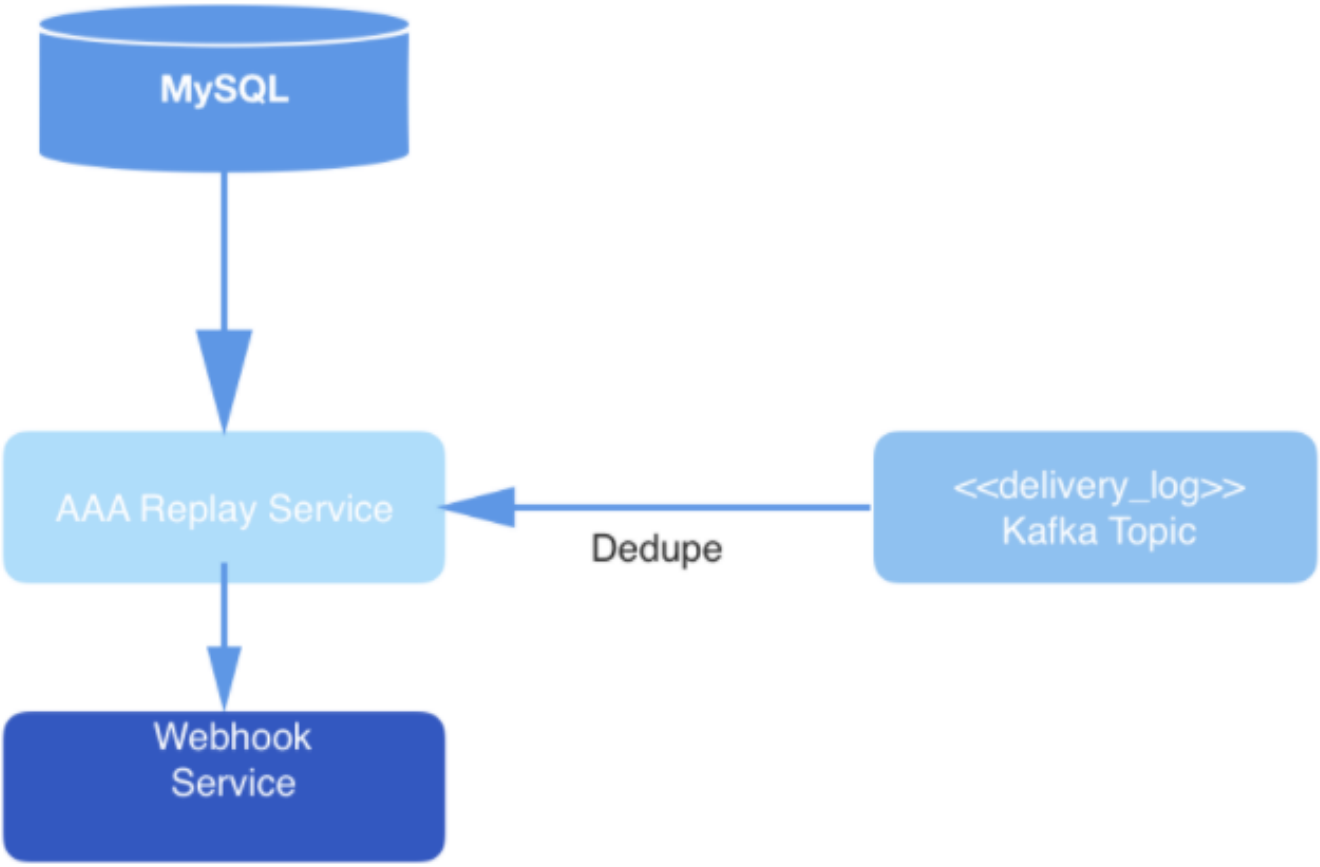
请求与处理

在我们设计的系统中，一个 API 调用发送重放请求。通过请求的参数里面，我们可以获得 webhookId 和应该重放事件的日期范围。这些请求被持久化到 MySQL 中并进入队列，直到重放服务处理它们为止。请求上的日期范围用于确定要开始读取的分区上的偏移量，消费者上的 offsetForTimes 函数用于获取偏移量。



重放服务实例处理每个重放请求，重放服务实例使用 MySQL
相互协调，以处理数据库中的每个重放请求。每个重放 worker 定期轮询 MySQL 以了解应该处理的作业。每个请求都有一个完整的生命周期：未被选中处理的请求处于打开状态（OPEN STATE）；刚刚出队的请求处于启动状态（STARTED STATE）；正在处理的请求处于正在进行的状态（ONGOING STATE）；已完成的请求将转换为已完成状态（COMPLETED STATE）。重放 worker 只选择尚未启动的请求，也就是处于 OPEN 状态的请求。

在 worker 将请求从队列中取出处理后，worker 会定期地在 MySQL
表写入一条时间戳数据作为心跳（heartbeats），以表示重放作业仍在进行中。当重放 worker 实例在处理请求时宕机的情况下，这样的作业将重新启动。因此，除了处理处于打开状态（OPEN STATE）的请求，重放 worker 还可以选择处理已启动（STARTED）或正在进行（ONGOING）状态的作业，这些作业在几分钟内没有心跳信息。



事件从主题中读取后会进行去重操作，然后发布到客户的 webhook URL 中。数据去重是通过维护正在读取事件的哈希缓存来完成的。如果遇到具有相同哈希值的事件，则不会传递该事件。

总之，我们的解决方案并不是我们熟悉的把 Kafka
当做传统的实时、流处理的组件。然而，我们成功地使用 Kafka 作为存储系统来构建一个 API，在事件回放中优化客户体验和数据访问。利用实时系统的设计使我们的系统易于维护。此外，我们客户数据的恢复速度与我们的预期一致。

本文翻译自：Kafka as a storage system