事务工作流

在第 2 章“Flume Channel 中的事务”一节中讨论过，Flume Channel 是事务性的。事务本质上是原子性写入 Channel 的批量事件。事件要么全部批量地存在在 Channel 要么全都不存在。事务提供了重要的保证，它能知晓什么时候事件被写入 Channel 或从 Channel 移除。例如，Sink 可以从 Channel 中读取一个事件，试图把它写到 HDFS 但是失败了，在这种情况下，事件应该回滚到 Channel，这样就可以被这个可用的 Sink 或另外一个 Sink 读取并写到 HDFS。

只有在事务提交后事件才被移除能保证事件不会丢失，即使写失败一次，此时 Sink 可以回滚该事务。事务可以有一个或多个事件，但由于性能的原因总是推荐每个事务有相当大数量的事件。

批量写入 Channel 是很重要的，尤其是持久的 Channel。甚至 Agent 或机器重启的情况下，持久 Channel 也能保证没有数据丢失，所以它们必须在事务提交期间刷新和同步所有缓冲事件的数据到磁盘，每批量发生一次。同步到磁盘是昂贵和耗时的操作，应该只在相当大部分的数据写入页面缓存时完成。另外，同步到磁盘需要时间，包括在实际同步之前重要的系统调用的消耗，这一切都随着时间的推移而增加。每一个这样的批量也表示为一个事务，使得事务对于性能以及可靠性越来越重要。

每个 Channel 可以有多个 Source 和 Sink，分别写入 Channel 和从 Channel 读取。Source 和 Sink 关于事务以稍微不同的方式工作。Source 不直接处理事务；相反，Source 的 Channel 处理器代表它处理事务。Channel 处理器处理事务的工作方式与 Sink 几乎是相同的（除了 Sink 是从 Channel 读取数据，而 Channel 处理器是将数据放入 Channel）。

Sink 用 Channel 发起事务是通过调用 Channel 的 `getTransaction` 方法，这个方法返回 `Transaction` 的一个实例。然后 Sink 开始调用事务对象，它允许 Channel 设置任何事务所需的内部状态。通常，这包括队列的创建，以用来暂时托管事件直到事务完成。

81

事务一旦开始，Sink 在 Channel 上调用 `take` 方法（Channel 处理器的情况下是 put 方法），直到 Sink 准备提交事务。一旦 Sink 读取一个事件，该事件将不会被用于相同的或另一个 Sink，除非事务回滚。

由于性能的原因 Sink（和 Channel 处理器）通常会将一些事件批量放入一个事务中。一旦 Sink 完成了它的批量任务，Sink 就对事务调用 `commit` 方法。一旦 sink-side 事务（只进行读取的事务）被提交，该事务中的事件被 Channel 标记为删除，也不能被其他 Sink 再次使用。一旦 source-side 事务（Channel 处理器所拥有的事务）被提交，Channel 中的事件就是安全的。另外，这意味着只有当 Sink 读取完事件并提交，这些事件才能从