File Channel 的设计和实现 *

前面已经讨论过，File Channel 持久化每个事件到磁盘，这保证了即使 Agent 或机器宕机重启时的事件也是可用的。File Channel 也持久化每个执行到磁盘的操作。这意味着当 Channel 关闭的时候，Channel 可以以相同的顺序重新回放每个记录，就像让它自己回到相同的状态一样。当 Channel 完成回放记录，就为正常的操作做好了准备。在这一节，我们将更加详细地探讨 File Channel 的内部原理。

File Channel 维护了两个独立的数据结构：Flume 事件队列（将被称为队列）和 *write-ahead* 日志（WAL）。每次写入、读取、提交和回滚都表示为一个事务事件记录（后面称为记录），记录的类型表示了操作——写入、读取、提交或回滚。每个 File Channel 在 WAL 中记录为一条记录。每次一个事件被写入 Channel，写入的记录就会写入 WAL 中，即使该事务实际上并没有被提交。

类似地，对于读取操作，会写出到一条读取记录。每条记录有一个唯一的单条递增的 ID，写入 ID，记录了该条记录是什么时候写入到 WAL 中的。每条记录也包含了事务的唯一 ID，记录是事务的一部分。因为每条写入（或读取）记录包含事务 ID，那么可以 91 通过映射事务对应的提交或回滚记录，得知哪些事件被提交，哪些事务被回滚。

通过读取 WAL 并以实际发生的顺序执行每条操作（可以从操作的写入 ID 推算得到），我们可以在任何时候重新构建 Channel 的文件状态。当 Channel 通过读取所有的数据文件被完整重构，这称为全回放（*full replay*）。全回放通常是消耗时间和破坏性的，因为需要读取所有的数据文件，WAL 中的每个操作必须执行。这时尤其如此，当 WAL 包含数百万条写入和读取记录——即使最后的状态包含很少的事件，每条记录都必须被读取并存入内存，直到我们读取到相应的提交或回滚记录。

每次写入发生时，写记录写入到磁盘。使用文件中记录的文件 ID 和偏移，Channel 为该记录构建一个唯一的 Flume 事件指针。每次事件写入到 Channel，指针向事务指出了在本地内存队列存储的记录。当事务被提交，本地队列的指针被复制到 File Channel 主队列的尾部：Flume 事件队列。因此，那个队列代表了任何时候 Channel 的实时状态。

当 Sink 从 Channel 读取事件时，队列的头部被移除，指针失效。该事件随后被存储到该事务的本地队列。对于提交，本地队列被忽略，因为事件已经被完全移除。对于回滚，事件被后推到队列。实际上队列是内存映射文件——映射缓冲区更新，并且在检查点中被推送到磁盘。

在启动时，Flume 开始一个被称为回放的程序，可以让 Channel 回到之前停止时的确切状态。启动时仅仅通过内存映射检查点文件，队列就可以被加载到内存。然后从检查点的时刻从偏移量读取数据文件（偏移量被记录在检查点时刻数据文件的元数据中），并且