论是直接或者通过其他 Flume Agent），通过简单增加更多的 Flume Agent 就能够扩展服务器的数量并将大量数据写入到 HDFS。

每个 Flume Agent 有三个组件：*Source*、*Channel* 和 *Sink*。*Source* 负责获取事件到 Flume Agent，而 *Sink* 负责从 Agent 移走事件并转发它们到拓扑结构中的下一个 Agent，或者到 HDFS、HBase、Solr 等。*Channel* 是一个存储 Source 已经接收到的数据的缓冲区，直到 Sink 已经将数据成功写入到下一阶段或者最终目的地。

实际上，在一个 Flume Agent 中的数据流以下面几种方式运行：生产 / 接收的数据源写入数据到一个或者更多 Channel，一个或者多个 Sink 从 Channel 读取这些事件，然后推送它们到下一个 Agent，或者存储和索引系统。

Flume Agent 可以被配置成在数据被写入到目的地之前，从管道的一个 Agent 发送数据到另一个 Agent。一旦数据到达 Flume Agent，数据的持久性完全取决于 Agent 使用的 Channel 的持久性保证。在一般情况下，当一个 Flume agent 被配置成使用任何的内置 Source 或 Sink 以及一个持久的 Channel，Agent 保证不会丢失数据。凭借独立 Agent 不丢失数据的优势，Flume 管道也不会丢失任何数据。

如果 Flume 管道中有意想不到的错误 / 超时并进行了重试，Flume 会产生重复的数据最终被写出。如果托管持久 Channel 的硬盘遇到不可恢复的失败，Flume 可能会因为硬盘故障而丢失数据。虽然可能会引起冗余，Flume 允许用户通过冗余流复写事件，以确保硬盘和 Agent 失败是可控的。所以，用户可能必须通过做一些后期处理来确保妥善处理这些冗余。

Flume 是否适合呢？

Flume 将数据表示为事件。事件是非常简单的数据结构，具有一个主体和一个报头集合。事件的主体是一个字节数组，通常是 Flume 传送的负载。报头被表示为一个 map，其中有字符串 key 和字符串 value。报头并不是用来传输数据的，只为了路由的意图和跟踪发送事件的优先级和严重性。报头也可以用于给事件增加事件 ID 或者 UUID。

每个事件本质上必须是一个独立的记录，而不是记录的一部分。这也要求每个事件要适应 Flume Agent JVM 的内存。如果使用 File Channel，应该有足够的硬盘空间来支持。如果数据不能表示为多个独立记录，Flume 可能不适用于这个用例。

Flume 主要是从大量生产服务器将数据推送到 HDFS、HBase 等。在 Flume 不是很适合的场景下，通常有更容易的方法，例如 Web HDFS 或者 HBase HTTP API，都可以用于写数据。如果只有少量的生产服务器生产数据并且数据不要求实时写出，那么通过 Web