很明显，失败导致每一层都缓冲数据，直到 Channel 填满，此时它开始回退到前一层，直到所有层都被填满。此时，客户端开始看到错误，那么客户端必须现在通过缓冲数据或删除数据进行处理，从而导致数据丢失。在这种情况下，容量必须进行设计，停机时不应该创建发生这种情况的情景。我们将在第 8 章谈论容量的设计。

批量的重要性

当事件被发送到 Source，Source 通过网络从 RPC Sink 或远程客户端接收数据，Source 通过委托这个任务给 Channel 处理器，在单个事务中成批写出所有事件。

27 对于 File Channel，这是持久的 Channel 实现，每个事务的提交导致一个**文件同步**操作[fsync]。**文件同步**是 POSIX 标准定义的系统调用，用于告知操作系统为一个特定的文件描述符刷新内部缓冲区的所有数据到磁盘。如果每个事务写入的数据量很小，启动一个系统调用（在时间和资源消耗方面），切换到内核空间，在实际同步到磁盘之前刷新所有缓冲数据，将占据**文件同步**调用本身总成本非常高的分额。

对于 Memory Channel，这是内存 Channel 的实现，与整个 Channel 同步相关的成本，在提交期间发挥作用，但这远远小于**文件同步**调用的开销。

由于实际调用相关的元数据和所有额外的 TCP 开销，RPC 调用有额外开销。当发送的数据量非常小时，这些开销是每个 RPC 调用成本的一大部分，导致不必要的网络利用率等。为了避免这种开销，单个 RPC 调用中批量处理几个事件（当然，除非每个事件本身就很大）或从远程客户端写入，总是一个好主意。

尽管 Flume 的 RPC 客户端和 RPC Sink 支持没有批量或批处理大小为 1 来写事件，为避免比必要的开销支付更多的成倍的开销成本，捆绑事件到合理大小的批次几乎总是一个好主意。理想的批量大小将取决于具体的用例，但是对于最大为几个字节的事件，批量大小在 100 和 1000 之间通常性能很好（虽然特定的硬件、网络和其他因素影响该值，在测试过不同的值，且找到一个匹配所需性能的值，才最后设置）。

批处理影响 RPC Sink 和其他通过网络写数据的 Sink 的性能。如前所述，RPC Sink 性能受多个因素影响。甚至对于 HDFS Sink，当每一批量提交，Flume 刷新事件到所有数据节点的内存。因此，为所有 Sink 使用合理的批量大小总是一个好主意。

有控制批量大小的 Source，例如 Exec Source、JMS Source 等。出于同样的原因，这些 Source 为了性能也应该批处理事件。Source 写事件到 Channel，且它们应该以合理大小的批量写出，以避免在这一节之前讨论的文件同步或同步问题。所以，即使对于能控制自己批量大小的 Source，它们被配置为使用合理大小的批量也是很重要的。