目录

1、什么是Flume

Apache Flume 是一个分布式，可靠的，可信赖的系统，用于从不同的sources高效的收集，聚合并且移动大量的日志数据到一个数据中心存储仓库中。
Flume用单跳消息传递来保证语义来为系统提供端到端的可靠性。为了实现这一点，某些新概念已被纳入其设计中，而某些其他现有概念已经被重新定义，重用或完全丢弃。

2、核心概念

1）、Event：Flume从起始点到最后目的地的数据单元，它是以字节形式并且带有可选的字符串头部 来表示的。
2）、Flow：events从起始点到最终目的地的移动就被叫做data flow，或者simple flow。
3）、Client：操作起始点的events并且把他们分发到Flume Agent的一个接口实现。客户端通常在它们消费数据的应用程序的进程空间中操作。
4）、Agent：一个独立的进程，托管Flume组件，如源，通道和sink，可以接收，存储并且将事件转发到下一跳的目标。
5）、Source：通过特殊的机制可以消费发送给它的events的接口实现，例如，一个Avro source 就是一个source，它可以用来接收在flow中来自clients 或者其他agents 的Avro event。当source接收到event后，source就会将他交给一个或者多个channel。

Source有两种类型

a、PollableSourceRunner（KafkaSource）

b、EventDrivenSourceRunner

（AvroSource,ExecSource,NetcatSource,SpoolDirectorySource ）

6）、Channel：events的临时存储，其中events通过在Agent内操作的source传递到通道，一个放在channel中的event会保留在channel中直到sink 从中移除进行下一步的传输。Channel在确保Flow的耐久性方面起重要作用。

Channel作用：

a、在3、 Flow Pipeline所说的使用类似于生产者消费者数据交换模型来使source与sink的解耦。

b、一个source复制多个channel，冗余实现了容错的特性；

c、缓冲event。防止source突然增加event而导致sink不能及时消费；有一个agent失败以后可以缓冲发送过来的event防止数据的丢失。

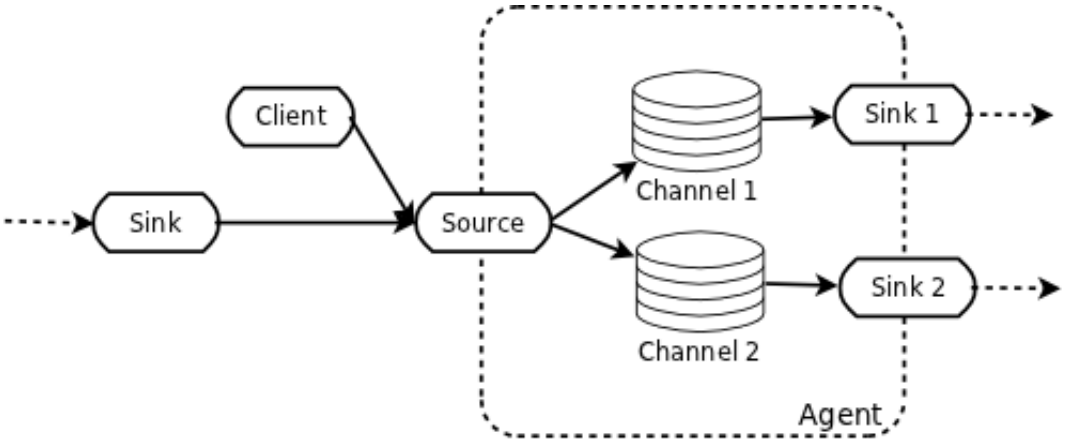
d、put，take实现了事务，sink从channel中取event记为step1开启一个事务，source向channel中放数据记为step2，只有step2返回成功信号以后step1才能进行下一步，实现event传递的保证。

7）、Sink：可以将event从channel中移除并且在flow中传递到下一个的Agent或者event 的终点的接口实现。

3、Flow Pipeline

在Flume NG中一个Flow开始于Client，client将event传递到它的下一个目的地。这个目的地是一个agent。更准确的说，destination是agent内的一个source操作。source接收event后会将它分发给一个或者多个channel,,在同一个agent内的channel接收的event会被一个或者多个sink操作消耗。如果一个agent是一个普通的agent，它将会把event转发到下一跳的目的地也就是另一个agent。如果sink是一个终端sink，它将event转发到最终目的地。Channel允许使用类似于生产者消费者数据交换模型来使source与sink的解耦。这就使得source和sink有不同的性能和运行时特性，并且能够有效地使用系统可用的物理资源。

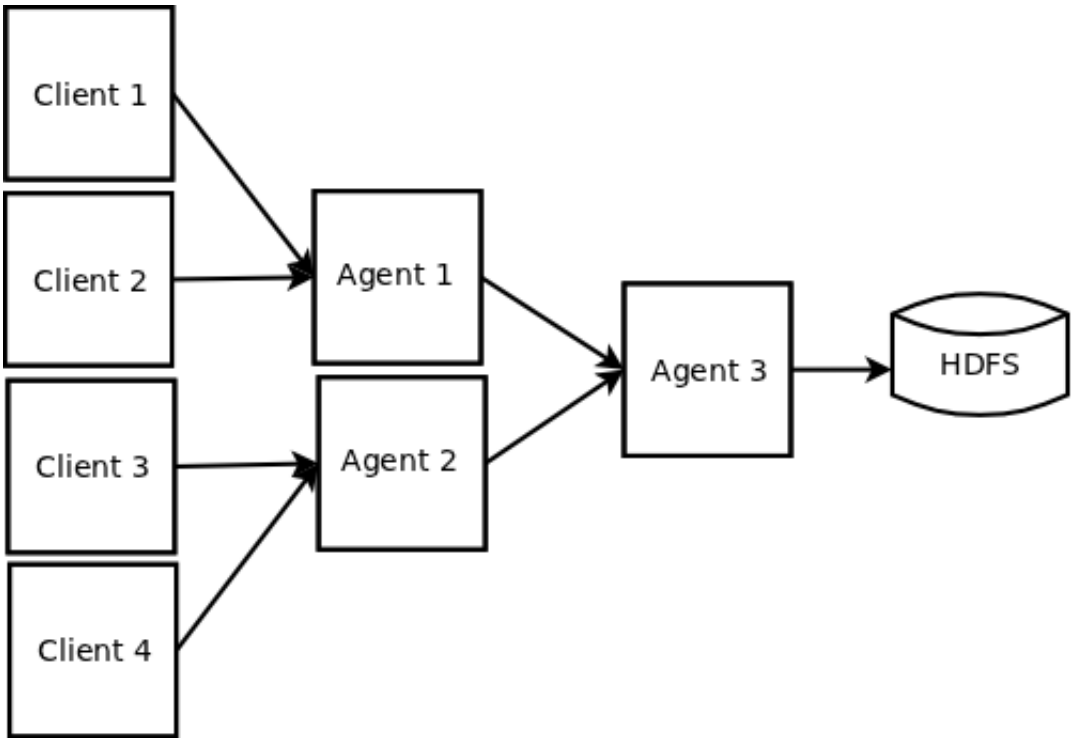
图一 如下图所示在一个flow pipline中各个组件怎么交互



图一 ：一个Flow中的逻辑组件示意图。 箭头表示event在系统上行进的方向。 这也说明了通过使一个source将event写入多个channel，flow如何扇出。

通过配置一个source分发event到多个channel，flows可以扇出到多个目的地。图一说明了这个问题，即agent中的source将事件写出到了两个channel–channel1 和channel2。

相反，可以通过使多个source在相同agent中操作来向相同channel写入来使flow汇聚。 汇聚flo的物理布局的示例在下面的图2中示出。



图二 一个简单的flow汇聚示意图

4、高可靠与容错处理

Flume NG 使用基于channel的事务来保证消息传递的可靠性。当消息从一个agent传递到另一个agent的时候，会开启两个事务，一个在agent上转发event，另一个在agent上会接收event。为了发送agent提交它的事务，它必须接收到从接收agent的成功信号。接收的agent只会返回一个成功的信号当它自己的事务首先正确的执行。这确保了在flow所做的跳之间的语义传递的保证（guaranteed delivery semantics ）。下面的图3示出了说明在两个交互agent内操作的事务的相对范围和持续时间的序列图。

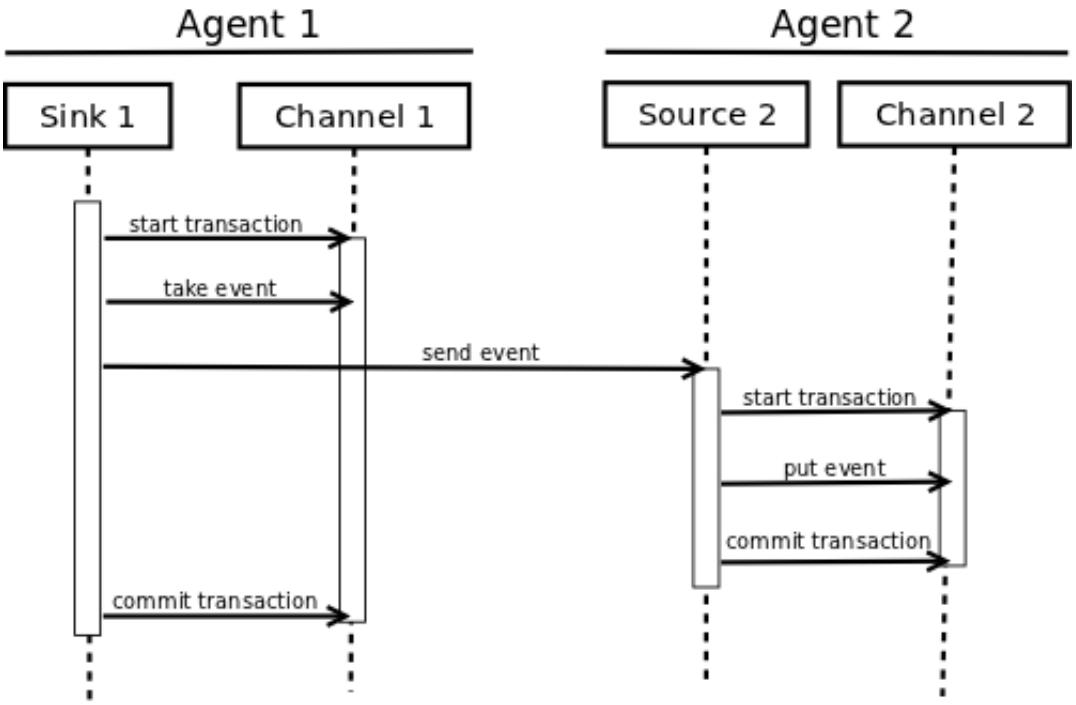


图 3: agent之间的事务交换

这个机制也形成了错误处理的基本原理。当通过许多不同代理的flow在flow的任何分支上遇到通信故障时，受影响的event开始在flow的最后未受影响的agent处被缓冲。如果错误未能及时解决，这可能会导致最后未受影响的agent的失败，接下来会强制它前面的agent开始缓冲event。最终，如果当客户端将事件发送到其第一跳目的地时发生故障，则故障将被报告回客户端，然后客户端可以允许生成事件的应用采取适当的动作。

另一方面，如果故障在第一跳的agent失败之前解决，则下游各种agent中的缓冲事件将开始向其目的地排出，最终，flow将恢复到其原始特性吞吐量水平，下图图4说明了在客户端和中央存储器之间的两个中间agent的flow经历瞬时故障的情况。故障发生在agent 2 和 存储器之间，到最后event在agent2 上缓存，一但故障解决，缓冲的events就会向中心存储器排出并且flow就会恢复原始吞吐量水平。

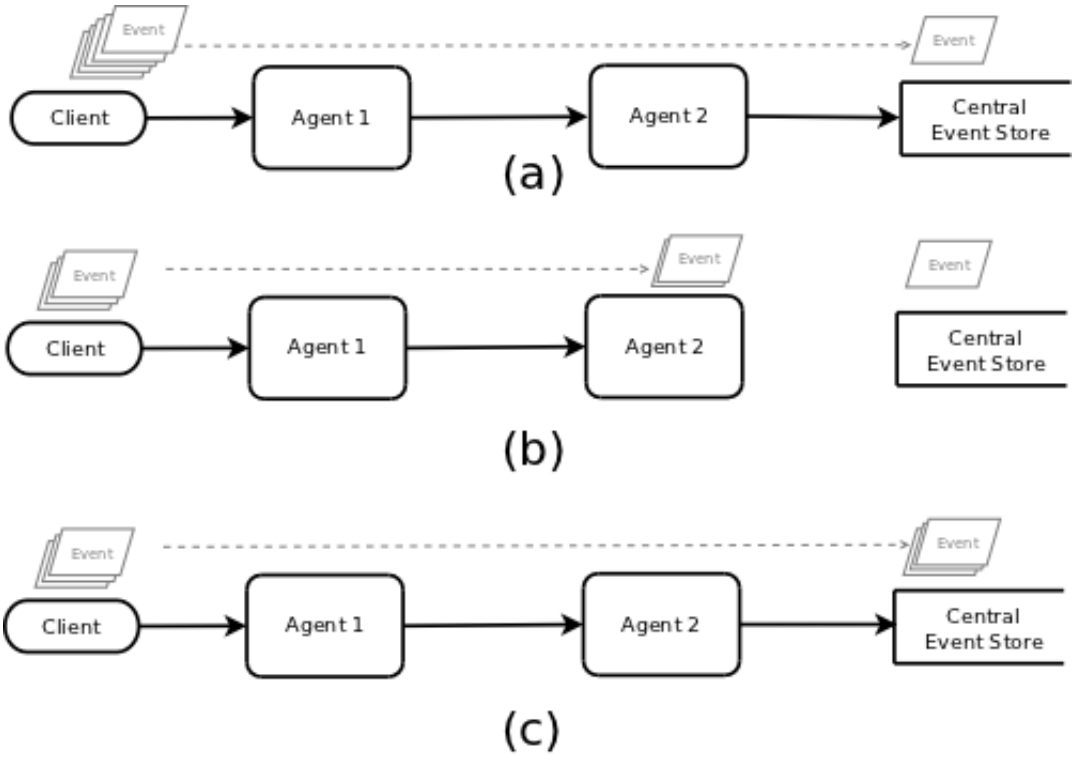


图4：flow的错误处理。
在(a)中flow是正常的，event卡伊从client流向中央存储器，在(b)中Agent2和最终存储器通讯失败导致event被缓存在Agent2中，在(c)中，引起错误的原因解决，flow修复在agent2中缓存的event流向存储器。

5、拓扑设计需要考虑的

Flume非常灵活，允许大范围的可能部署场景。 如果您计划在大型生产部署中使用Flume，那么谨慎的做法是花一些时间考虑如何根据Flume拓扑来表达您的问题。 本节包括几个注意事项。

5.1、Flume是否适合你的问题

如果你需要将文本日志数据录入Hadoop / HDFS，那么Flume是完全适合你的问题。 对于其他用例，请参阅以下指南：
Flume设计用于在相对稳定，可能复杂的拓扑中传输和摄取定期生成的事件数据。“event data”这个定义是非常广泛的。对于Flume，一个event只是一个通用的blob的字节。一个event的大小是有限制的，例如，不能大于一台机器的内存或者磁盘大小，但是实际上Flume的event可以是从文本图像文件的日志条目。事件的关键属性是它们以连续的流传输方式生成，如果你的Flume数据不是定期产生的，flume仍然会工作。但是可能会overkill你的情况。
Flume喜欢相对稳定的拓扑结构。你的拓扑结构不必是不可变的因为Flume可以处理拓扑的变化而不丢失数据也可以容忍周期性的由于故障切换或者添加设备的重新配置。他可能不会正常工作如果每天改变拓扑结构，因为重新配置需要花费一些思考和开销。

5.2、Flume中Flow的可靠性

Flume flow的可靠性由多个因素决定，通过调整这些因素，你可以您可以使Flume实现多种可靠性选项。

a)、使用哪种类型的channel

Flume由可持久的channel（持久化到磁盘）和非持久化得channel，持久化channel使用基于磁盘的存储，并且存储在此类通道中的数据将在整个机器重新启动或非磁盘相关故障中保留。

b)、Channel是否为工作负载有效配置

Channel在Flume中扮演着缓存的角色在不同跳之间。这些buffers有着固定容量，一旦容量满了就会在flow上的前一个点创建black pressure，如果pressure传递到flow的源，Flume就会变为不可访问并且可能丢失数据。

c)、是否使用冗余的拓扑结构

Flume让你在冗余拓扑中的复制flow。 这可以提供非常容易的容错source，并且能够克服磁盘或机器故障。

在Flume拓扑中最好方式考虑可靠性的方法是考虑各种失败的常见以及他们的输出。磁盘故障了怎么办?机器故障了怎么办？最终的sink某刻时间挂了或者有back presssure怎么办？可能设计的空间是巨大的，但你需要问的基本问题只是一个少数。

5.3、Flume拓扑设计

设计Flume拓扑的第一步是枚举数据的所有源和目标（终端sink）。 这些将定义拓扑的边缘点。 下一个考虑是是否引入中间聚合层或事件路由。 如果要从大量源收集数据，则汇总数据可以有所帮助，以便简化在终端接收器处的摄取。 聚合层还可以通过充当缓冲器来平滑来自source的突发或在sink处的不可用性。 如果您在不同位置之间路由数据，您可能还希望在不同点分流：这创建子拓扑本身可以包括聚合点。

5.4、调整Flume部署大小

一旦你知道你的拓扑将是什么样子，下一个问题是需要多少硬件和网络容量。这首先是量化您生成的数据量。这不总是一个简单的任务！大多数数据流是突发的（例如，由于昼夜模式）并且可能不可预测。一个很好的出发点是要想想在拓扑中的每一层的最大吞吐量，无论是在events per second and bytes per second。一旦知道给定层的所需吞吐量，就可以计算该层所需的节点数量的下限。要确定可达到的吞吐量，最好使用合成或采样事件数据在您的硬件上尝试Flume。一般来说，基于磁盘的通道应该获得10的MB / s，基于内存的通道应该获得100的MB / s或更多。性能将有很大变化，但是根据硬件和操作环境。

调整聚合吞吐量可为您提供每个层所需的节点数的下限。有几个原因需要有额外的节点，例如增加的冗余和更好的吸收突发负载的能力。

5.5、处理agent故障

如果Flume agent故障，则托管在该代理上的所有流都将中止。 一旦代理重新启动，则flow将恢复。 使用文件通道或其他稳定通道的flow将恢复处理其中断的事件。 如果无法在同一硬件上重新启动代理，则可以选择将数据库迁移到另一个硬件并设置一个新的Flume代理，

该代理可以恢复处理保存在数据库中的事件。 数据库HA特性可以用来将Flume agent移动到另一个主机。

6、配置

6.1、配置文件

Flume 配置是存储在一个本地配置文件中，它是依照java属性文件格式的文本文件。一个或者多个agent的配置可以放到同一个配置文件中。配置文件包括一个agent的的每个source，sink。channel的属性并且说明他们怎么连接起来形成data flow；

6.2、配置单个组件

flow中的每个组件都具有特定于类型和实例化的名称，类型和属性集。例如，AVro source需要一个hostname 以及一个端口号来接收数据。Memory channel 必须有最大队列长度，HDFS sink需要知道文件系统的URL，路径来创建文件，文件生成的频率等。一个组件的所有这些属性都需要在承载Flume agent的属性文件中设置。

6.3、把每一部分都链接起来

agent需要知道那个单独的组件加载了也要知道怎么连在一起构成flow。这是通过在agent中列出每个source，channel以及sink的名字来完成的。