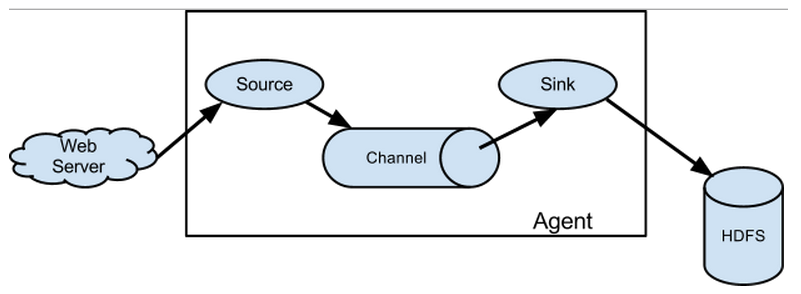
**Flume**

## 一、概述

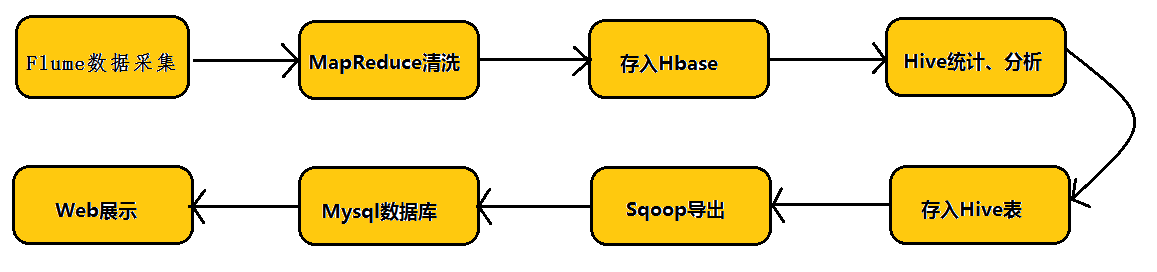
**1 Flume是Cloudera提供的一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统**，Flume支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。当前Flume有两个版本Flume 0.9X版本的统称Flume-og，Flume1.X版本的统称Flume-ng。由于Flume-ng经过重大重构，与Flume-og有很大不同，使用时请注意区分。

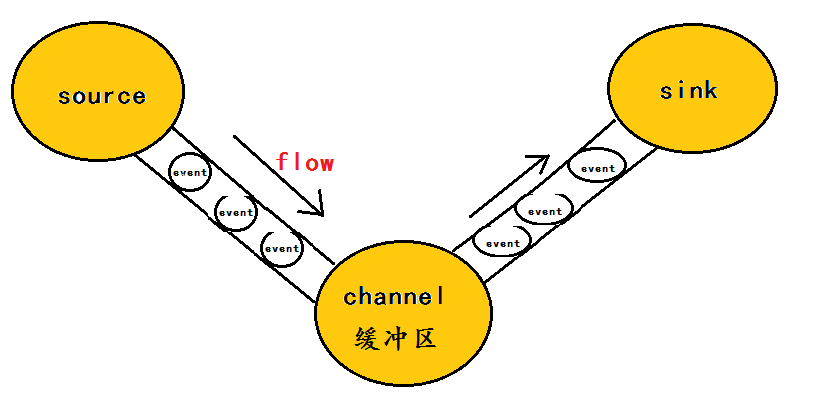
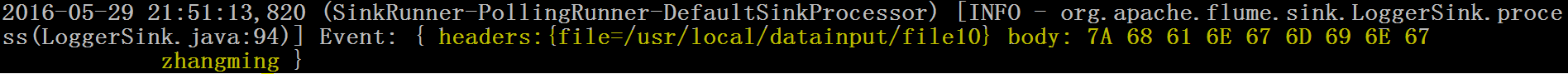
**2 flume是分布式的日志收集系统**，它将各个服务器中的数据收集起来并送到指定的地方去，比如说送到图中的HDFS，简单来说flume就是收集日志的。



**3 Flume的架构**

从Hadoop的业务开发流程图中可以看出，在大数据的业务处理过程中，对于数据的采集是十分重要的一步，也是不可避免的一步，从而引出我们本文的主角—Flume。本文将围绕Flume的架构、Flume的应用(日志采集)进行详细的介绍。



**4 Event的概念**   
在这里有必要先介绍一下flume中event的相关概念：flume的核心是把数据从数据源(source)收集过来，在将收集到的数据送到指定的目的地(sink)。为了保证输送的过程一定成功，在送到目的地(sink)之前，会先缓存数据(channel),待数据真正到达目的地(sink)后，flume在删除自己缓存的数据。   
在整个数据的传输的过程中，流动的是event，即事务保证是在event级别进行的。那么什么是event呢？—–event将传输的数据进行封装，是flume传输数据的基本单位，如果是文本文件，通常是一行记录，event也是事务的基本单位。event从source，流向channel，再到sink，本身为一个字节数组，并可携带headers(头信息)信息。event代表着一个数据的最小完整单元，从外部数据源来，向外部的目的地去。   
为了方便大家理解，给出一张event的数据流向图：   
  
一个完整的event包括：event headers、event body、event信息(即文本文件中的单行记录)，如下所以：   
  
其中event信息就是flume收集到的日记记录。

**5 flume架构介绍**

flume之所以这么神奇，是源于它自身的一个设计，这个设计就是agent，agent本身是一个java进程，运行在日志收集节点—所谓日志收集节点就是服务器节点。

agent里面包含3个核心的组件：source—->channel—–>sink,类似生产者、仓库、消费者的架构。

**source：**source组件是专门用来收集数据的，可以处理各种类型、各种格式的日志数据,包括avro、thrift、exec、jms、spooling directory、netcat、sequence generator、syslog、http、legacy、自定义。

**channel：**source组件把数据收集来以后，临时存放在channel中，即channel组件在agent中是专门用来存放临时数据的——对采集到的数据进行简单的缓存，可以存放在memory、jdbc、file等等。

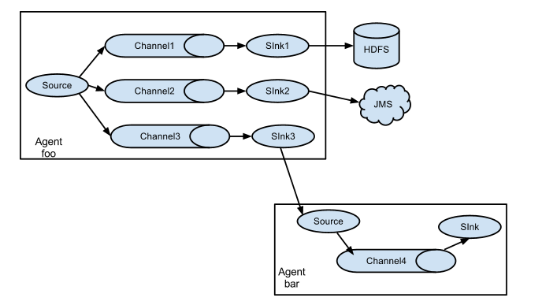
**sink：**sink组件是用于把数据发送到目的地的组件，目的地包括hdfs、logger、avro、thrift、ipc、file、null、hbase、solr、自定义。

**6、flume的运行机制**

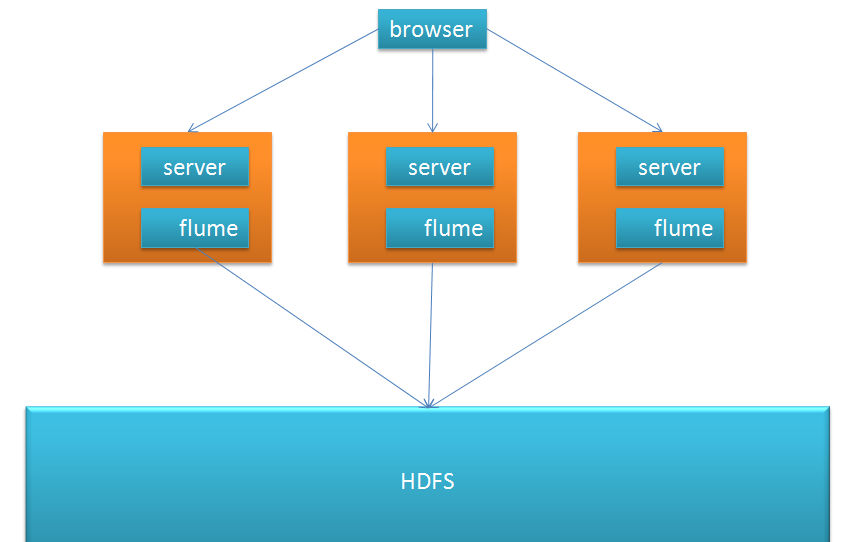
flume的核心就是一个agent，这个agent对外有两个进行交互的地方，一个是接受数据的输入——source，一个是数据的输出sink，sink负责将数据发送到外部指定的目的地。source接收到数据之后，将数据发送给channel，chanel作为一个数据缓冲区会临时存放这些数据，随后sink会将channel中的数据发送到指定的地方—-例如HDFS等，注意：只有在sink将channel中的数据成功发送出去之后，channel才会将临时数据进行删除，这种机制保证了数据传输的可靠性与安全性。

**7、flume的广义用法**

flume之所以这么神奇—-其原因也在于flume可以支持多级flume的agent，即flume可以前后相继，例如sink可以将数据写到下一个agent的source中，这样的话就可以连成串了，可以整体处理了。flume还支持扇入(fan-in)、扇出(fan-out)。所谓扇入就是source可以接受多个输入，所谓扇出就是sink可以将数据输出多个目的地destination中。



**8** **[flume安装配置-采集日志到hadoop存储](http://blog.csdn.net/liangjianyong007/article/details/52964488)**



flume其实就是一个日志采集agent，在每台应用服务器安装一个flume agent，然后事实采集日志到HDFS集群环境存储，以便后续使用hive或者pig等大数据分析日志，然后可转存到mysql供运维查询或分析用户行为等。

**9 fume与hadoop集群规划**

hadoop集群安装参照：[http://blog.csdn.net/liangjianyong007/article/details/52893234](http://blog.csdn.net/liangjianyong007/article/details/52893234" \t "_blank)

集群规划：Hive只在一个节点（hadoop3）上安装即可

**主机名     IP              安装的软件       运行的进程**

    hadoop1 192.168.31.10   jdk、hadoop                  NameNode、DFSZKFailoverController、

    hadoop2 192.168.31.20   jdk、hadoop  、hive、mysql         NameNode、DFSZKFailoverController、hive、mysql

    hadoop3 192.168.31.30   jdk、hadoop  、flume              ResourceManager、fulme

    hadoop4 192.168.31.40   jdk、hadoop、zookeeper        DataNode、NodeManager、JournalNode、QuorumPeerMain

    hadoop5 192.168.31.50   jdk、hadoop、zookeeper        DataNode、NodeManager、JournalNode、QuorumPeerMain

    hadoop6 192.168.31.60   jdk、hadoop、zookeeper            DataNode、NodeManager、JournalNode、QuorumPeerMain

**10 flume下载安装**

1）. 下载flume：[http://archive.apache.org/dist/flume/](http://archive.apache.org/dist/flume/" \t "_blank)

2）. 安装

tar -zxvf apache-flume-1.5.0-bin.tar.gz -C /usr/cloud/flume

3）.配置环境变量

vim /etc/profile

export JAVA\_HOME=/usr/cloud/java/jdk1.6.0\_24

export HADOOP\_HOME=/usr/cloud/hadoop/hadoop-2.2.0

export HBASE\_HOME=/usr/cloud/hbase/hbase-0.96.2

export FLUME\_HOME=/usr/cloud/flume/apache-flume-1.5.0

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin:$HBASE\_HOME/bin:$FLUME\_HOME/bin

source /etc/profile

**11 要把hadoop集群配置好的hdfs-site.xml和core-site.xml 放到hbase/conf下**

       #执行hdfs的配置

scp $HADOOP\_HOME/etc/hadoop/core-site.xml  $FLUME\_HOME/conf

scp $HADOOP\_HOME/etc/hadoop/hdfs-site.xml  $FLUME\_HOME/conf

**12 修改flume-env.sh**

sudo vim $FLUME\_HOME/conf/flume-env.sh

#设置JAVA\_HOME

JAVA\_HOME=/usr/cloud/java/jdk1.6.0\_24

**13 拷贝flume依赖的hadoopjar包到$FLUME\_HOME/lib：**

scp $HADOOP\_HOME/share/hadoop/common/hadoop-common-2.2.0.jar                  $FLUME\_HOME/lib/  
scp $HADOOP\_HOME/share/hadoop/common/lib/hadoop-auth-2.2.0.jar                     $FLUME\_HOME/lib/  
scp $HADOOP\_HOME/share/hadoop/common/lib/commons-configuration-1.6.jar       $FLUME\_HOME/lib/

**14 配置flume source，channel，sink ,在conf下新建文件flume.conf**

vim /FLUME\_HOME/conf/flume.conf

#定义agent名， source、channel、sink的名称

a4.sources = r1

a4.channels = c1

a4.sinks = k1

#具体定义source

a4.sources.r1.type = spooldir

a4.sources.r1.spoolDir = /usr/cloud/flume/log

#具体定义channel

a4.channels.c1.type = memory

a4.channels.c1.capacity = 10000

a4.channels.c1.transactionCapacity = 100

#定义拦截器，为消息添加时间戳

a4.sources.r1.interceptors = i1

a4.sources.r1.interceptors.i1.type = org.apache.flume.interceptor.TimestampInterceptor$Builder

#具体定义sink

a4.sinks.k1.type = hdfs

a4.sinks.k1.hdfs.path = hdfs://ns1/flume/%Y%m%d

a4.sinks.k1.hdfs.filePrefix = events-

a4.sinks.k1.hdfs.fileType = DataStream

#不按照条数生成文件

a4.sinks.k1.hdfs.rollCount = 0

#HDFS上的文件达到128M时生成一个文件

a4.sinks.k1.hdfs.rollSize = 134217728

#HDFS上的文件达到60秒生成一个文件

a4.sinks.k1.hdfs.rollInterval = 60

#组装source、channel、sink

a4.sources.r1.channels = c1

a4.sinks.k1.channel = c1

**15 启动flume，保证hdfs已经启动**

#编写脚本 start-flume.sh

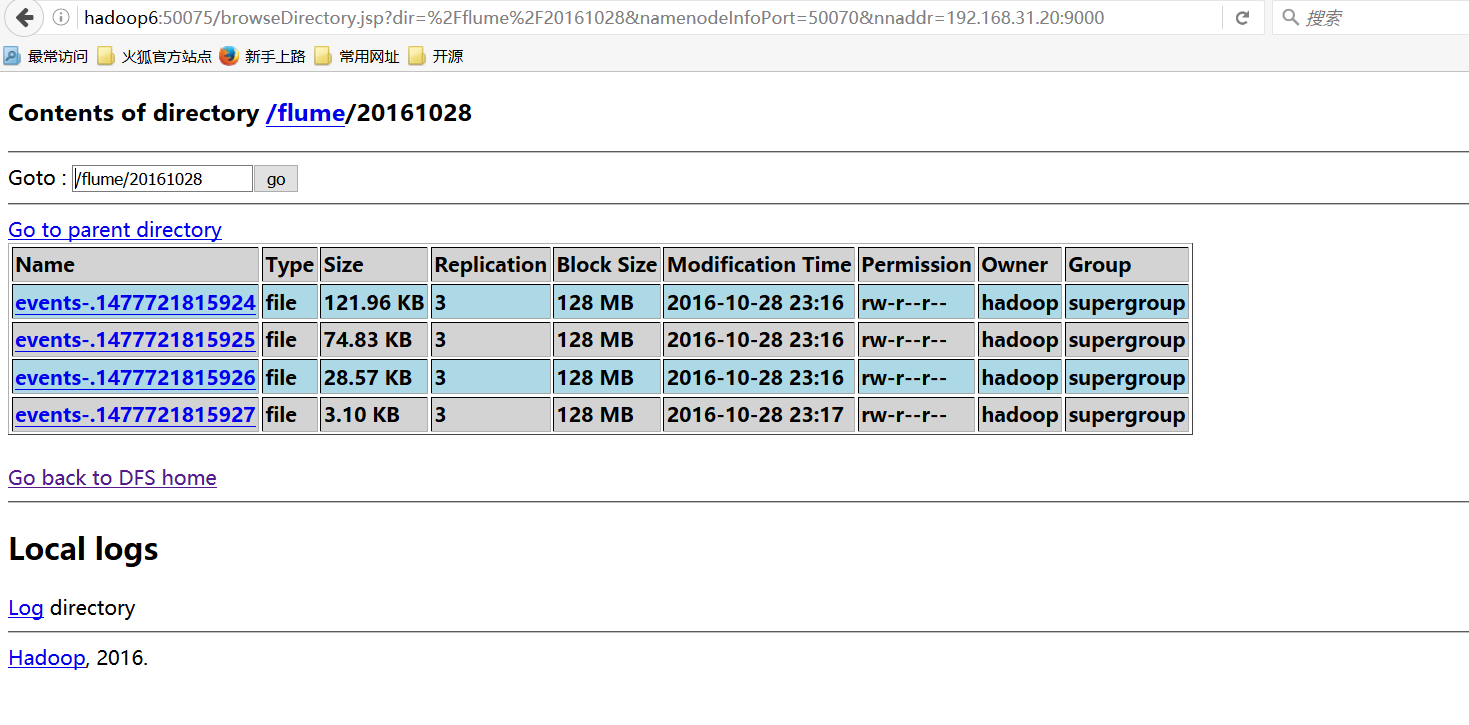
vim $FLUME\_HOME/bin/start-flume.sh

$FLUME\_HOME/bin/flume-ng agent -n a4 -c conf -f $FLUME\_HOME/conf/a4.conf -Dflume.root.logger=INFO,console

#启动flume

$FLUME\_HOME/bin/start-flume.sh

检查配置的文件是否上传到hdfs



/usr/cloud/flume/log在存储文件biz.log,内容：

ERROR [2016-10-27 14:44:02,482] com.alibaba.scm.common.monitor.log.ScmBizLogger:30 - |<>|traceId==>UUID:3ff0f75f-0ea8-466b-9c15-b96747b8d636<||>bizType==>3000<||>T==>107<||>SUCC==>true<||>bizName==>createInboundOrder\_begain<||>IN==>{"appointmentDate":"Oct 27, 2016 2:43:52 PM","sellId":3693710345,"sellName":"天猫国际012","storeCode":"STORE\_230095","customsInfoNo":"ASN00105244708","purchaseOrderNo":"PO160822222226","supplierId":300000000001402,"consignOrderNo":"CO160323143157012226","abroad":false,"demandDate":"Oct 27, 2016 2:43:52 PM","saleType":2,"shipType":0,"consignOrderItemDTOList":[{"scItemId":2100771461843,"itemId":2100770298741,"quantity":100,"gmtExpired":"Oct 10, 2016 12:00:00 AM","rejectDays":10,"managerType":3,"rowVersion":0}],"bizType":3000,"rowVersion":0}  <||>bizCode==>CO160323143157012226<||>CM==>InboundOrderClientImpl.createInboundOrder<||>OUT==>{"success":true,"isRetry":false}<||>

ERROR [2016-10-27 14:44:03,716] com.alibaba.scm.common.monitor.log.ScmBizLogger:30 - |<>|traceId==>UUID:eb51567f-f83a-4dfb-8d38-9b6b58d84021<||>bizType==>3000<||>T==>1359<||>SUCC==>true<||>bizName==>createInboundOrder\_end<||>IN==>{"appointmentDate":"Oct 27, 2016 2:43:52 PM","sellId":3693710345,"sellName":"天猫国际012","storeCode":"STORE\_230095","customsInfoNo":"ASN00105244708","purchaseOrderNo":"PO160822222226","supplierId":300000000001402,"consignOrderNo":"CO160323143157012226","abroad":false,"demandDate":"Oct 27, 2016 2:43:52 PM","saleType":2,"shipType":0,"consignOrderItemDTOList":[{"scItemId":2100771461843,"itemId":2100770298741,"quantity":100,"gmtExpired":"Oct 10, 2016 12:00:00 AM","rejectDays":10,"managerType":3,"rowVersion":0}],"bizType":3000,"rowVersion":0}  <||>bizCode==>CO160323143157012226<||>CM==>InboundOrderClientImpl.createInboundOrder<||>OUT==>{"success":true,"isRetry":false,"model":{"purchaseOrderNo":"PO160822222226","inboundNo":"IO16102714440300373001","consignOrderNo":"CO160323143157012226","storeCode":"STORE\_230095","supplierId":300000000001402,"sendQuantity":100,"preArrival":"Oct 27, 2016 2:43:52 PM","status":10,"demandDate":"Oct 27, 2016 2:43:52 PM","inboundItemDOList":[{"inboundNo":"IO16102714440300373001","inboundId":373001,"scItemId":2100771461843,"itemId":2100770298741,"sendQuantity":100,"status":1,"rejectDays":10,"managerType":3,"gmtExpired":"Oct 10, 2016 12:00:00 AM","storeCode":"STORE\_230095","purchaseOrderNo":"PO160822222226","bizType":3000,"attribute":"MXSDUEDATE:20161010","attributeMap":{"MXSDUEDATE":"20161010"}}],"id":373001,"bizType":3000}}<||>

**16 四种配置方式**

**（1）配置1：监听某一个端口将流经端口的数据捕获写入到日志**

#定义agent名， source、channel、sink的名称

a1.sources = r1

a1.channels = c1

a1.sinks = k1

#具体定义source

a1.sources.r1.type = netcat

a1.sources.r1.bind = localhost

a1.sources.r1.port = 8888

#具体定义channel

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

#具体定义sink

a1.sinks.k1.type = logger

#组装source、channel、sink

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

**（2）配置2：从命令行中得到数据并输出到日志文件**

#bin/flume-ng agent -n a2 -f /home/hadoop/a2.conf -c conf -Dflume.root.logger=INFO,console

#定义agent名， source、channel、sink的名称

a2.sources = r1

a2.channels = c1

a2.sinks = k1

#具体定义source

a2.sources.r1.type = exec

a2.sources.r1.command = tail -F /home/hadoop/a.log

#具体定义channel

a2.channels.c1.type = memory

a2.channels.c1.capacity = 1000

a2.channels.c1.transactionCapacity = 100

#具体定义sink

a2.sinks.k1.type = logger

#组装source、channel、sink

a2.sources.r1.channels = c1

a2.sinks.k1.channel = c1

**配置3：从目录中读取数据写入到日志文件**

#定义agent名， source、channel、sink的名称

a3.sources = r1

a3.channels = c1

a3.sinks = k1

#具体定义source

a3.sources.r1.type = spooldir

a3.sources.r1.spoolDir = /root/logs

#具体定义channel

a3.channels.c1.type = memory

a3.channels.c1.capacity = 1000

a3.channels.c1.transactionCapacity = 100

#具体定义sink

a3.sinks.k1.type = logger

#组装source、channel、sink

a3.sources.r1.channels = c1

a3.sinks.k1.channel = c1

**配置4：主要是从目录获取数据并将数据写入hdfs**

#定义agent名， source、channel、sink的名称

a4.sources = r1

a4.channels = c1

a4.sinks = k1

#具体定义source

a4.sources.r1.type = spooldir

a4.sources.r1.spoolDir = /home/hadoop/logs

#具体定义channel

a4.channels.c1.type = memory

a4.channels.c1.capacity = 10000

a4.channels.c1.transactionCapacity = 100

#定义拦截器，为消息添加时间戳

a4.sources.r1.interceptors = i1

a4.sources.r1.interceptors.i1.type = org.apache.flume.interceptor.TimestampInterceptor$Builder

#具体定义sink

a4.sinks.k1.type = hdfs

a4.sinks.k1.hdfs.path = hdfs://ns1/flume/%Y%m%d

a4.sinks.k1.hdfs.filePrefix = events-

a4.sinks.k1.hdfs.fileType = DataStream

#不按照条数生成文件

a4.sinks.k1.hdfs.rollCount = 0

#HDFS上的文件达到128M时生成一个文件

a4.sinks.k1.hdfs.rollSize = 134217728

#HDFS上的文件达到60秒生成一个文件

a4.sinks.k1.hdfs.rollInterval = 60

#组装source、channel、sink

a4.sources.r1.channels = c1

a4.sinks.k1.channel = c1

常用配置模式一

扫描指定文件

agent.sources.s1.type=exec

agent.sources.s1.command=tail -F /Users/it-od-m/Downloads/abc.log

agent.sources.s1.channels=c1

agent.channels.c1.type=memory

agent.channels.c1.capacity=10000

agent.channels.c1.transactionCapacity=100

#设置Kafka接收器

agent.sinks.k1.type= org.apache.flume.sink.kafka.KafkaSink

#设置Kafka的broker地址和端口号

agent.sinks.k1.brokerList=127.0.0.1:9092

#设置Kafka的Topic

agent.sinks.k1.topic=testKJ1

#设置序列化方式

agent.sinks.k1.serializer.class=kafka.serializer.StringEncoder

agent.sinks.k1.channel=c1

常用配置模式二

Agent名称定义为agent.

Source:可以理解为输入端，定义名称为s1

channel：传输频道，定义为c1，设置为内存模式

sinks：可以理解为输出端，定义为sk1,

agent.sources = s1

agent.channels = c1

agent.sinks = sk1

#设置Source的内省为netcat 端口为5678，使用的channel为c1

agent.sources.s1.type = netcat

agent.sources.s1.bind = localhost

agent.sources.s1.port = 3456

agent.sources.s1.channels = c1

#设置Sink为logger模式，使用的channel为c1

agent.sinks.sk1.type = logger

agent.sinks.sk1.channel = c1

#设置channel信息

agent.channels.c1.type = memory #内存模式

agent.channels.c1.capacity = 1000

agent.channels.c1.transactionCapacity = 100 #传输参数设置。

常用配置模式三

扫描目录新增文件

agent.sources = s1

agent.channels = c1

agent.sinks = sk1

#设置spooldir

agent.sources.s1.type = spooldir

agent.sources.s1.spoolDir = /Users/it-od-m/logs

agent.sources.s1.fileHeader = true

agent.sources.s1.channels = c1

agent.sinks.sk1.type = logger

agent.sinks.sk1.channel = c1

#In Memory !!!

agent.channels.c1.type = memory

agent.channels.c1.capacity = 10004

agent.channels.c1.transactionCapacity = 100

**17 flume特点**

(1) 可靠性

当节点出现故障时，日志能够被传送到其他节点上而不会丢失。Flume提供了三种级别的可靠性保障，从强到弱依次分别为：end-to-end（收到数据agent首先将event写到磁盘上，当数据传送成功后，再删除；如果数据发送失败，可以重新发送。），Store on failure（这也是scribe采用的策略，当数据接收方crash时，将数据写到本地，待恢复后，继续发送），Best effort（数据发送到接收方后，不会进行确认）。

(2) 可扩展性

Flume采用了三层架构，分别为agent，collector和storage，每一层均可以水平扩展。其中，所有agent和collector由master统一管理，这使得系统容易监控和维护，且master允许有多个（使用ZooKeeper进行管理和负载均衡），这就避免了单点故障问题。

(3) 可管理性

所有agent和colletor由master统一管理，这使得系统便于维护。多master情况，Flume利用ZooKeeper和gossip，保证动态配置数据的一致性。用户可以在master上查看各个数据源或者数据流执行情况，且可以对各个数据源配置和动态加载。Flume提供了web 和shell script command两种形式对数据流进行管理。

(4) 功能可扩展性

用户可以根据需要添加自己的agent，collector或者storage。此外，Flume自带了很多组件，包括各种agent（file， syslog等），collector和storage（file，HDFS等）。

**17 kafka和flume**

（1）kafka和flume都是日志系统。kafka是分布式消息中间件，自带存储，提供push和pull存取数据功能。flume分为agent（数据采集器）,collector（数据简单处理和写入）,storage（存储器）三部分，每一部分都是可以定制的。比如agent采用RPC（Thrift-RPC）、text（文件）等，storage指定用hdfs做。

（2）kafka做日志缓存应该是更为合适的，但是 flume的数据采集部分做的很好，可以定制很多数据源，减少开发量。

所以比较流行flume+kafka模式，如果为了利用flume写hdfs的能力，也可以采用kafka+flume的方式。

采集层 主要可以使用Flume, Kafka两种技术。

Flume：Flume 是管道流方式，提供了很多的默认实现，让用户通过参数部署，及扩展API.

Kafka：Kafka是一个可持久化的分布式的消息队列。

Kafka 是一个非常通用的系统。你可以有许多生产者和很多的消费者共享多个主题Topics。相比之下,Flume是一个专用工具被设计为旨在往HDFS,HBase发送数据。它对HDFS有特殊的优化，并且集成了Hadoop的安全特性。**所以，Cloudera 建议如果数据被多个系统消费的话，使用kafka；如果数据被设计给Hadoop使用，使用Flume。**

正如你们所知Flume内置很多的source和sink组件。然而，Kafka明显有一个更小的生产消费者生态系统，并且Kafka的社区支持不好。希望将来这种情况会得到改善，但是目前：使用Kafka意味着你准备好了编写你自己的生产者和消费者代码。如果已经存在的Flume Sources和Sinks满足你的需求，并且你更喜欢不需要任何开发的系统，请使用Flume。

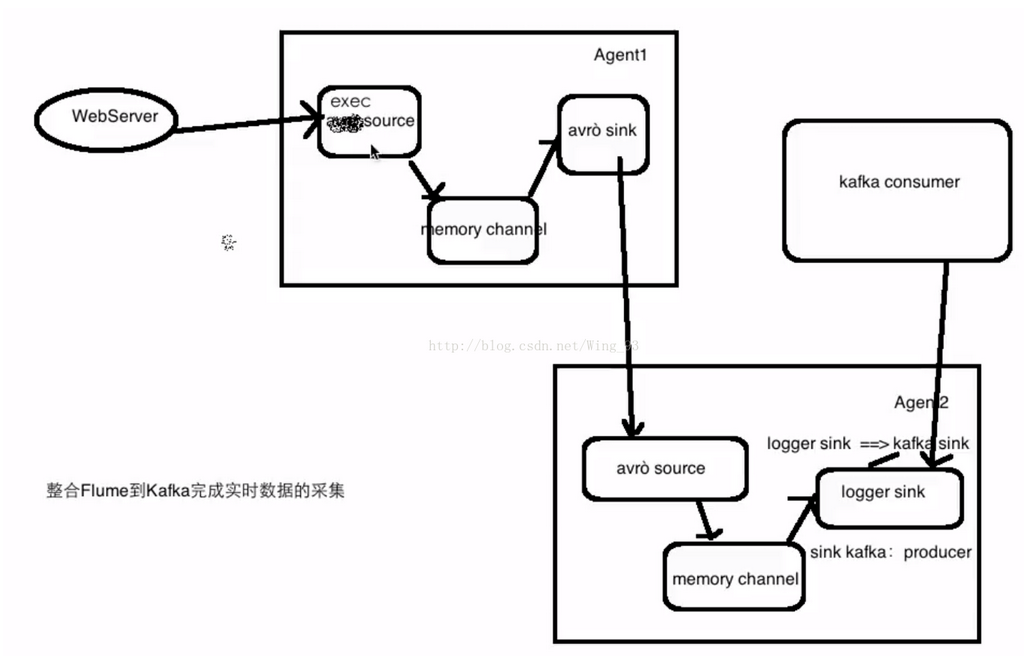
Flume可以使用拦截器实时处理数据。这些对数据屏蔽或者过量是很有用的。Kafka需要外部的流处理系统才能做到。

Kafka和Flume都是可靠的系统,通过适当的配置能保证零数据丢失。然而，Flume不支持副本事件。于是，如果Flume代理的一个节点奔溃了，即使使用了可靠的文件管道方式，你也将丢失这些事件直到你恢复这些磁盘。如果你需要一个高可靠行的管道，那么使用Kafka是个更好的选择。

Flume和Kafka可以很好地结合起来使用。如果你的设计需要从Kafka到Hadoop的流数据，使用Flume代理并配置Kafka的Source读取数据也是可行的：你没有必要实现自己的消费者。你可以直接利用Flume与HDFS及HBase的结合的所有好处。你可以使用Cloudera Manager对消费者的监控，并且你甚至可以添加拦截器进行一些流处理。

Flume和Kafka可以结合起来使用。通常会使用Flume + Kafka的方式。其实如果为了利用Flume已有的写HDFS功能，也可以使用Kafka + Flume的方式。

**Flume收集WEB服务器的数据信息，然后再对Flume进行串联，即对Flume进行扩展，选中一台flume，把信息sink到Kafka去，此时这台Flume的 sink就相当于是kafka的生产者，再启动一台kafka的消费者，直接对接到kafka的生产者，这样就完成了Flume到Kafka的实时数据采集。**



**Flume ：管道 ----个人认为比较适合有多个生产者场景，或者有写入Hbase、HDFS和kafka需求的场景。**

**Kafka ：消息队列-----由于Kafka是Pull模式，因此适合有多个消费者的场景。**

目前应用场景，一台日志转发机负责产生日志。后端需要通过Strom消费日志信息，建议可以设置成log-->Kafka->Strom.如果以后有写入Hbase或者HDFS的需求可以，在Kafka后面再接上Strom，或者在日志转发机上直接日志落地，由Flume去读取日志消息。

**17 基于Flume的美团日志收集系统(一)架构和设计**

美团的日志收集系统负责美团的所有业务日志的收集，并分别给Hadoop平台提供离线数据和Storm平台提供实时数据流。美团的日志收集系统基于Flume设计和搭建而成。《基于Flume的美团日志收集系统》将分两部分给读者呈现美团日志收集系统的架构设计和实战经验。

第一部分架构和设计，将主要着眼于日志收集系统整体的架构设计，以及为什么要做这样的设计。

第二部分改进和优化，将主要着眼于实际部署和使用过程中遇到的问题，对Flume做的功能修改和优化等。

**（1） 日志收集系统简介**

日志收集是大数据的基石。

许多公司的业务平台每天都会产生大量的日志数据。收集业务日志数据，供离线和在线的分析系统使用，正是日志收集系统的要做的事情。高可用性，高可靠性和可扩展性是日志收集系统所具有的基本特征。

目前常用的开源日志收集系统有Flume, Scribe等。Flume是Cloudera提供的一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统，目前已经是Apache的一个子项目。Scribe是Facebook开源的日志收集系统，它为日志的分布式收集，统一处理提供一个可扩展的，高容错的简单方案。

**（2） 用的开源日志收集系统对比**

下面将对常见的开源日志收集系统Flume和Scribe的各方面进行对比。对比中Flume将主要采用Apache下的Flume-NG为参考对象。同时，我们将常用的日志收集系统分为三层（Agent层，Collector层和Store层）来进行对比。

**（3）美团日志收集系统架构**

美团的日志收集系统负责美团的所有业务日志的收集，并分别给Hadoop平台提供离线数据和Storm平台提供实时数据流。美团的日志收集系统基于Flume设计和搭建而成。目前每天收集和处理约T级别的日志数据。

**（4）美团日志收集系统架构**

a. 整个系统分为三层：Agent层，Collector层和Store层。其中Agent层每个机器部署一个进程，负责对单机的日志收集工作；Collector层部署在中心服务器上，负责接收Agent层发送的日志，并且将日志根据路由规则写到相应的Store层中；Store层负责提供永久或者临时的日志存储服务，或者将日志流导向其它服务器。

b. Agent到Collector使用LoadBalance策略，将所有的日志均衡地发到所有的Collector上，达到负载均衡的目标，同时并处理单个Collector失效的问题。

c. Collector层的目标主要有三个：SinkHdfs, SinkKafka和SinkBypass。分别提供离线的数据到Hdfs，和提供实时的日志流到Kafka和Bypass。其中SinkHdfs又根据日志量的大小分为SinkHdfs\_b，SinkHdfs\_m和SinkHdfs\_s三个Sink，以提高写入到Hdfs的性能，具体见后面介绍。

d. 对于Store来说，Hdfs负责永久地存储所有日志；Kafka存储最新的7天日志，并给Storm系统提供实时日志流；Bypass负责给其它服务器和应用提供实时日志流。

**（5）美团日志收集系统架构**

a. 模块命名规则：所有的Source以src开头，所有的Channel以ch开头，所有的Sink以sink开头；

b. Channel统一使用美团开发的DualChannel，具体原因后面详述；对于过滤掉的日志使用NullChannel，具体原因后面详述；

c. 模块之间内部通信统一使用Avro接口；

**18 架构设计考虑**

下面将从可用性，可靠性，可扩展性和兼容性等方面，对上述的架构做细致的解析。

**4.1 可用性(availablity)**

对日志收集系统来说，可用性(availablity)指固定周期内系统无故障运行总时间。要想提高系统的可用性，就需要消除系统的单点，提高系统的冗余度。下面来看看美团的日志收集系统在可用性方面的考虑。

**4.1.1 Agent死掉**

Agent死掉分为两种情况：机器死机或者Agent进程死掉。

对于机器死机的情况来说，由于产生日志的进程也同样会死掉，所以不会再产生新的日志，不存在不提供服务的情况。

对于Agent进程死掉的情况来说，确实会降低系统的可用性。对此，我们有下面三种方式来提高系统的可用性。首先，所有的Agent在supervise的方式下启动，如果进程死掉会被系统立即重启，以提供服务。其次，对所有的Agent进行存活监控，发现Agent死掉立即报警。最后，对于非常重要的日志，建议应用直接将日志写磁盘，Agent使用spooldir的方式获得最新的日志。

**4.1.2 Collector死掉**

由于中心服务器提供的是对等的且无差别的服务，且Agent访问Collector做了LoadBalance和重试机制。所以当某个Collector无法提供服务时，Agent的重试策略会将数据发送到其它可用的Collector上面。所以整个服务不受影响。

**4.1.3 Hdfs正常停机**

我们在Collector的HdfsSink中提供了开关选项，可以控制Collector停止写Hdfs，并且将所有的events缓存到FileChannel的功能。

**4.1.4 Hdfs异常停机或不可访问**

假如Hdfs异常停机或不可访问，此时Collector无法写Hdfs。由于我们使用DualChannel，Collector可以将所收到的events缓存到FileChannel，保存在磁盘上，继续提供服务。当Hdfs恢复服务以后，再将FileChannel中缓存的events再发送到Hdfs上。这种机制类似于Scribe，可以提供较好的容错性。

**4.1.5 Collector变慢或者Agent/Collector网络变慢**

如果Collector处理速度变慢（比如机器load过高）或者Agent/Collector之间的网络变慢，可能导致Agent发送到Collector的速度变慢。同样的，对于此种情况，我们在Agent端使用DualChannel，Agent可以将收到的events缓存到FileChannel，保存在磁盘上，继续提供服务。当Collector恢复服务以后，再将FileChannel中缓存的events再发送给Collector。

**4.1.6 Hdfs变慢**

当Hadoop上的任务较多且有大量的读写操作时，Hdfs的读写数据往往变的很慢。由于每天，每周都有高峰使用期，所以这种情况非常普遍。

对于Hdfs变慢的问题，我们同样使用DualChannel来解决。当Hdfs写入较快时，所有的events只经过MemChannel传递数据，减少磁盘IO，获得较高性能。当Hdfs写入较慢时，所有的events只经过FileChannel传递数据，有一个较大的数据缓存空间。

**4.2 可靠性(reliability)**

对日志收集系统来说，可靠性(reliability)是指Flume在数据流的传输过程中，保证events的可靠传递。

对Flume来说，所有的events都被保存在Agent的Channel中，然后被发送到数据流中的下一个Agent或者最终的存储服务中。那么一个Agent的Channel中的events什么时候被删除呢？当且仅当它们被保存到下一个Agent的Channel中或者被保存到最终的存储服务中。这就是Flume提供数据流中点到点的可靠性保证的最基本的单跳消息传递语义。

那么Flume是如何做到上述最基本的消息传递语义呢？

首先，Agent间的事务交换。Flume使用事务的办法来保证event的可靠传递。Source和Sink分别被封装在事务中，这些事务由保存event的存储提供或者由Channel提供。这就保证了event在数据流的点对点传输中是可靠的。在多级数据流中，如下图，上一级的Sink和下一级的Source都被包含在事务中，保证数据可靠地从一个Channel到另一个Channel转移。

其次，数据流中 Channel的持久性。Flume中MemoryChannel是可能丢失数据的（当Agent死掉时），而FileChannel是持久性的，提供类似mysql的日志机制，保证数据不丢失。

**4.3 可扩展性(scalability)**

对日志收集系统来说，可扩展性(scalability)是指系统能够线性扩展。当日志量增大时，系统能够以简单的增加机器来达到线性扩容的目的。对于基于Flume的日志收集系统来说，需要在设计的每一层，都可以做到线性扩展地提供服务。下面将对每一层的可扩展性做相应的说明。

**4.3.1 Agent层**

对于Agent这一层来说，每个机器部署一个Agent，可以水平扩展，不受限制。一个方面，Agent收集日志的能力受限于机器的性能，正常情况下一个Agent可以为单机提供足够服务。另一方面，如果机器比较多，可能受限于后端Collector提供的服务，但Agent到Collector是有Load Balance机制，使得Collector可以线性扩展提高能力。

**4.3.2 Collector层**

对于Collector这一层，Agent到Collector是有Load Balance机制，并且Collector提供无差别服务，所以可以线性扩展。其性能主要受限于Store层提供的能力。

**4.3.3 Store层**

对于Store这一层来说，Hdfs和Kafka都是分布式系统，可以做到线性扩展。Bypass属于临时的应用，只对应于某一类日志，性能不是瓶颈。

**4.4 Channel的选择**

Flume1.4.0中，其官方提供常用的MemoryChannel和FileChannel供大家选择。其优劣如下：MemoryChannel: 所有的events被保存在内存中。优点是高吞吐。缺点是容量有限并且Agent死掉时会丢失内存中的数据。

FileChannel: 所有的events被保存在文件中。优点是容量较大且死掉时数据可恢复。缺点是速度较慢。

上述两种Channel，优缺点相反，分别有自己适合的场景。然而，对于大部分应用来说，我们希望Channel可以同提供高吞吐和大缓存。基于此，我们开发了DualChannel。

DualChannel：基于 MemoryChannel和 FileChannel开发。当堆积在Channel中的events数小于阈值时，所有的events被保存在MemoryChannel中，Sink从MemoryChannel中读取数据； 当堆积在Channel中的events数大于阈值时， 所有的events被自动存放在FileChannel中，Sink从FileChannel中读取数据。这样当系统正常运行时，我们可以使用MemoryChannel的高吞吐特性；当系统有异常时，我们可以利用FileChannel的大缓存的特性。

**4.5 和scribe兼容**

在设计之初，我们就要求每类日志都有一个category相对应，并且Flume的Agent提供AvroSource和ScribeSource两种服务。这将保持和之前的Scribe相对应，减少业务的更改成本。

**4.6 权限控制**

在目前的日志收集系统中，我们只使用最简单的权限控制。只有设定的category才可以进入到存储系统。所以目前的权限控制就是category过滤。

如果权限控制放在Agent端，优势是可以较好地控制垃圾数据在系统中流转。但劣势是配置修改麻烦，每增加一个日志就需要重启或者重载Agent的配置。

如果权限控制放在Collector端，优势是方便进行配置的修改和加载。劣势是部分没有注册的数据可能在Agent/Collector之间传输。

考虑到Agent/Collector之间的日志传输并非系统瓶颈，且目前日志收集属内部系统，安全问题属于次要问题，所以选择采用Collector端控制。

**4.7 提供实时流**

美团的部分业务，如实时推荐，反爬虫服务等服务，需要处理实时的数据流。因此我们希望Flume能够导出一份实时流给Kafka/Storm系统。一个非常重要的要求是实时数据流不应该受到其它Sink的速度影响，保证实时数据流的速度。这一点，我们是通过Collector中设置不同的Channel进行隔离，并且DualChannel的大容量保证了日志的处理不受Sink的影响。

**5 系统监控**

对于一个大型复杂系统来说，监控是必不可少的部分。设计合理的监控，可以对异常情况及时发现，只要有一部手机，就可以知道系统是否正常运作。对于美团的日志收集系统，我们建立了多维度的监控，防止未知的异常发生。

**5.1 发送速度，拥堵情况，写Hdfs速度**

通**过发送给zabbix的数据，我们可**以绘制出发送数量、拥堵情况和写Hdfs速度的图表，对于超预期的拥堵，我们会报警出来查找原因。

**5.2 flume写hfds状态的监控**

Flume写入Hdfs会先生成tmp文件，对于特别重要的日志，我们会每15分钟左右检查一下各个Collector是否都产生了tmp文件，对于没有正常产生tmp文件的Collector和日志我们需要检查是否有异常。这样可以及时发现Flume和日志的异常.

**5.3 日志大小异常监控**

对于重要的日志，我们会每个小时都监控日志大小周同比是否有较大波动，并给予提醒，这个报警有效的发现了异常的日志，且多次发现了应用方日志发送的异常，及时给予了对方反馈，帮助他们及早修复自身系统的异常。

通过上述的讲解，我们可以看到，基于Flume的美团日志收集系统已经是具备高可用性，高可靠性，可扩展等特性的分布式服务。