Flume

Flume介绍

定义

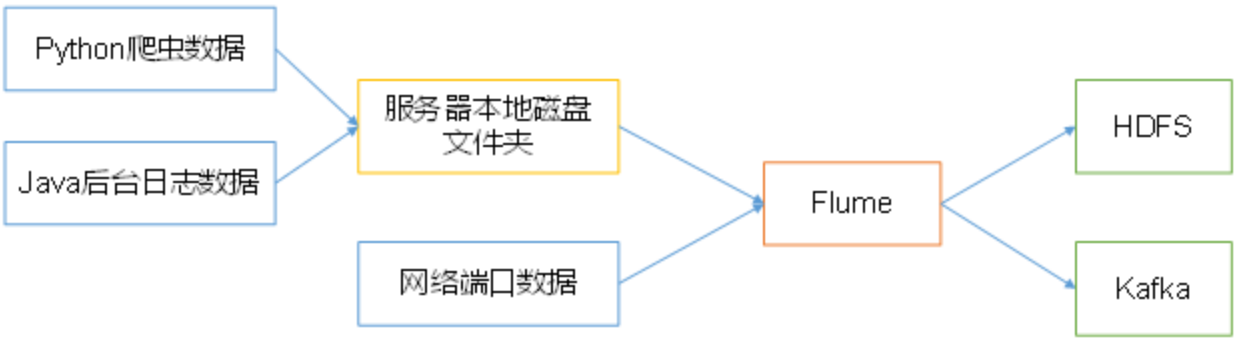
Flume是Cloudera提供的一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统。

Agent是Flume任务启动和运行的基本单元。

Flume基于流式架构，灵活简单。

Flume经常和HDFS、Hbase、Kakfa和Hive配合使用，用于数据的收集、存储和分析。

以下是Flume的简单应用架构。



优势

1、直接和大数据集群对接，如数据直接存储到HDFS和Hbase中

2、自动平衡数据传输速率。当Flume收集速度大于Flume数据输出的速度时，会自动平衡

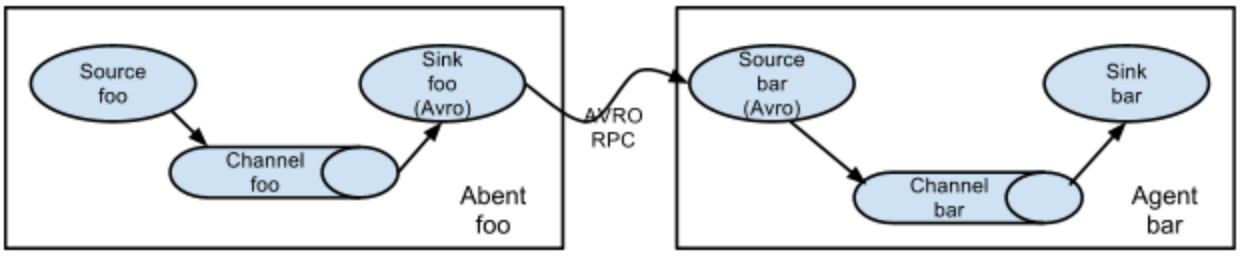
3、Flume易于水平扩展

4、丰富的数据对接类型，并支持自定义对接类型

应用场景

1、数据采集

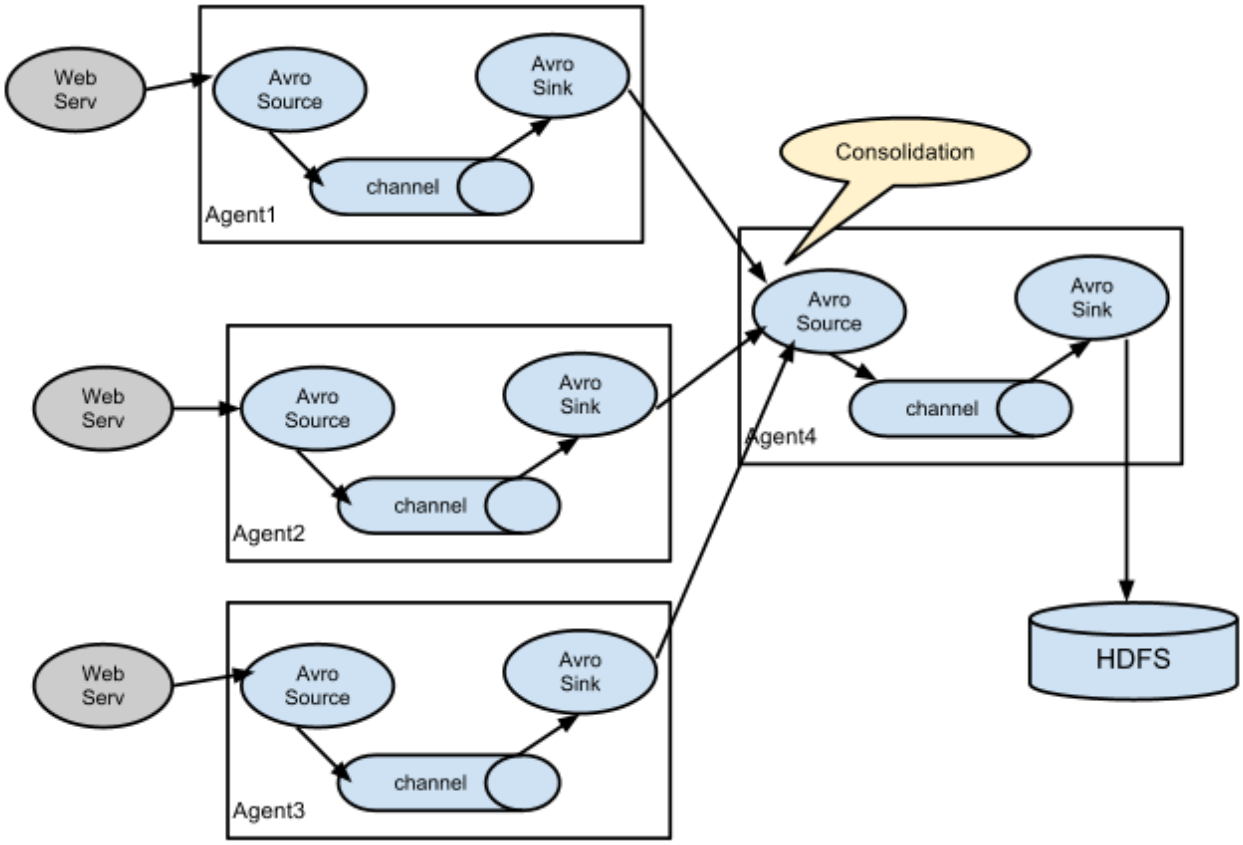
单一数据采集。



2、日志合并

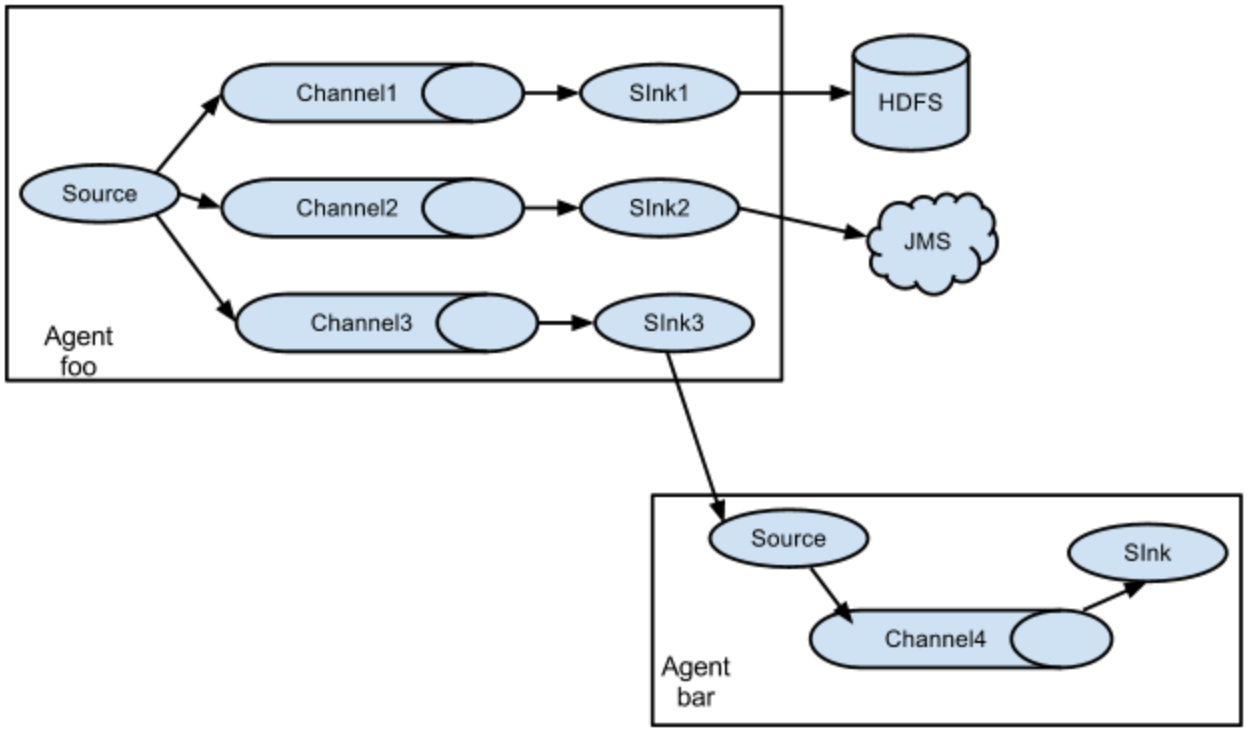
日志收集中的一种非常常见的情况是，大量的日志生成客户端将数据发送到连接到存储子系统的几个使用方代理。例如，从数百台Web服务器收集的

日志发送到许多写入HDFS群集的代理。



3、多路分发（Multiplexing Channel Selector）-或复制分发（Replicating Channel Selector ）

将事件流复用到一个或多个目的地。这是通过定义一种流多路复用器来实现的，该流多路复用器可以将事件复制或选择性地路由到一个或多个通道。



架构

Flume每个任务由一个Agent构成，Agent是一个JVM进程，每个Agent由三个组件构成，Source、Channel和Sink。其中Event是组件间数据传输的基本单位。

Source是数据采集的组件，事件采集后形成Event消息传给Channel。

Channel是中间数据缓冲区，将Event写入Sink。

Sink是数据输出的组件，将Event事件写入到外部系统，如Kafka、HDFS、Hbase或Flume。

source

Source是负责接收数据到Flume Agent的组件。Source组件可以处理各种类型、各种格式的日志数据：

- avro，从Avro客户端接收流数据
- thrift，从Thrift客户端接收流数据
- exec，从命令行接收数据，如tail –f /var/log/flume/flume.out
- jms，从消息队列获取数据，Kafka
- spooling directory，监视目录并从目录中获取数据，只能新增数据文件，不能对已有文件进行修改。TailDir
- netcat，从端口监听数据获取数据流，区分UDP和TCP
- sequence generator，生成序列化的数字，从0开始，，常用于测试

channel

Channel是位于Source和Sink之间的缓冲区。因此，Channel允许Source和Sink运作在不同的速率上。Channel是线程安全的，可以同时处理几个Source的写入操作和几个Sink的读取操作。

Flume自带两种Channel：Memory Channel和File Channel。

Memory Channel是内存中的队列。Memory Channel在不需要关心数据丢失的情景下适用。如果需要关心数据丢失，那么Memory Channel就不应该使用，因为程序死亡、机器宕机或者重启都会导致数据丢失。

File Channel将所有事件写到磁盘。因此在程序关闭或机器宕机的情况下不会丢失数据。

Sink

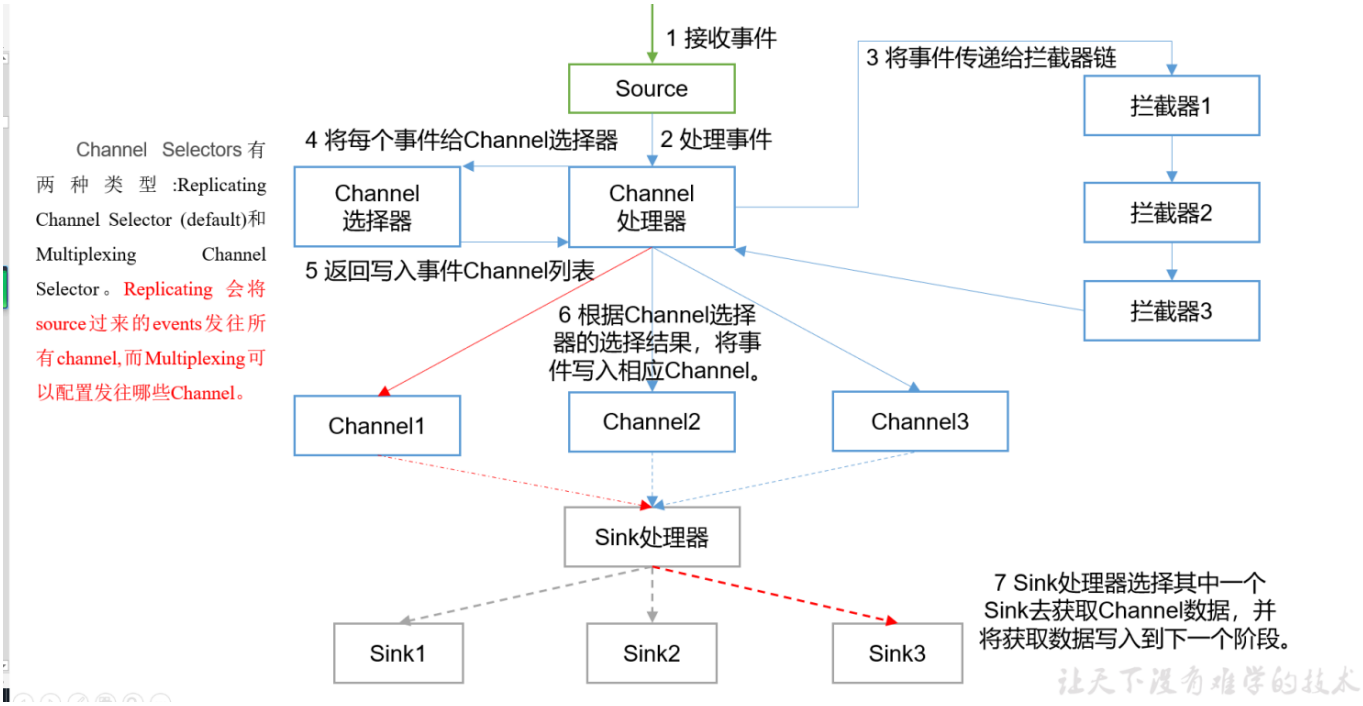
Sink不断地轮询Channel中的事件且批量地移除它们，并将这些事件批量写入到存储或索引系统、或者被发送到另一个Flume Agent。

Sink是完全事务性的。在从Channel批量删除数据之前，每个Sink用Channel启动一个事务。批量事件一旦成功写出到存储系统或下一个Flume Agent，Sink就利用Channel提交事务。事务一旦被提交，该Channel从自己的内部缓冲区删除事件。

Sink组件目的地如下：

- hdfs，将数据写入到HDFS文件系统
- hive，将数据写入Hive表或Hive表的分区中
- avro，写入avro服务，和Source的Avro类型相匹配
- file，将数据写入文件
- HBase，将数据写入Hbase
- solr，将数据写入Solr
- Kafka，将数据写入到Kafka

Agent处理流程



案例

以下操作在节点cdh-3.bigdata.com完成，相关配置文件在/app目录下。

Flume到HDFS

此操作演示使用CDH集成的flume，使用节点3演示。

读取目录到HDFS，操作目录是/app/flume_data。

```
tier1.sources  = source1  # source的名字
tier1.channels = ch-1     # channel的名字
tier1.sinks    = sink1    # sink的名字

tier1.sources.source1.type = spooldir   # source的类型
tier1.sources.source1.channels = ch-1   # source对应的channel类型
tier1.sources.source1.spoolDir = /app/flume_data   # source的目录
tier1.sources.source1.fileHeader = true   # 使用绝对路径作为header

tier1.channels.ch-1.type   = memory    # channel的类型

tier1.sinks.sink1.type         = hdfs   #  sink的输出
tier1.sinks.sink1.channel      = ch-1  # sink对应的channel
tier1.sinks.sink1.hdfs.path = /flume/events/%y-%m-%d/   #  输出路径
tier1.sinks.sink1.hdfs.filePrefix = events-             # 文件前缀
tier1.sinks.sink1.hdfs.round = true         #
tier1.sinks.sink1.hdfs.roundValue = 10  #
tier1.sinks.sink1.hdfs.roundUnit = minute   #
tier1.sinks.sink1.hdfs.useLocalTimeStamp = true  # 使用本地时间作为e
vent header

tier1.channels.channel1.capacity = 10000    # channel的事件数量
```

上述配置会产生很多小文件。小文件合并策略，使用CDH-2节点二的agent演示。

```
tier1.sources  = source1  # source的名字
tier1.channels = ch-1     # channel的名字
```

```
tier1.sinks    = sink1    # sink的名字

tier1.sources.source1.type = spooldir   # source的类型
tier1.sources.source1.channels = ch-1   # source对应的channel类型
tier1.sources.source1.spoolDir = /app/flume_data   # source的目录
tier1.sources.source1.fileHeader = true   # 使用绝对路径作为header

tier1.channels.ch-1.type   = memory    # channel的类型

tier1.sinks.sink1.type         = hdfs   #  sink的输出
tier1.sinks.sink1.channel      = ch-1  # sink对应的channel
tier1.sinks.sink1.hdfs.path = /flume/events_roll/%y-%m-%d/   # 输出路径
tier1.sinks.sink1.hdfs.filePrefix = events-             # 文件前缀
tier1.sinks.sink1.hdfs.rollSize=0
tier1.sinks.sink1.hdfs.rollCount=0
tier1.sinks.sink1.hdfs.useLocalTimeStamp = true  # 使用本地时间作为event header

tier1.channels.channel1.capacity = 10000    # channel的事件数量
```

小文件的其他处理方式呢?

Flume到Kafka

此操作演示使用命令行方式，在节点三cdh-3.bigdata.com上进行演示。

配置文件在：/app/flume_conf/flume2kafka.conf，操作也再次目录中。

```
pro.sources = s1
pro.channels = c1
pro.sinks = k1

pro.sources.s1.type = exec
pro.sources.s1.command = tail -F /app/flume_kafka.log

pro.channels.c1.type = memory
```

```
pro.channels.c1.capacity = 1000
pro.channels.c1.transactionCapacity = 100

pro.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
pro.sinks.k1.kafka.topic = flume
pro.sinks.k1.kafka.bootstrap.servers = cdh-1:9092,cdh-2:9092
pro.sinks.k1.kafka.flumeBatchSize = 20
pro.sinks.k1.kafka.producer.acks = 1
pro.sinks.k1.kafka.producer.linger.ms = 1
pro.sinks.k1.kafka.producer.compression.type = snappy

pro.sources.s1.channels = c1
pro.sinks.k1.channel = c1
```

启动命令

flume-ng agent -c /app/flume_conf/ -n pro -f flume2kafka.conf

需要注意的问题是，权限问题。

Flume的复制分发

此操作演示使用命令行方式，在节点三cdh-3.bigdata.com上进行演示。

配置文件在：/app/flume_conf/flume_multi_sink.conf，操作也再次目录中。

```
pro.sources = s1
pro.channels = c1 c2
pro.sinks = k1 k2

pro.sources.s1.type = exec
pro.sources.s1.command = tail -F /app/flume_kafka.log
pro.sources.s1.shell = /bin/bash -c

```

```
pro.channels.c1.type = memory
pro.channels.c1.capacity = 1000
pro.channels.c1.transactionCapacity = 100


pro.channels.c2.type = memory
pro.channels.c2.capacity = 1000
pro.channels.c2.transactionCapacity = 100

pro.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
pro.sinks.k1.kafka.topic = flume2
pro.sinks.k1.kafka.bootstrap.servers = cdh-1:9092,cdh-2:9092
pro.sinks.k1.kafka.flumeBatchSize = 20
pro.sinks.k1.kafka.producer.acks = 1
pro.sinks.k1.kafka.producer.linger.ms = 1
pro.sinks.k1.kafka.producer.compression.type = snappy

pro.sinks.k2.type=logger

pro.sources.s1.selector.type = replicating
pro.sources.s1.channels = c1 c2
pro.sinks.k1.channel = c1
pro.sinks.k2.channel = c2
```

启动命令

flume-ng agent -c /app/flume_conf/ -n pro -f flume_multi_sink.conf

多路分发的配置如下所示。

```
pro.sources = s1
pro.channels = c1 c2
pro.sinks = k1 k2

```

```
pro.sources.s1.type = exec
pro.sources.s1.command = tail -F /app/flume_kafka.log
pro.sources.s1.shell = /bin/bash -c

pro.channels.c1.type = memory
pro.channels.c1.capacity = 1000
pro.channels.c1.transactionCapacity = 100


pro.channels.c2.type = memory
pro.channels.c2.capacity = 1000
pro.channels.c2.transactionCapacity = 100

pro.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
pro.sinks.k1.kafka.topic = flume2
pro.sinks.k1.kafka.bootstrap.servers = cdh-1:9092,cdh-2:9092
pro.sinks.k1.kafka.flumeBatchSize = 20
pro.sinks.k1.kafka.producer.acks = 1
pro.sinks.k1.kafka.producer.linger.ms = 1
pro.sinks.k1.kafka.producer.compression.type = snappy

pro.sinks.k2.type=logger

################################  多路分发


pro.sources.s1.selector.type = multiplexing
pro.sources.s1.selector.header = state
pro.sources.s1.selector.mapping.CZ = c1
pro.sources.s1.selector.mapping.US = c2
# pro.sources.s1.selector.default = c4

pro.sourcess1.s1.channels = c1 c2
pro.sinks.k1.channel = c1
pro.sinks.k2.channel = c2
```