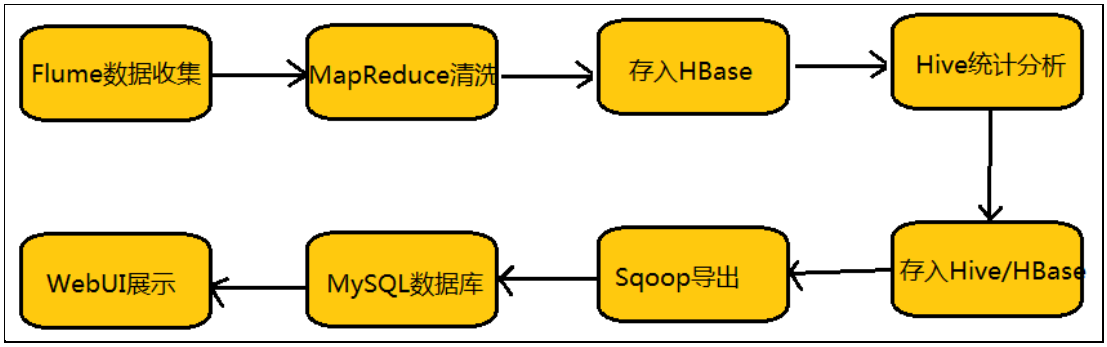
Flume 数据采集组件

目录

1、数据收集工具/系统产生背景

Hadoop 业务的整体开发流程：



任何完整的大数据平台，一般都会包括以下的基本处理过程：

数据采集
数据 ETL
数据存储
数据计算/分析
数据展现

其中，数据采集是所有数据系统必不可少的，随着大数据越来越被重视，数据采集的挑战也变的尤为突出。这其中包括：

数据源多种多样
数据量大，变化快
如何保证数据采集的可靠性的性能
如何避免重复数据
如何保证数据的质量

我们今天就来看看当前可用的一些数据采集的产品，重点关注一些它们是如何做到高可靠，高性能和高扩展。

总结：
数据的来源大体上包括：

1、业务数据
2、爬虫爬取的网络公开数据
3、购买数据
4、自行采集手机的日志数据

2、专业的数据收集工具

2.1、Chukwa

Apache Chukwa 是 Apache 旗下另一个开源的数据收集平台，它远没有其他几个有名。Chukwa 基于 Hadoop 的 HDFS 和 MapReduce 来构建（显而易见，它用 Java 来实现），提供扩展性和可靠性。Chukwa 同时提供对数据的展示，分析和监视。很奇怪的是它的上一次 Github 的更新事 7 年前。可见该项目应该已经不活跃了。
官网：http://chukwa.apache.org/

2.2、Scribe

Scribe 是 Facebook 开源的日志收集系统，在 Facebook 内部已经得到的应用。它能够从各种日志源上收集日志，存储到一个中央存储系统（可以是 NFS，HDFS，或者其他分布式文件系统等）上，以便于进行集中统计分析处理。
官网：https://www.scribesoft.com/

2.3、Fluentd

Fluentd 是另一个开源的数据收集框架。Fluentd 使用 C/Ruby 开发，使用 JSON 文件来统一日志数据。它的可插拔架构，支持各种不同种类和格式的数据源和数据输出。最后它也同时提供了高可靠和很好的扩展性。
官网：https://www.fluentd.org/

2.4、Logstash

Logstash 是著名的开源数据栈 ELK（ElasticSearch，Logstash，Kibana）中的那个 L。几乎在大部分的情况下 ELK 作为一个栈是被同时使用的。所有当你的数据系统使用 ElasticSearch 的情况下，Logstash 是首选。Logstash 用 JRuby 开发，所以运行时依赖 JVM。
官网：https://www.elastic.co/cn/products/logstash

2.5、Apache Flume

Flume 是 Apache 旗下，开源，高可靠，高扩展，容易管理，支持客户扩展的数据采集系统。Flume 使用 JRuby 来构建，所以依赖 Java 运行环境。Flume 最初是由 Cloudera 的工程师设计用于合并日志数据的系统，后来逐渐发展用于处理流数据事件。
官网：http://flume.apache.org/

2.6、DataX

DataX 是阿里巴巴集团内被广泛使用的离线数据同步工具/平台，实现包括 MySQL、SQL Server、Oracle、PostgreSQL、HDFS、Hive、HBase、OTS、ODPS 等各种异构数据源之间高效的数据同步功能。
可以适当关注阿里，腾讯，百度的技术体系

3、Flume 概述

3.1、Flume 概念

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.
Flume 是一个分布式、可靠、高可用的海量日志聚合系统，支持在系统中定制各类数据发送方，用于收集数据，同时，Flume 提供对数据的简单处理，并写到各种数据接收方的能力。

1、 Apache Flume 是一个分布式、可靠、和高可用的海量日志采集、聚合和传输的系统，和 Sqoop 同属于数据采集系统组件，但是 Sqoop 用来采集关系型数据库数据，而 Flume 用来采集流动型数据。

2、 Flume 名字来源于原始的近乎实时的日志数据采集工具，现在被广泛用于任何流事件数据的采集，它支持从很多数据源聚合数据到 HDFS。

3、 一般的采集需求，通过对 flume 的简单配置即可实现。Flume 针对特殊场景也具备良好的自定义扩展能力，因此，flume 可以适用于大部分的日常数据采集场景

4、 Flume 最初由 Cloudera 开发，在 2011 年贡献给了 Apache 基金会，2012 年变成了 Apache 的顶级项目。Flume OG（Original Generation）是 Flume 最初版本，后升级换代成 Flume NG（Next/New Generation）

5、 Flume 的优势：可横向扩展、延展性、可靠性

3.2、Flume 版本介绍

Flume 在 0.9.x and 1.x 之间有较大的架构调整：
1.x 版本之后的改称 Flume NG
0.9.x 版本称为 Flume OG，最后一个版本是 0.94，之后是由 Apache 进行了重构

N 和 O 的意思就是 new 和 old 的意思！

System Requirements ¶

1. Java Runtime Environment - Java 1.8 or later
2. Memory - Sufficient memory for configurations used by sources, channels or sinks
3. Disk Space - Sufficient disk space for configurations used by channels or sinks
4. Directory Permissions - Read/Write permissions for directories used by agent

官网文档：http://flume.apache.org/FlumeUserGuide.html

3.3、Flume 数据源和输出方式

Flume 提供了从 console(控制台)、RPC(Thrift-RPC)、text(文件)、tail(UNIX tail)、syslog(syslog 日志系统，支持 TCP 和 UDP 等 2 种模式)，exec(命令执行)等数据源上收集数据的能力，在我们的系统中目前使用 exec 方式进行日志采集。

Flume 的数据接受方，可以是 console(控制台)、text(文件)、dfs(HDFS 文件)、RPC(Thrift-RPC)和 syslogTCP(TCP syslog 日志系统)等。最常用的是 Kafka

4、Flume 体系结构/核心组件

4.1、概述

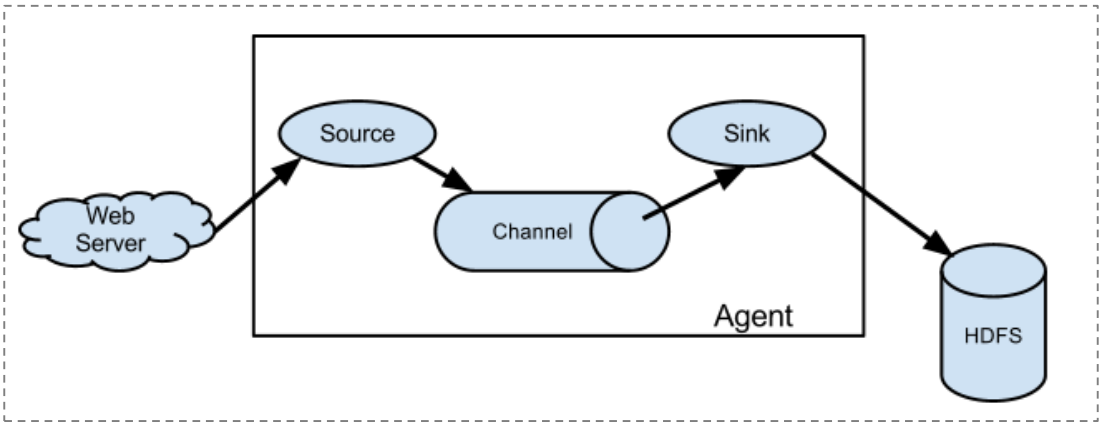
Flume 的数据流由事件(Event)贯穿始终。事件是 Flume 的基本数据单位，它携带日志数据(字节数组形式)并且携带有头信息，这些 Event 由 Agent 外部的 Source 生成，当 Source 捕获事件后会进行特定的格式化，然后 Source 会把事件推入(单个或多个)Channel 中。你可以把 Channel 看作是一个缓冲区，它将保存事件直到 Sink 处理完该事件。Sink 负责持久化日志或者把事件推向另一个 Source。

Flume 以 agent 为最小的独立运行单位。

一个 agent 就是一个 JVM。

单 agent 由 Source、Sink 和 Channel 三大组件构成。

如下图：



组件	功能
Agent	使用 JVM 运行 Flume。每台机器运行一个 agent，但是可以在一个 agent 中包含多个 sources 和 sinks。
Client	生产数据，运行在一个独立的线程。
Source	从 Client 收集数据，传递给 Channel。
Sink	从 Channel 收集数据，运行在一个独立线程。
Channel	连接 sources 和 sinks，这个有点像一个队列。
Events	可以是日志记录、avro 对象等。

4.2、Flume 三大核心组件

Event

Event 是 Flume 数据传输的基本单元。

Flume 以事件的形式将数据从源头传送到最终的目的地。

Event 由可选的 header 和载有数据的一个 byte array 构成。

载有的数据度 flume 是不透明的。

Header 是容纳了 key-value 字符串对的无序集合，key 在集合内是唯一的。

Header 可以在上下文路由中使用扩展

Client

Client 是一个将原始 log 包装成 events 并且发送他们到一个或多个 agent 的实体

目的是从数据源系统中解耦 Flume

在 Flume 的拓扑结构中不是必须的。

Client 实例

flume log4j Appender

可以使用 Client SDK（org.apache.flume.api)定制特定的 Client

Agent

一个 Agent 包含 source，channel，sink 和其他组件。

它利用这些组件将 events 从一个节点传输到另一个节点或最终目的地

agent 是 flume 流的基础部分。
flume 为这些组件提供了配置，声明周期管理，监控支持。

Agent 之 Source

Source 负责接收 event 或通过特殊机制产生 event，并将 events 批量的放到一个或多个
包含 event 驱动和轮询两种类型。
不同类型的 Source
与系统集成的 Source：Syslog，Netcat，监测目录池
自动生成事件的 Source：Exec
用于 Agent 和 Agent 之间通信的 IPC source：avro，thrift
source 必须至少和一个 channel 关联

Agent 之 Channel

Channel 位于 Source 和 Sink 之间，用于缓存进来的 event
当 sink 成功的将 event 发送到下一个的 channel 或最终目的 event 从 channel 删除
不同的 channel 提供的持久化水平也是不一样的
Memory Channel：volatile（不稳定的）
File Channel：基于 WAL（预写式日志 Write-Ahead Logging）实现
JDBC Channel：基于嵌入式 database 实现
channel 支持事务，提供较弱的顺序保证
可以和任何数量的 source 和 sink 工作

Agent 之 Sink

Sink 负责将 event 传输到吓一跳或最终目的地，成功后将 event 从 channel 移除
不同类型的 sink
存储 event 到最终目的地终端 sink，比如 HDFS，HBase
自动消耗的 sink 比如 null sink
用于 agent 间通信的 IPC：sink：Avro
必须作用于一个确切的 channel

Iterator

作用于 Source，按照预设的顺序在必要地方装饰和过滤 events

Channel Selector

允许 Source 基于预设的标准，从所有 channel 中，选择一个或者多个 channel

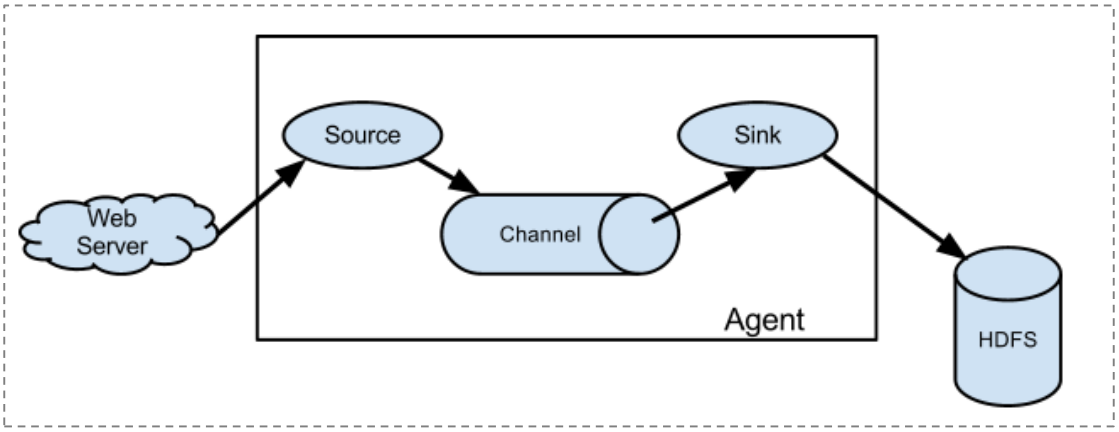
Sink Processor

多个 sink 可以构成一个 sink group
sink processor 可以通过组中所有 sink 实现负载均衡
也可以在一个 sink 失败时转移到另一个

Sources	Channels	Sinks
• Avro Source • Thrift Source • Exec Source • JMS Source • Spooling Directory Source • Twitter 1% firehose Source • Kafka Source • NetCat Source • Sequence Generator Source • Syslog Sources • Syslog TCP Source • Multiport Syslog TCP Source • Syslog UDP Source • HTTP Source • Stress Source • Legacy Sources • Thrift Legacy Source • Custom Source • Scribe Source	• Memory Channel • JDBC Channel • Kafka Channel • File Channel • Spillable Memory Channel • Pseudo Transaction Channel	• HDFS Sink • Hive Sink • Logger Sink • Avro Sink • Thrift Sink • IRC Sink • File Roll Sink • Null Sink • HBaseSink • AsyncHBaseSink • MorphlineSolrSink • ElasticSearchSink • Kite Dataset Sink • Kafka Sink

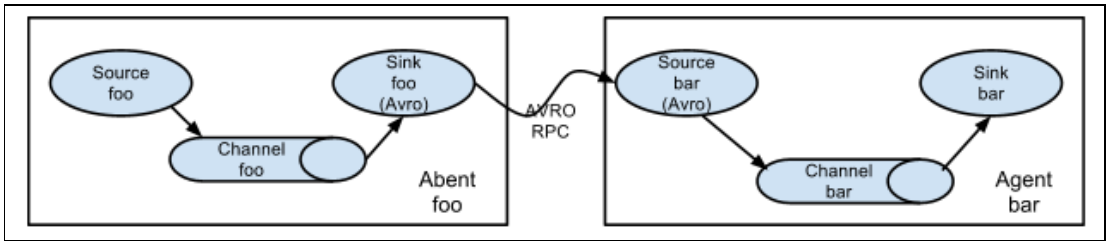
4.3、Flume 经典部署方案

4.3.1、单 Agent 采集数据



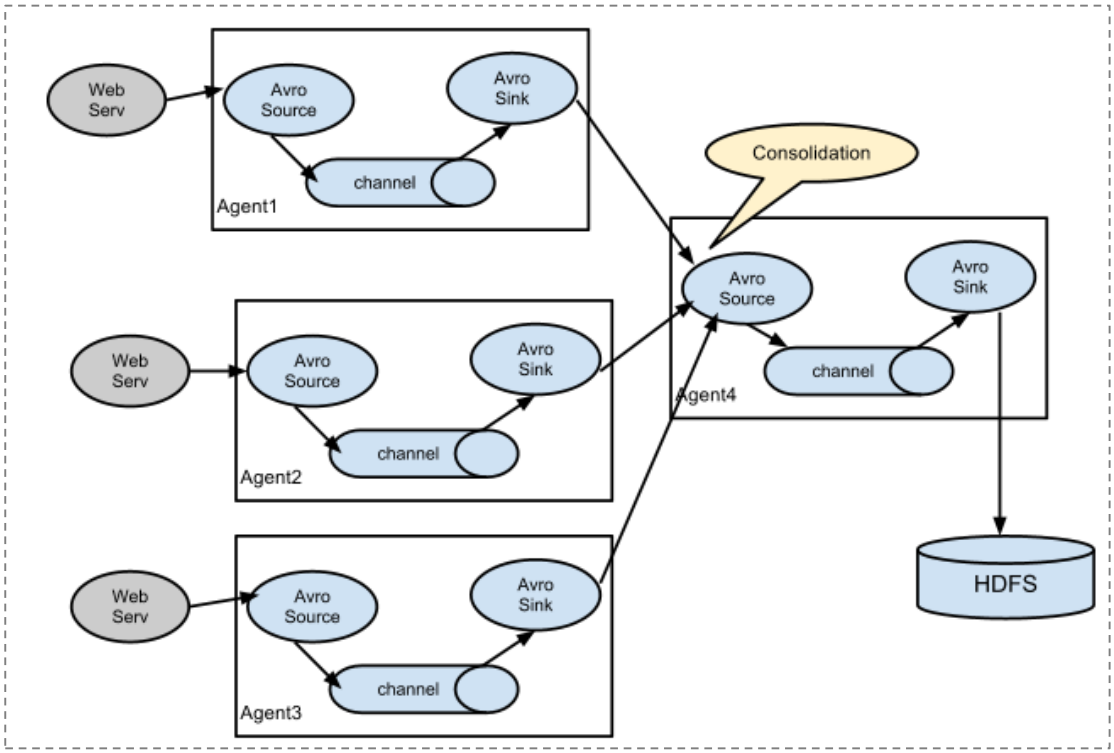
由一个 agent 负责把从 web server 中收集数据到 HDFS

4.3.2、多 Agent 串联



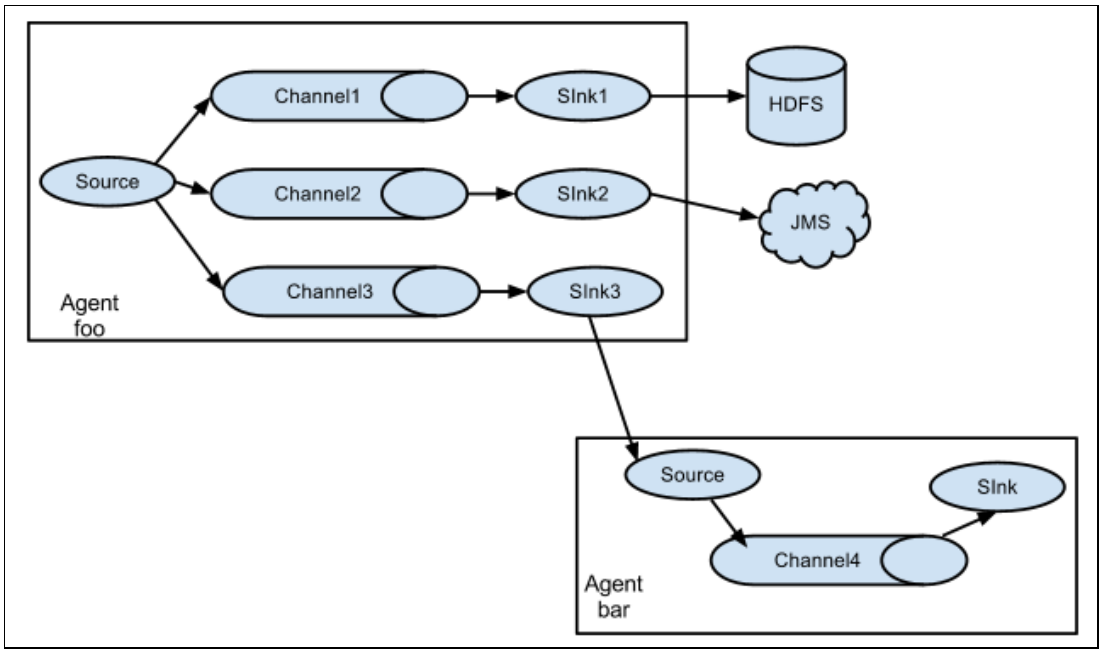
在收集数据的过程中，可以让多个 agent 串联起来，形成一条 event 数据线，进行传输，但是注意的是：相邻两个 agent 的前一个 agent 的 sink 类型要和后一个 agent 的 source 类型一致。

4.3.3、多 Agent 合并串联



多个 agent 串联，并联成一个复杂的 topo 图形收集架构。反映了 flume 的部署灵活。并且针对关键节点，还可以进行高可用配置

4.3.4、多路复用



一份数据流，可以被复制成多份数据流，交给多个不同组件进行处理。一般用于一边永久存储一边进行计算。

5、Flume 实战案例

5.1、安装部署 Flume

1、Flume 的安装非常简单，只需要解压即可，当然，前提是已有 Hadoop 环境上传安装包到数据源所在节点上
然后解压 tar -zxvf apache-flume-1.8.0-bin.tar.gz
然后进入 flume 的目录，修改 conf 下的 flume-env.sh，在里面配置 JAVA_HOME

2、根据数据采集的需求配置采集方案，描述在配置文件中
(文件名可任意自定义：对后缀名没要求，但是配置文件中的配置形式使用一行一个 key-value 的格式，也就是流行的 properties 配置文件格式，具体配置参见官网)

3、指定采集方案配置文件，在相应的节点上启动 flume agent

先用一个最简单的例子来测试一下程序环境是否正常

1、在$FLUME_HOME/agentconf 目录下创建一个数据采集方案，该方案就是从一个网络端口收集数据，也就是创一个任意命名的配置文件如下：netcat-logger.properties
文件内容如下：

```
# 定义这个 agent 中各个组件的名字
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# 描述和配置 source 组件：r1
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# 描述和配置 sink 组件：k1
a1.sinks.k1.type = logger

# 描述和配置 channel 组件，此处使用是内存缓存的方式
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# 描述和配置 source  channel  sink 之间的连接关系
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

2、启动 agent 去采集数据：

在$FLUME_HOME 下执行如下命令：

```
bin/flume-ng agent -c conf -f agentconf/netcat-logger.properties -n a1 -Dflume.root.logger=INFO,console
```

-c conf	指定 flume 自身的配置文件所在目录
-f conf/netcat-logger.perproties	指定我们所描述的采集方案
-n a1	指定我们这个 agent 的名字

3、测试

先要往 agent 的 source 所监听的端口上发送数据，让 agent 有数据可采

例如在本机节点，使用 **telnet localhost 44444** 命令就可以

如果这个命令的执行过程中发现抛出异常说：command not found

那么请使用：**sudo yum -y install telnet** 这个命令进行 telnet 的安装

输入两行数据：

hello huangbo

1 2 3 4

```
[hadoop@hadoop05 ~]$ telnet localhost 44444
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
hello huangbo
OK
1 2 3 4
OK
```

4、Flume-Agent 接收的结果：

```
2018-07-02 08:34:06,943 (lifecycleSupervisor-1-1) [INFO - org.apache.flume.source.NetcatSource.start(Netcat
Source.java:166)] Created serverSocket:sun.nio.ch.ServerSocketChannelImpl[/127.0.0.1:44444]
2018-07-02 08:34:13,390 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.Logge
rSink.process(LoggerSink.java:95)] Event: { headers:{} body: 68 65 6C 6C 6F 20 68 75 61 6E 67 62 6F 0D
 hello huangbo. }
2018-07-02 08:34:17,919 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.Logge
rSink.process(LoggerSink.java:95)] Event: { headers:{} body: 31 20 32 20 33 20 34 0D
 1 2 3 4. }
```

总结套路：

1、 一个 agent 的形式，是由一个配置文件来决定，也就是说，你要启动什么类型的 agent 从哪里收集数据到哪里，那么请在配置文件中指定 agent 的相关配置

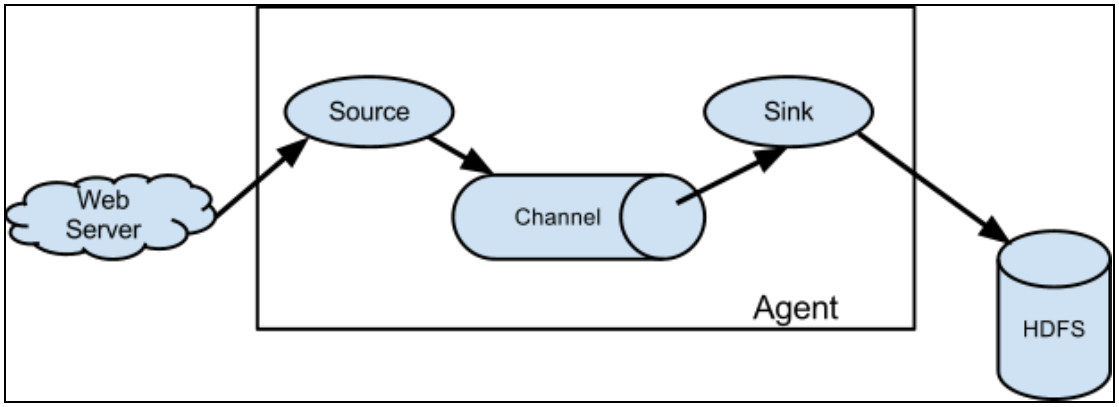
2、 agent 的配置分为五个重要的组成部分：

agent 名称和对应的 source,channel,sink 组件，
source 相关组件，可以配置多个成一组
channel 相关组件配置，可以配置多个成一组
sink 相关组件配置，可以配置多个成一组
绑定 source,sink,channel 们之间的关系

5.2、Flume 实战案例

5.2.1、采集目录到 HDFS

采集需求：某服务器的某特定目录下，会不断产生新的文件，每当有新文件出现，就需要把文件采集到 HDFS 中去



根据需求，首先定义以下 3 大要素

数据源组件，即 source ——监控文件目录：spooldir

spooldir 特性：

1、监视一个目录，只要目录中出现新文件，就会采集文件中的内容

2、采集完成的文件，会被 agent 自动添加一个后缀：.COMPLETED

3、所监视的目录中不允许重复出现相同文件名的文件

下沉组件，即 sink——HDFS 文件系：hdfs sink

通道组件，即 channel——可用 file channel 也可以用内存 channel

配置文件编写：**spooldir-hdfs.properties**

```
#定义三大组件的名称
agent1.sources = source1
agent1.sinks = sink1
agent1.channels = channel1

# 配置 source 组件
agent1.sources.source1.type = spooldir
agent1.sources.source1.spoolDir = /home/hadoop/flumelogs/
agent1.sources.source1.fileHeader = false

# 配置 sink 组件
agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path=hdfs://myha01/flume_log/%y-%m-%d/%H-%M
agent1.sinks.sink1.hdfs.filePrefix = events
agent1.sinks.sink1.hdfs.maxOpenFiles = 5000
agent1.sinks.sink1.hdfs.batchSize= 100
agent1.sinks.sink1.hdfs.fileType = DataStream
agent1.sinks.sink1.hdfs.writeFormat =Text
agent1.sinks.sink1.hdfs.rollSize = 102400
agent1.sinks.sink1.hdfs.rollCount = 1000000
agent1.sinks.sink1.hdfs.rollInterval = 60
#agent1.sinks.sink1.hdfs.round = true
#agent1.sinks.sink1.hdfs.roundValue = 10
#agent1.sinks.sink1.hdfs.roundUnit = minute
agent1.sinks.sink1.hdfs.useLocalTimeStamp = true

# Use a channel which buffers events in memory
agent1.channels.channel1.type = memory
agent1.channels.channel1.keep-alive = 120
agent1.channels.channel1.capacity = 500000
agent1.channels.channel1.transactionCapacity = 600

# Bind the source and sink to the channel
agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1
```

Channel 参数解释：

capacity：默认该通道中最大的可以存储的 event 数量

trasactionCapacity：每次最大可以从 source 中拿到或者送到 sink 中的 event 数量

keep-alive：event 添加到通道中或者移出的允许时间

启动：

```
bin/flume-ng agent -c conf -f agentconf/spooldir-hdfs.properties -n agent1
```

测试：
1、如果 HDFS 集群是高可用集群，那么必须要放入 core-site.xml 和 hdfs-site.xml 文件到 $FLUME_HOME/conf 目录中
2、查看监控的/home/hadoop/flumelogs 文件夹中的文件是否被正确上传到 HDFS 上
3、在该目录中创建文件，或者从其他目录往该目录加入文件，验证是否新增的文件能被自动的上传到 HDFS

5.2.2、采集文件到 HDFS

采集需求：比如业务系统使用 log4j 生成的日志，日志内容不断增加，需要把追加到日志文件中的数据实时采集到 HDFS

根据需求，首先定义以下 3 大要素
采集源，即 source--监控文件内容更新：exec "tail -F file"
下沉目标，即 sink--HDFS 文件系：hdfs sink
Source 和 sink 之间的传递通道——channel，可用 file channel 也可以用内存 channel

配置文件编写：**tail-hdfs.properties**

```
agent1.sources = source1
agent1.sinks = sink1
agent1.channels = channel1

# Describe/configure tail -F source1
agent1.sources.source1.type = exec
agent1.sources.source1.command = tail -F /home/hadoop/flumelogs/catalina.out
agent1.sources.source1.channels = channel1

# Describe sink1
agent1.sinks.sink1.type = hdfs
#a1.sinks.k1.channel = c1
agent1.sinks.sink1.hdfs.path =hdfs://myha01/weblog/flume-event/%y-%m-%d/%H-%M
agent1.sinks.sink1.hdfs.filePrefix = tomcat_

agent1.sinks.sink1.hdfs.maxOpenFiles = 5000
agent1.sinks.sink1.hdfs.batchSize= 100
agent1.sinks.sink1.hdfs.fileType = DataStream
agent1.sinks.sink1.hdfs.writeFormat =Text
agent1.sinks.sink1.hdfs.rollSize = 102400
agent1.sinks.sink1.hdfs.rollCount = 1000000
agent1.sinks.sink1.hdfs.rollInterval = 60
agent1.sinks.sink1.hdfs.round = true
```

```
agent1.sinks.sink1.hdfs.roundValue = 10
agent1.sinks.sink1.hdfs.roundUnit = minute
agent1.sinks.sink1.hdfs.useLocalTimeStamp = true

# Use a channel which buffers events in memory
agent1.channels.channel1.type = memory
agent1.channels.channel1.keep-alive = 120
agent1.channels.channel1.capacity = 500000
agent1.channels.channel1.transactionCapacity = 600

# Bind the source and sink to the channel
agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1
```

启动：

```
bin/flume-ng agent -c conf -f agentconf/tail-hdfs.properties -n agent1
```

测试：

1、模拟像指定的日志文件/home/hadoop/flumelogs/catalina.out 追加内容

2、验证 HDFS 上的对应文件是否有新增内容

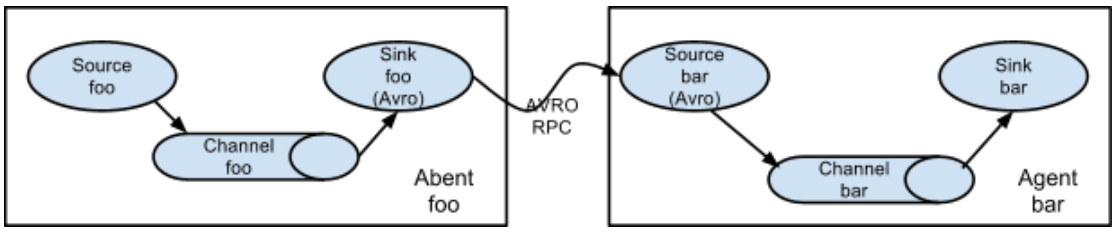
5.2.3、多路复用采集

作业

5.2.4、多 agent 串联采集

架构设计：

从 hadoop04 的 flume agent 传送数据到 hadoop05 的 flume agent：



如现在我在两台机器上的测试，192.168.123.104 和 192.168.123.105 上面做 agent 的传递：

分别是：

hadoop04：**tail-avro.properties**

使用 exec “tail -F /home/hadoop/testlog/welog.log”获取采集数据

使用 avro sink 数据都下一个 agent

hadoop05：**avro-hdfs.properties**

使用 avro 接收采集数据

使用 hdfs sink 数据到目的地

第一步：准备 hadoop04

在 IP 为 192.168.123.104 的 hadoop04 上的 agentconf 下创建一个 **tail-avro.properties**：

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F /home/hadoop/testlog/date.log
a1.sources.r1.channels = c1

# Describe the sink
a1.sinks.k1.type = avro
a1.sinks.k1.channel = c1
a1.sinks.k1.hostname = hadoop05
a1.sinks.k1.port = 4141
a1.sinks.k1.batch-size = 2

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

第二步：准备 hadoop05

再在 IP 为 192.168.123.105 的 hadoop05 机器上配置采集方案 **avro-hdfs.properties**：

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = avro
a1.sources.r1.channels = c1
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 4141

# Describe k1
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path =hdfs://myha01/testlog/flume-event/%y-%m-%d/%H-%M
a1.sinks.k1.hdfs.filePrefix = date_
```

```
a1.sinks.k1.hdfs.maxOpenFiles = 5000
a1.sinks.k1.hdfs.batchSize= 100
a1.sinks.k1.hdfs.fileType = DataStream
a1.sinks.k1.hdfs.writeFormat =Text
a1.sinks.k1.hdfs.rollSize = 102400
a1.sinks.k1.hdfs.rollCount = 1000000
a1.sinks.k1.hdfs.rollInterval = 60
a1.sinks.k1.hdfs.round = true
a1.sinks.k1.hdfs.roundValue = 10
a1.sinks.k1.hdfs.roundUnit = minute
a1.sinks.k1.hdfs.useLocalTimeStamp = true

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

第三步：最终测试

1、首先启动 hadoop05 机器上的 agent

bin/flume-ng agent -c conf -n a1 -f agentconf/avro-hdfs.properties -Dflume.root.logger=INFO,console

2、再启动 hadoop04 上的 agent

bin/flume-ng agent -c conf -n a1 -f agentconf/tail-avro.properties -Dflume.root.logger=INFO,console

3、执行一个普通的脚本往 hadoop04 的**/home/hadoop/testlog/date.log** 中追加数据：

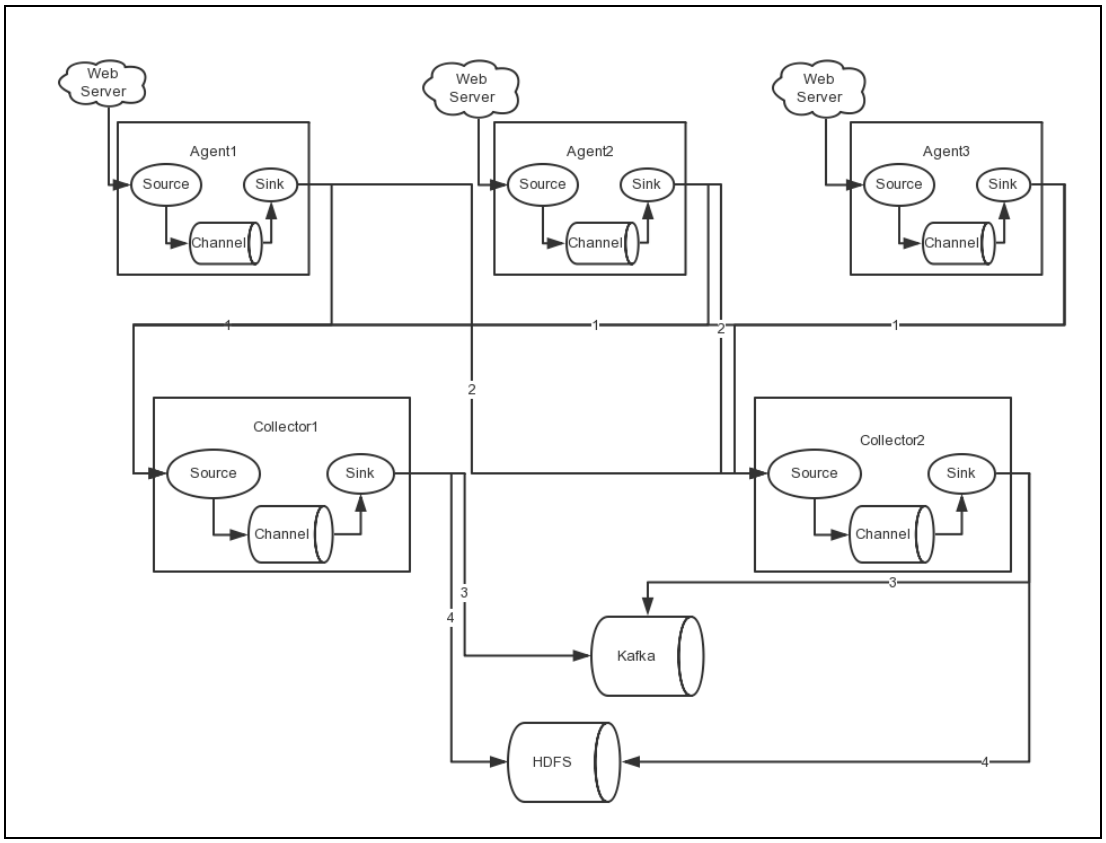
```
#!/bin/bash
while true
do
        echo `date` >> /home/hadoop/testlog/date.log
        sleep 1
done
```

4、至此会发现在 hadoop04 agent 发送的数据会转到 hadoop05 agent，然后被 sink 到了 HDFS 的对应目录 **hdfs://myha01/testlog/flume-event/**

5.2.5、高可用部署采集

第一步：

Flume-NG 的高可用架构图：



图中，我们可以看出，Flume 的存储可以支持多种，这里只列举了 HDFS 和 Kafka（如：存储最新的一周日志，并给 Storm 系统提供实时日志流）。

第二步：节点分配

Flume 的 Agent 和 Collector 分布如下表所示：

名称	Host	角色
Agent1	hadoop02	日志服务器
Agent2	hadoop03	日志服务器
Agent3	hadoop04	日志服务器
Collector1	hadoop04	AgentMaster1
Collector1	hadoop05	AgentMaster2

图中所示，Agent1，Agent2，Agent3 数据分别流入到 Collector1 和 Collector2，Flume NG 本身提供了 Failover 机制，可以自动切换和恢复。在上图中，有 3 个产生日志服务器分布在不同的机房，要把所有的日志都收集到一个集群中存储。下面我们开发配置 Flume NG 集群

第三步：配置信息

在下面单点 Flume 中，基本配置都完成了，我们只需要新添加两个配置文件，它们是 ha_agent.properties 和 ha_collector.properties，其配置内容如下所示：

ha_agent.properties 配置：

```
#agent name: agent1
agent1.channels = c1
agent1.sources = r1
#set gruop
agent1.sinkgroups = g1
#set sinks
agent1.sinks = k1 k2
#set sink group
agent1.sinkgroups.g1.sinks = k1 k2
#set failover
agent1.sinkgroups.g1.processor.type = failover
agent1.sinkgroups.g1.processor.priority.k1 = 10
agent1.sinkgroups.g1.processor.priority.k2 = 1
agent1.sinkgroups.g1.processor.maxpenalty = 10000

#set channel
agent1.channels.c1.type = memory
agent1.channels.c1.capacity = 1000
agent1.channels.c1.transactionCapacity = 100

agent1.sources.r1.channels = c1
agent1.sources.r1.type = exec
agent1.sources.r1.command = tail -F /home/hadoop/testlog/testha.log

agent1.sources.r1.interceptors = i1 i2
agent1.sources.r1.interceptors.i1.type = static
agent1.sources.r1.interceptors.i1.key = Type
agent1.sources.r1.interceptors.i1.value = LOGIN
agent1.sources.r1.interceptors.i2.type = timestamp

# set sink1
agent1.sinks.k1.channel = c1
agent1.sinks.k1.type = avro
agent1.sinks.k1.hostname = hadoop04
agent1.sinks.k1.port = 52020

# set sink2
agent1.sinks.k2.channel = c1
agent1.sinks.k2.type = avro
agent1.sinks.k2.hostname = hadoop05
agent1.sinks.k2.port = 52020
```

ha_collector.properties 配置：

```
#set agent name
a1.sources = r1
a1.channels = c1
a1.sinks = k1

#set channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# other node,nna to nns
a1.sources.r1.type = avro
## 当前主机为什么，就修改成什么主机名
a1.sources.r1.bind = hadoop04
a1.sources.r1.port = 52020
a1.sources.r1.interceptors = i1
a1.sources.r1.interceptors.i1.type = static
a1.sources.r1.interceptors.i1.key = Collector
## 当前主机为什么，就修改成什么主机名
a1.sources.r1.interceptors.i1.value = hadoop04
a1.sources.r1.channels = c1

#set sink to hdfs
a1.sinks.k1.type=hdfs
a1.sinks.k1.hdfs.path= hdfs://myha01/flume_ha/loghdfs
a1.sinks.k1.hdfs.fileType=DataStream
a1.sinks.k1.hdfs.writeFormat=TEXT
a1.sinks.k1.hdfs.rollInterval=10
a1.sinks.k1.hdfs.filePrefix=%Y-%m-%d
a1.sinks.k1.channel=c1
```

注意：在把 ha_collector.properties 文件拷贝到另外一台 collector 的时候，记得更改该配置文件中的主机名。在该配置文件中有注释

第四步：启动

先启动 hadoop04 和 hadoop05 上的 collector 角色：

bin/flume-ng agent -c conf -f agentconf/ha_collector.properties -n a1 -Dflume.root.logger=INFO,console

然后启动 hadoop02，hadoop03，hadoop04 上的 agent 角色：

bin/flume-ng agent -c conf -f agentconf/ha_agent.properties -n agent1 -Dflume.root.logger=INFO,console

5.2.6、更多 Source 和 Sink 组件

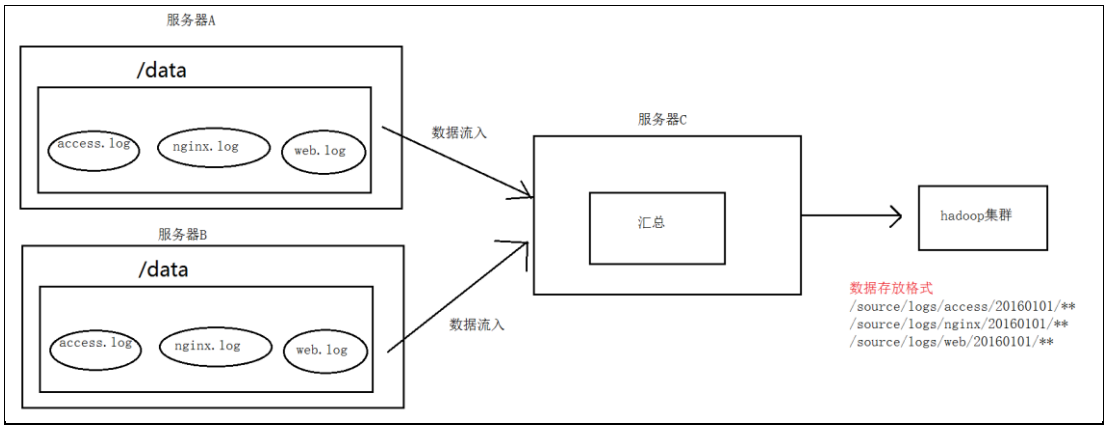
更多 Sources：http://flume.apache.org/FlumeUserGuide.html#flume-sources
更多 Channels：http://flume.apache.org/FlumeUserGuide.html#flume-channels
更多 Sinks：http://flume.apache.org/FlumeUserGuide.html#flume-sinks

6、综合案例

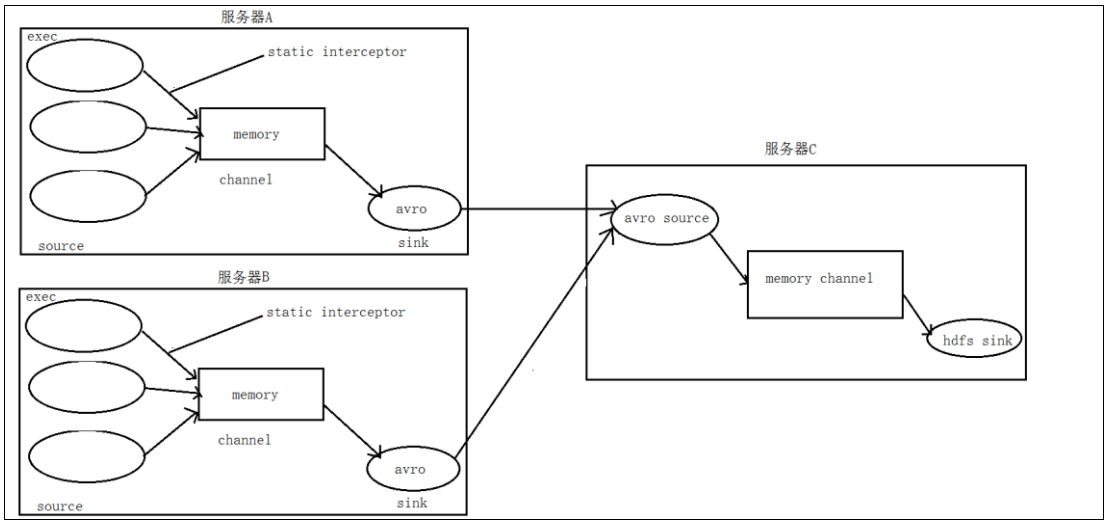
6.1、案例场景/需求

A、B 两台日志服务机器实时生产日志主要类型为 access.log、nginx.log、web.log
现在要求：
把 A、B 机器中的 access.log、nginx.log、web.log 采集汇总到 C 机器上然后统一收集到 HDFS 中。
但是在 hdfs 中要求的目录为：
/source/logs/access/20160101/**
/source/logs/nginx/20160101/**
/source/logs/web/20160101/**

6.2、场景分析



6.3、数据处理流程分析



6.4、需求实现

第一：准备 3 台服务器

服务器 A 对应的 IP 为 192.168.123.103，主机名为 hadoop03
服务器 B 对应的 IP 为 192.168.123.104，主机名为 hadoop04
服务器 C 对应的 IP 为 192.168.123.105，主机名为 hadoop05

第二：设计采集方案 exec_source_avro_sink.properties

在服务器 hadoop03 和服务器 hadoop04 上的$FLUME_HOME/agentconf 创建采集方案的配置文件 exec_source_avro_sink.properties，文件内容为：

```
# 指定各个核心组件
a1.sources = r1 r2 r3
a1.sinks = k1
a1.channels = c1

# 准备数据源
## static 拦截器的功能就是往采集到的数据的 header 中插入自己定义的 key-value 对
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F /home/hadoop/flume_data/access.log
a1.sources.r1.interceptors = i1
a1.sources.r1.interceptors.i1.type = static
a1.sources.r1.interceptors.i1.key = type
a1.sources.r1.interceptors.i1.value = access

a1.sources.r2.type = exec
a1.sources.r2.command = tail -F /home/hadoop/flume_data/nginx.log
a1.sources.r2.interceptors = i2
```

```
a1.sources.r2.interceptors.i2.type = static
a1.sources.r2.interceptors.i2.key = type
a1.sources.r2.interceptors.i2.value = nginx

a1.sources.r3.type = exec
a1.sources.r3.command = tail -F /home/hadoop/flume_data/web.log
a1.sources.r3.interceptors = i35
a1.sources.r3.interceptors.i3.type = static
a1.sources.r3.interceptors.i3.key = type
a1.sources.r3.interceptors.i3.value = web

# Describe the sink
a1.sinks.k1.type = avro
a1.sinks.k1.hostname = hadoop05
a1.sinks.k1.port = 41414

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 20000
a1.channels.c1.transactionCapacity = 10000

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sources.r2.channels = c1
a1.sources.r3.channels = c1
a1.sinks.k1.channel = c1
```

第三：准备 avro_source_hdfs_sink.properties 配置文件

在服务器 C 上的$FLUME_HOME/agentconf 中创建配置文件 avro_source_hdfs_sink.properties 文件内容为：

```
#定义 agent 名， source、channel、sink 的名称
a1.sources = r1
a1.sinks = k1
a1.channels = c1

#定义 source
a1.sources.r1.type = avro
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port =41414

#添加时间拦截器
a1.sources.r1.interceptors = i1
a1.sources.r1.interceptors.i1.type=org.apache.flume.interceptor.TimestampInterceptor$Builder
```

```
#定义 channels
a1.channels.c1.type = memory
a1.channels.c1.capacity = 20000
a1.channels.c1.transactionCapacity = 10000

#定义 sink
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path=hdfs://myha01/source/logs/%{type}/%Y%m%d
a1.sinks.k1.hdfs.filePrefix =events
a1.sinks.k1.hdfs.fileType = DataStream
a1.sinks.k1.hdfs.writeFormat = Text
#时间类型
a1.sinks.k1.hdfs.useLocalTimeStamp = true
#生成的文件不按条数生成
a1.sinks.k1.hdfs.rollCount = 0
#生成的文件按时间生成
a1.sinks.k1.hdfs.rollInterval = 30
#生成的文件按大小生成
a1.sinks.k1.hdfs.rollSize = 10485760
#批量写入 hdfs 的个数
a1.sinks.k1.hdfs.batchSize = 20
#flume 操作 hdfs 的线程数（包括新建，写入等）
a1.sinks.k1.hdfs.threadsPoolSize=10
#操作 hdfs 超时时间
a1.sinks.k1.hdfs.callTimeout=30000

#组装 source、channel、sink
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

第四：启动

配置完成之后，在服务器 A 和 B 上的/home/hadoop/data 有数据文件 access.log、nginx.log、web.log。

先启动服务器 C（hadoop05）上的 flume，启动命令：在 flume 安装目录下执行：

bin/flume-ng agent -c conf -f agentconf/avro_source_hdfs_sink.properties -name a1 -Dflume.root.logger=DEBUG,console

然后在启动服务器上的 A（hadoop03）和 B（hadoop04），启动命令：在 flume 安装目录下执行：

bin/flume-ng agent -c conf -f agentconf/exec_source_avro_sink.properties -name a1 -Dflume.root.logger=DEBUG,console

第五：测试

自行测试