尚硅谷大数据技术之 Flume

（作者：尚硅谷大数据研发部）

版本：V3.0

第 **1** 章 **Flume** 概述

# Flume 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。Flume 基于流式架构，灵活简单。



为什么选用Flume

Flume最主要的作用就是，实时读取服务器本地磁盘的数据，将数据写入到HDFS。

Kafka

网络端口数据

Flume

Java后台日志数据

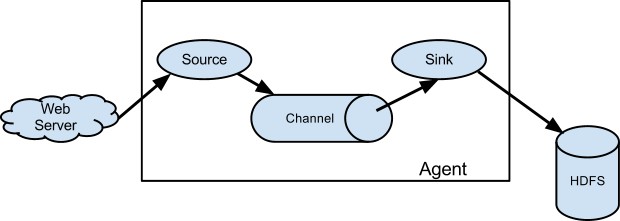
HDFS

服务器本地磁盘文件夹

Python爬虫数据

# Flume 基础架构

Flume 组成架构如下图所示。



# Agent

Agent 是一个 JVM 进程，它以事件的形式将数据从源头送至目的。

Agent 主要有 3 个部分组成，Source、Channel、Sink。

# 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，包括 avro、thrift、exec、jms、spooling directory、netcat、taildir、 sequence generator、syslog、http、legacy。

# 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 组件目的地包括 hdfs、logger、avro、thrift、ipc、file、HBase、solr、自定义。

# Channel

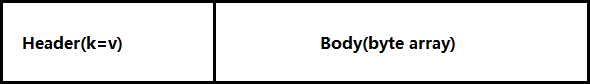
Channel 是位于 Source 和 Sink 之间的缓冲区。因此，Channel 允许 Source 和 Sink 运作在不同的速率上。Channel 是线程安全的，可以同时处理几个 Source 的写入操作和几个 Sink 的读取操作。

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。

# Event

传输单元，Flume 数据传输的基本单元，以 Event 的形式将数据从源头送至目的地。 Event 由 Header 和Body 两部分组成，Header 用来存放该 event 的一些属性，为K-V 结构， Body 用来存放该条数据，形式为字节数组。

第 **2** 章 **Flume** 入门

# Flume 安装部署

## 安装地址

* + - 1. Flume 官网地址：<http://flume.apache.org/>
      2. 文档查看地址：<http://flume.apache.org/FlumeUserGuide.html>
      3. 下载地址：<http://archive.apache.org/dist/flume/>

## 安装部署

1. 将 apache-flume-1.9.0-bin.tar.gz 上传到 linux 的/opt/software 目录下
2. 解压 apache-flume-1.9.0-bin.tar.gz 到/opt/module/目录下

[atguigu@hadoop102 software]$ tar -zxf /opt/software/apache- flume-1.9.0-bin.tar.gz -C /opt/module/

1. 修改 apache-flume-1.9.0-bin 的名称为 flume

[atguigu@hadoop102 module]$ mv /opt/module/apache-flume-1.9.0-bin

/opt/module/flume

1. 将 lib 文件夹下的guava-11.0.2.jar 删除以兼容 Hadoop 3.1.3

[atguigu@hadoop102 lib]$ rm /opt/module/flume/lib/guava- 11.0.2.jar

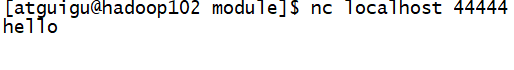
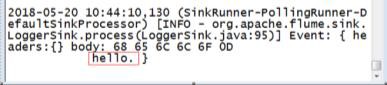
# Flume 入门案例

## 监控端口数据官方案例

1. 案例需求：

使用 Flume 监听一个端口，收集该端口数据，并打印到控制台。

1. 需求分析：



监听数据端口案例分析

1 通过netcat工具向本机的44444端 2 Flume监控本机的44444端口。 3 Flume将获取的数据通过口发送数据 通过Flume的source端读取数据。 Sink端写出到控制台

hello

hello

测试命令

nc localhost 44444

bin/flume-ng agent --conf conf/

--name a1

--conf-file job/flume-telnet.conf

-Dflume.root.logger==INFO,console

控制台

44444端口

1. 实现步骤：
   1. 安装 netcat 工具

[atguigu@hadoop102 software]$ sudo yum install -y nc

* 1. 判断 44444 端口是否被占用

[atguigu@hadoop102 flume-telnet]$ sudo netstat -nlp | grep 44444

* 1. 创建 Flume Agent 配置文件 flume-netcat-logger.conf
  2. 在 flume 目录下创建 job 文件夹并进入 job 文件夹。

[atguigu@hadoop102 flume]$ mkdir job [atguigu@hadoop102 flume]$ cd job/

* 1. 在 job 文件夹下创建 Flume Agent 配置文件 flume-netcat-logger.conf。

[atguigu@hadoop102 job]$ vim flume-netcat-logger.conf

* 1. 在 flume-netcat-logger.conf 文件中添加如下内容。

添加内容如下：

# Name the components on this agent a1.sources = r1

a1.sinks = k1 a1.channels = c1

# Describe/configure the source a1.sources.r1.type = netcat a1.sources.r1.bind = localhost a1.sources.r1.port = 44444

# Describe the sink a1.sinks.k1.type = logger

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

a1.channels.c1.transactionCapacity = 100

注：配置文件来源于官方手册 <http://flume.apache.org/FlumeUserGuide.html>



配置文件解析

# Name the components on this agent

**a1:表示agent的名称**

a1.sources = r1 r1:表示a1的Source的名称 a1.sinks = k1 k1:表示a1的Sink的名称 a1.channels = c1 c1:表示a1的Channel的名称

# Describe/configure the source a1.sources.r1.type = netcat a1.sources.r1.bind = localhost a1.sources.r1.port = 44444

表示a1的输入源类型为netcat端口类型

表示a1的监听的主机 表示a1的监听的端口号

# Describe the sink

a1.sinks.k1.type = logger 表示a1的输出目的地是控制台logger类型

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

a1.channels.c1.transactionCapacity = 100

表示a1的channel类型是memory内存型表示a1的channel总容量1000个event

表示a1的channel传输时收集到了100条event以后再去提交事务

# Bind the source and sink to the channel

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

表示将r1和c1连接起来

表示将k1和c1连接起来

* 1. 先开启 flume 监听端口

第一种写法：

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a1 --conf-file job/flume-netcat-logger.conf - Dflume.root.logger=INFO,console

第二种写法：

[atguigu@hadoop102 flume]$ bin/flume-ng agent -c conf/ -n a1 -f job/flume-netcat-logger.conf -Dflume.root.logger=INFO,console

参数说明：

--conf/-c：表示配置文件存储在 conf/目录

--name/-n：表示给 agent 起名为 a1

--conf-file/-f：flume 本次启动读取的配置文件是在 job 文件夹下的 flume-telnet.conf

文件。

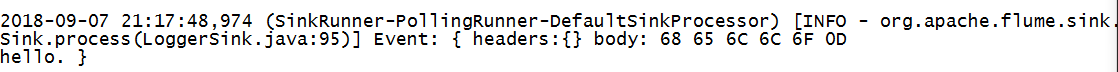
-Dflume.root.logger=INFO,console ：-D 表示 flume 运行时动态修改 flume.root.logger参数属性值，并将控制台日志打印级别设置为 INFO 级别。日志级别包括:log、info、warn、 error。

* 1. 使用 netcat 工具向本机的 44444 端口发送内容

[atguigu@hadoop102 ~]$ nc localhost 44444 hello

atguigu

* 1. 在 Flume 监听页面观察接收数据情况



## 实时监控单个追加文件

1. 案例需求：实时监控 Hive 日志，并上传到 HDFS 中
2. 需求分析：



实时读取本地文件到HDFS案例

Hive实时更新日志

3 开启Hive，

生成日志

1. 创建符合条件的flume配置文件
2. 执行配置文件，开启监控

4 查看HDFS

上数据

HDFS

Hive日志文件

/opt/module/hive/lo gs/hive.log

Exec Source HDFS Sink Flume监控文件

Memory Channel

1. 实现步骤：
   1. Flume 要想将数据输出到 HDFS，依赖 Hadoop 相关 jar 包

检查/etc/profile.d/my\_env.sh 文件，确认 Hadoop 和 Java 环境变量配置正确

JAVA\_HOME=/opt/module/jdk1.8.0\_212 HADOOP\_HOME=/opt/module/ha/hadoop-3.1.3

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

* 1. 创建 flume-file-hdfs.conf 文件创建文件

[atguigu@hadoop102 job]$ vim flume-file-hdfs.conf

注：要想读取 Linux 系统中的文件，就得按照 Linux 命令的规则执行命令。由于 Hive日志在 Linux 系统中所以读取文件的类型选择：exec 即 execute 执行的意思。表示执行 Linux 命令来读取文件。

添加如下内容

# Name the components on this agent a2.sources = r2

a2.sinks = k2

### 尚硅谷大数据技术之 Flume

—————————————————————————————

a2.channels = c2

# Describe/configure the source a2.sources.r2.type = exec

a2.sources.r2.command = tail -F /opt/module/hive/logs/hive.log

# Describe the sink a2.sinks.k2.type = hdfs

a2.sinks.k2.hdfs.path = hdfs://hadoop102:9820/flume/%Y%m%d/%H #上传文件的前缀

a2.sinks.k2.hdfs.filePrefix = logs- #是否按照时间滚动文件夹 a2.sinks.k2.hdfs.round = true

#多少时间单位创建一个新的文件夹

a2.sinks.k2.hdfs.roundValue = 1 #重新定义时间单位 a2.sinks.k2.hdfs.roundUnit = hour #是否使用本地时间戳

a2.sinks.k2.hdfs.useLocalTimeStamp = true #积攒多少个 Event 才 flush 到 HDFS 一次 a2.sinks.k2.hdfs.batchSize = 100

#设置文件类型，可支持压缩

a2.sinks.k2.hdfs.fileType = DataStream #多久生成一个新的文件 a2.sinks.k2.hdfs.rollInterval = 60

#设置每个文件的滚动大小

a2.sinks.k2.hdfs.rollSize = 134217700 #文件的滚动与Event 数量无关 a2.sinks.k2.hdfs.rollCount = 0

# Use a channel which buffers events in memory a2.channels.c2.type = memory a2.channels.c2.capacity = 1000

a2.channels.c2.transactionCapacity = 100

# Bind the source and sink to the channel a2.sources.r2.channels = c2 a2.sinks.k2.channel = c2

注意：对于所有与时间相关的转义序列，Event Header 中必须存在以 “timestamp”的 key（除非 hdfs.useLocalTimeStamp 设置为 true，此方法会使用 TimestampInterceptor 自动添加 timestamp）。

a3.sinks.k3.hdfs.useLocalTimeStamp = true



实时读取本地文件到HDFS案例

# Name the components on this agent

a2.sources = r2 #定义source

a2.sinks = k2 #定义sink

a2.channels = c2 #定义channel

# Use a channel which buffers events in memory a2.channels.c2.type = memory a2.channels.c2.capacity = 1000

a2.channels.c2.transactionCapacity = 100

# Describe/configure the source

a2.sources.r2.type = exec #定义source类型为exec可执行命令的 # Bind the source and sink to the channel a2.sources.r2.command = tail -F /opt/module/hive/logs/hive.log a2.sources.r2.channels = c2

a2.sources.r2.shell = /bin/bash -c

# Describe the sink a2.sinks.k2.type = hdfs

a2.sinks.k2.hdfs.path = hdfs://hadoop102:9000/flume/%Y%m%d/%H

#执行shell脚本的绝对路径

a2.sinks.k2.channel = c2

a2.sinks.k2.hdfs.filePrefix = logs- #上传文件的前缀 a2.sinks.k2.hdfs.round = true #是否按照时间滚动文件夹 a2.sinks.k2.hdfs.roundValue = 1 #多少时间单位创建一个新的文件夹

a2.sinks.k2.hdfs.roundUnit = hour

#重新定义时间单位

a2.sinks.k2.hdfs.useLocalTimeStamp = true #是否使用本地时间戳

a2.sinks.k2.hdfs.batchSize = 1000 a2.sinks.k2.hdfs.fileType = DataStream a2.sinks.k2.hdfs.rollInterval = 60

a2.sinks.k2.hdfs.rollSize = 134217700

a2.sinks.k2.hdfs.rollCount = 0

#积攒多少个Event才flush到HDFS一次

#设置文件类型，可支持压缩

#多久生成一个新的文件

#设置每个文件的滚动大小

#文件的滚动与Event数量无关

* 1. 运行Flume

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a2 --conf-file job/flume-file-hdfs.conf

* 1. 开启Hadoop 和 Hive 并操作 Hive 产生日志

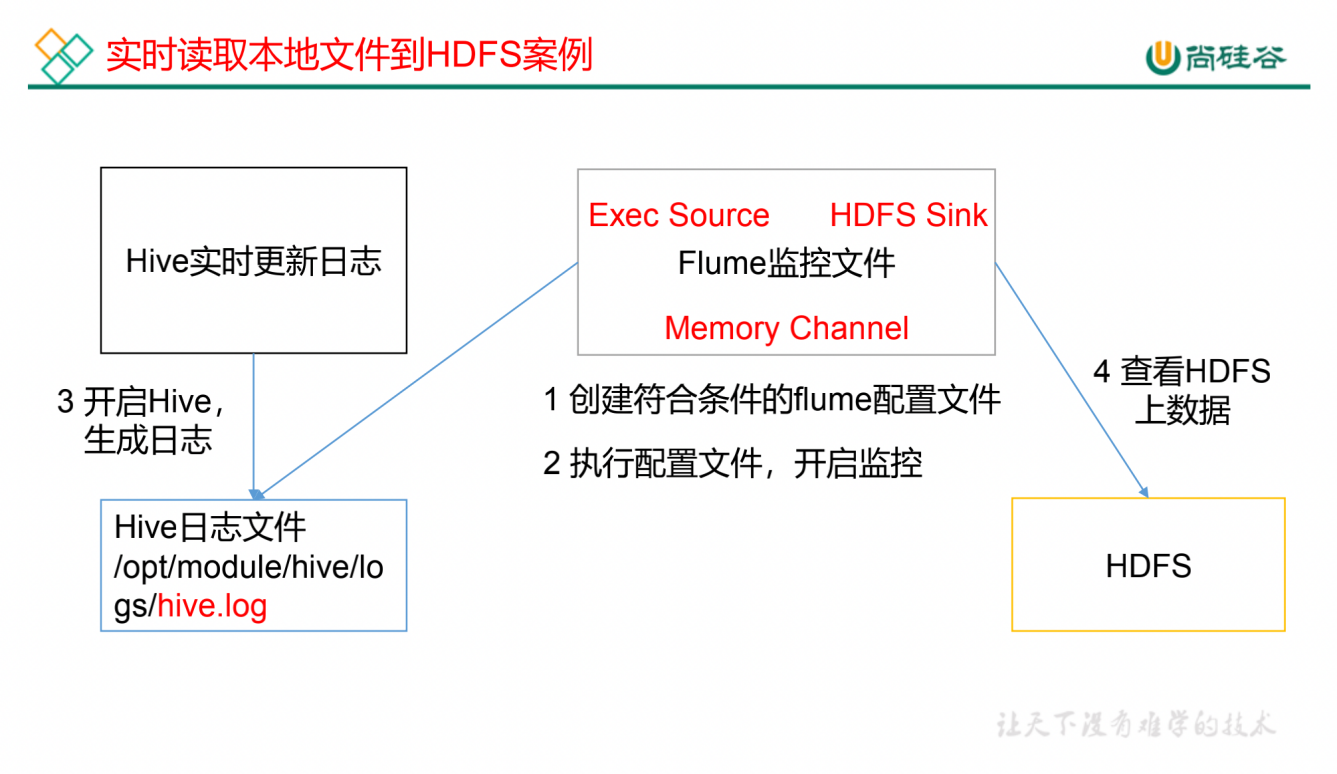
[atguigu@hadoop102 hadoop-2.7.2]$ sbin/start-dfs.sh [atguigu@hadoop103 hadoop-2.7.2]$ sbin/start-yarn.sh

[atguigu@hadoop102 hive]$ bin/hive hive (default)>

* 1. 在 HDFS 上查看文件。

## 实时监控目录下多个新文件

1. 案例需求：使用 Flume 监听整个目录的文件，并上传至 HDFS
2. 需求分析：



1. 实现步骤：
   1. 创建配置文件 flume-dir-hdfs.conf

创建一个文件

[atguigu@hadoop102 job]$ vim flume-dir-hdfs.conf

添加如下内容

a3.sources = r3 a3.sinks = k3 a3.channels = c3

# Describe/configure the source a3.sources.r3.type = spooldir a3.sources.r3.spoolDir = /opt/module/flume/upload a3.sources.r3.fileSuffix = .COMPLETED a3.sources.r3.fileHeader = true

#忽略所有以.tmp 结尾的文件，不上传

a3.sources.r3.ignorePattern = ([^ ]\*\.tmp)

# Describe the sink a3.sinks.k3.type = hdfs a3.sinks.k3.hdfs.path =

hdfs://hadoop102:9820/flume/upload/%Y%m%d/%H #上传文件的前缀

a3.sinks.k3.hdfs.filePrefix = upload- #是否按照时间滚动文件夹 a3.sinks.k3.hdfs.round = true

#多少时间单位创建一个新的文件夹

a3.sinks.k3.hdfs.roundValue = 1 #重新定义时间单位 a3.sinks.k3.hdfs.roundUnit = hour #是否使用本地时间戳

a3.sinks.k3.hdfs.useLocalTimeStamp = true #积攒多少个 Event 才 flush 到 HDFS 一次 a3.sinks.k3.hdfs.batchSize = 100



实时读取目录文件到HDFS案例

a3.sources = r3 a3.sinks = k3 a3.channels = c3

#定义source #定义sink

#定义channel

a3.sinks.k3.hdfs.rollInterval = 60 #多久生成新文件

# Describe/configure the source

a3.sinks.k3.hdfs.rollSize = 134217700 #多大生成新文件 a3.sinks.k3.hdfs.rollCount = 0 #多少event生成新文件

# Use a channel which buffers events in memory

a3.sources.r3.type = spooldir #定义source类型为目录 a3.channels.c3.type = memory a3.sources.r3.spoolDir = /opt/module/flume/upload #定义监控目录 a3.channels.c3.capacity = 1000 a3.sources.r3.fileSuffix = .COMPLETED #定义文件上传完，后缀 a3.channels.c3.transactionCapacity = 100 a3.sources.r3.fileHeader = true #是否有文件头

a3.sources.r3.ignorePattern = ([^ ]\*\.tmp)

# Describe the sink

a3.sinks.k3.type = hdfs

#忽略所有以.tmp结尾的文件，不上传

#sink类型为hdfs

# Bind the source and sink to the channel a3.sources.r3.channels = c3 a3.sinks.k3.channel = c3

a3.sinks.k3.hdfs.path =

hdfs://hadoop102:9000/flume/upload/%Y%m%d/% H

a3.sinks.k3.hdfs.filePrefix = upload- a3.sinks.k3.hdfs.round = true a3.sinks.k3.hdfs.roundValue = 1 a3.sinks.k3.hdfs.roundUnit = hour a3.sinks.k3.hdfs.useLocalTimeStamp = true a3.sinks.k3.hdfs.batchSize = 100 a3.sinks.k3.hdfs.fileType = DataStream

#文件上传到hdfs的路径

#上传文件到hdfs的前缀

#是否按时间滚动文件

#多少时间单位创建一个新的文件夹 #重新定义时间单位

#是否使用本地时间戳

#积攒多少个Event才flush到HDFS一次

#设置文件类型，可支持压缩

#设置文件类型，可支持压缩

a3.sinks.k3.hdfs.fileType = DataStream #多久生成一个新的文件 a3.sinks.k3.hdfs.rollInterval = 60

#设置每个文件的滚动大小大概是 128M

a3.sinks.k3.hdfs.rollSize = 134217700 #文件的滚动与Event 数量无关 a3.sinks.k3.hdfs.rollCount = 0

# Use a channel which buffers events in memory a3.channels.c3.type = memory a3.channels.c3.capacity = 1000

a3.channels.c3.transactionCapacity = 100

# Bind the source and sink to the channel a3.sources.r3.channels = c3 a3.sinks.k3.channel = c3

* 1. 启动监控文件夹命令

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a3 --conf-file job/flume-dir-hdfs.conf

说明：在使用 Spooling Directory Source 时，不要在监控目录中创建并持续修改文件；上传完成的文件会以.COMPLETED 结尾；被监控文件夹每 500 毫秒扫描一次文件变动。

* 1. 向 upload 文件夹中添加文件

在/opt/module/flume 目录下创建 upload 文件夹

[atguigu@hadoop102 flume]$ mkdir upload

向 upload 文件夹中添加文件

[atguigu@hadoop102 upload]$ touch atguigu.txt [atguigu@hadoop102 upload]$ touch atguigu.tmp [atguigu@hadoop102 upload]$ touch atguigu.log

* 1. 查看 HDFS 上的数据

## 实时监控目录下的多个追加文件

Exec source 适用于监控一个实时追加的文件，不能实现断点续传；Spooldir Source适合用于同步新文件，但不适合对实时追加日志的文件进行监听并同步；而 Taildir Source适合用于监听多个实时追加的文件，并且能够实现断点续传。

1. 案例需求:使用 Flume 监听整个目录的实时追加文件，并上传至 HDFS
2. 需求分析:

3）



Taildir Source

HDFS Sink

Flume监控目录

Memory Channel

实时读取目录文件到HDFS案例

3.向监控文件追加内容 echo hello >> files/file1.txt echo hello >> files/file2.txt

1. 创建符合条件的flume配置文件
2. 执行配置文件，开启监控

4 查看HDFS

上数据

HDFS

被监控的目录

/opt/module/flume/files

1. 实现步骤：
   1. 创建配置文件 flume-taildir-hdfs.conf

创建一个文件

[atguigu@hadoop102 job]$ vim flume-taildir-hdfs.conf

添加如下内容

a3.sources = r3 a3.sinks = k3 a3.channels = c3

# Describe/configure the source a3.sources.r3.type = TAILDIR

a3.sources.r3.positionFile = /opt/module/flume/tail\_dir.json a3.sources.r3.filegroups = f1 f2

a3.sources.r3.filegroups.f1 = /opt/module/flume/files/.\*file.\* a3.sources.r3.filegroups.f2 = /opt/module/flume/files2/.\*log.\*

# Describe the sink a3.sinks.k3.type = hdfs a3.sinks.k3.hdfs.path =

hdfs://hadoop102:9820/flume/upload2/%Y%m%d/%H #上传文件的前缀

a3.sinks.k3.hdfs.filePrefix = upload-



实时读取目录文件到HDFS案例

a3.sources = r3 a3.sinks = k3 a3.channels = c3

#定义source #定义sink

#定义channel

a3.sinks.k3.hdfs.rollInterval = 60 #多久生成新文件

# Describe/configure the source a2.sources.r2.type = TAILDIR #定义source类型

a2.sources.r2.positionFile = /opt/module/flume/tail\_dir.json

a2.sources.r2.filegroups = f1

#指定position\_file位置

a2.sources.r2.filegroups.f1 = /opt/module/flume/files/file\*

#定义监控目录文件

# Describe the sink

a3.sinks.k3.type = hdfs

a3.sinks.k3.hdfs.rollSize = 134217700 #多大生成新文件 a3.sinks.k3.hdfs.rollCount = 0 #多少event生成新文件

# Use a channel which buffers events in memory a3.channels.c3.type = memory a3.channels.c3.capacity = 1000

a3.channels.c3.transactionCapacity = 100

# Bind the source and sink to the channel a3.sources.r3.channels = c3 a3.sinks.k3.channel = c3

a3.sinks.k3.hdfs.path #sink类型为hdfs =

hdfs://hadoop102:9000/flume/upload/%Y%m%d/%H a3.sinks.k3.hdfs.filePrefix = upload- a3.sinks.k3.hdfs.round = true a3.sinks.k3.hdfs.roundValue = 1 a3.sinks.k3.hdfs.roundUnit = hour a3.sinks.k3.hdfs.useLocalTimeStamp = true a3.sinks.k3.hdfs.batchSize = 100 a3.sinks.k3.hdfs.fileType = DataStream

#文件上传到hdfs的路径

#上传文件到hdfs的前缀

#是否按时间滚动文件

#多少时间单位创建一个新的文件夹 #重新定义时间单位

#是否使用本地时间戳

#积攒多少个Event才flush到HDFS一次

#设置文件类型，可支持压缩

#是否按照时间滚动文件夹

a3.sinks.k3.hdfs.round = true #多少时间单位创建一个新的文件夹 a3.sinks.k3.hdfs.roundValue = 1

#重新定义时间单位

a3.sinks.k3.hdfs.roundUnit = hour #是否使用本地时间戳

a3.sinks.k3.hdfs.useLocalTimeStamp = true #积攒多少个 Event 才 flush 到 HDFS 一次 a3.sinks.k3.hdfs.batchSize = 100

#设置文件类型，可支持压缩

a3.sinks.k3.hdfs.fileType = DataStream #多久生成一个新的文件 a3.sinks.k3.hdfs.rollInterval = 60

#设置每个文件的滚动大小大概是 128M

a3.sinks.k3.hdfs.rollSize = 134217700 #文件的滚动与Event 数量无关 a3.sinks.k3.hdfs.rollCount = 0

# Use a channel which buffers events in memory a3.channels.c3.type = memory a3.channels.c3.capacity = 1000

a3.channels.c3.transactionCapacity = 100

# Bind the source and sink to the channel a3.sources.r3.channels = c3 a3.sinks.k3.channel = c3

* 1. 启动监控文件夹命令

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a3 --conf-file job/flume-taildir-hdfs.conf

* 1. 向 files 文件夹中追加内容

在/opt/module/flume 目录下创建 files 文件夹

[atguigu@hadoop102 flume]$ mkdir files

向 upload 文件夹中添加文件

[atguigu@hadoop102 files]$ echo hello >> file1.txt [atguigu@hadoop102 files]$ echo atguigu >> file2.txt

* 1. 查看 HDFS 上的数据

Taildir 说明：

Taildir Source 维护了一个 json 格式的 position File，其会定期的往 position File

中更新每个文件读取到的最新的位置，因此能够实现断点续传。Position File 的格式如下：

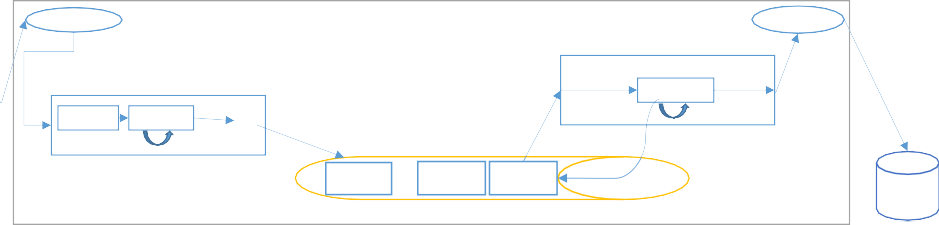
{"inode":2496272,"pos":12,"file":"/opt/module/flume/files/file1.t xt"}

{"inode":2496275,"pos":12,"file":"/opt/module/flume/files/file2.t xt"}

注：Linux 中储存文件元数据的区域就叫做 inode，每个 inode 都有一个号码，操作系统用 inode 号码来识别不同的文件，Unix/Linux 系统内部不使用文件名，而使用 inode 号码来识别文件。

第 **3** 章 **Flume** 进阶

# Flume 事务



Flume事务

数据输入端

Source

batch data

接收事件

Flume流式处理

Agent

数据输出端

Sink

Transaction

doTake

Transaction

转发事件

takeList

Web Server

推送事件

拉取事件

doPut

putList

batch data

doCommit

doCommit

Channel

doRollback

doRollback

…

Eventn Event2 Event1

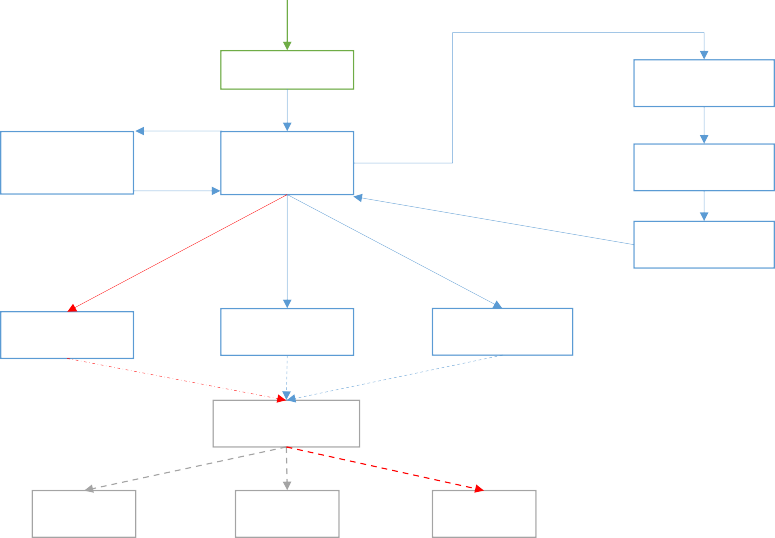
HDFS

Put事务流程

* doPut:将批数据先写入临时缓冲区putList
* doCommit:检查channel内存队列是否足够合并。
* doRollback:channel内存队列空间不足，回滚数据

Take事务

* doTake:将数据取到临时缓冲区takeList，并将数据发送到HDFS
* doCommit:如果数据全部发送成功，则清除临时缓冲区takeList
* doRollback:数据发送过程中如果出现异常，rollback将临时缓冲区t akeList中的数据归还给channel内存队列。
  1. **Flume Agent 内部原理**



Flume Agent内部原理

1 接收数据

Source

3 将事件传递给拦截器链

Interceptor

Channel Selectors 有

两 种 类 型 :Replicating Channel Selector (default)和

4 将每个事件给Channel选择器 2 处理事件

Channel Selector

5 返回写入事件Channel列表

Channel Processor

Interceptor

Multiplexing Channel Selector 。Replicating 会将 source过来的events发往所有channel, 而Multiplexing可

以配置发往哪些Channel。

6 根据Channel选择器的选择结果，将事件写入相应Channel。

Interceptor

Channel1 Channel2 Channel3

SinkProcessor

7 SinkProcessor有三种： DefaultSinkProcessor、

LoadBalancingSinkProcessor、 FailoverSinkProcessor

每种都有其各自的功能

Sink1

Sink2

Sink3

重要组件：

1. ChannelSelector

ChannelSelector 的作用就是选出 Event 将要被发往哪个 Channel。其共有两种类型，分别是 Replicating（复制）和 Multiplexing（多路复用）。

ReplicatingSelector 会将同一个 Event 发往所有的 Channel，Multiplexing 会根据相应的原则，将不同的 Event 发往不同的 Channel。

1. SinkProcessor

SinkProcessor 共 有 三 种 类 型 ， 分 别 是 DefaultSinkProcessor 、

LoadBalancingSinkProcessor 和 FailoverSinkProcessor

DefaultSinkProcessor 对应的是单个的 Sink ， LoadBalancingSinkProcessor 和 FailoverSinkProcessor 对应的是 Sink Group，LoadBalancingSinkProcessor 可以实现负载均衡的功能，FailoverSinkProcessor 可以错误恢复的功能。

# Flume 拓扑结构

## 简单串联

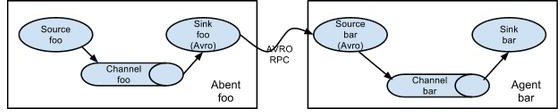


图 Flume Agent 连接

这种模式是将多个 flume 顺序连接起来了，从最初的 source 开始到最终 sink 传送的目的存储系统。此模式不建议桥接过多的 flume 数量， flume 数量过多不仅会影响传输速率，而且一旦传输过程中某个节点 flume 宕机，会影响整个传输系统。

## 复制和多路复用

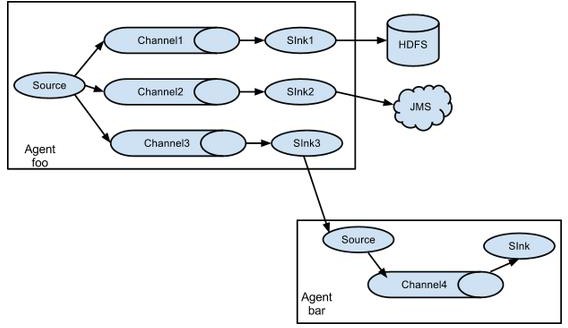


图 单 source，多 channel、sink

Flume 支持将事件流向一个或者多个目的地。这种模式可以将相同数据复制到多个 channel 中，或者将不同数据分发到不同的 channel 中，sink 可以选择传送到不同的目的地。

## 负载均衡和故障转移

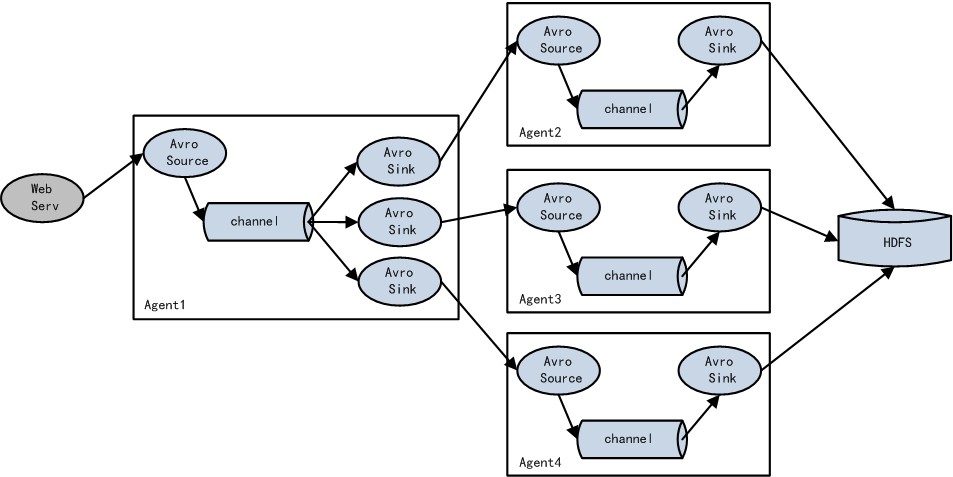


图 Flume 负载均衡或故障转移

Flume 支持使用将多个sink 逻辑上分到一个sink 组，sink 组配合不同的SinkProcessor

可以实现负载均衡和错误恢复的功能。

## 聚合

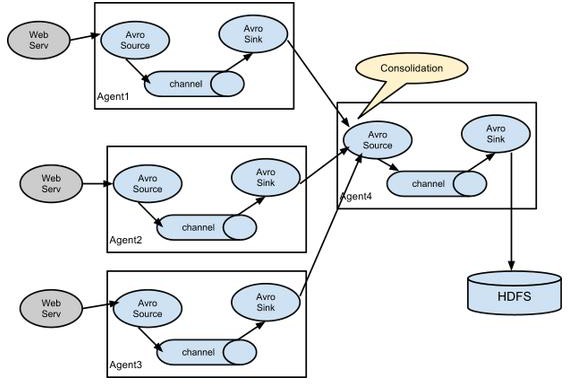


图 Flume Agent 聚合

这种模式是我们最常见的，也非常实用，日常 web 应用通常分布在上百个服务器，大者甚至上千个、上万个服务器。产生的日志，处理起来也非常麻烦。用 flume 的这种组合方式能很好的解决这一问题，每台服务器部署一个 flume 采集日志，传送到一个集中收集日志的

flume，再由此 flume 上传到 hdfs、hive、hbase 等，进行日志分析。

# Flume 企业开发案例

## 复制和多路复用

1. 案例需求

使用 Flume-1 监控文件变动，Flume-1 将变动内容传递给 Flume-2，Flume-2 负责存储到 HDFS。同时 Flume-1 将变动内容传递给 Flume-3，Flume-3 负责输出到 Local FileSystem。

1. 需求分析：



单数据源多出口案例（选择器）

Flume-1

Flume-2

Hive实时更新日志

Exec Source

Avro Sink1

Flume-file-flume监控文件 Memory Channel1 Memory Channel2

Flume-3

Avro Sink2

Avro Source File\_roll Sink flume-flume-dir

Memory Channel

Replicating ChannelSelector

本地目录：

/opt/module/datas/flume3

Hive日志文件

/opt/module/hive/logs

/hive.log

HDFS

Avro Source HDFS Sink flume-flume-hdfs

Memory Channel

1. 实现步骤：
   1. 准备工作

在/opt/module/flume/job 目录下创建 group1 文件夹

[atguigu@hadoop102 job]$ cd group1/

在/opt/module/datas/目录下创建 flume3 文件夹

[atguigu@hadoop102 datas]$ mkdir flume3

* 1. 创建 flume-file-flume.conf

配置 1 个接收日志文件的source 和两个 channel、两个 sink，分别输送给 flume-flume- hdfs 和 flume-flume-dir。

编辑配置文件

[atguigu@hadoop102 group1]$ vim flume-file-flume.conf

添加如下内容

# Name the components on this agent a1.sources = r1

a1.sinks = k1 k2 a1.channels = c1 c2

# 将数据流复制给所有 channel

a1.sources.r1.selector.type = replicating

# Describe/configure the source a1.sources.r1.type = exec

a1.sources.r1.command = tail -F /opt/module/hive/logs/hive.log a1.sources.r1.shell = /bin/bash -c

# Describe the sink

# sink 端的 avro 是一个数据发送者

a1.sinks.k1.type = avro a1.sinks.k1.hostname = hadoop102 a1.sinks.k1.port = 4141

a1.sinks.k2.type = avro a1.sinks.k2.hostname = hadoop102 a1.sinks.k2.port = 4142

# Describe the channel a1.channels.c1.type = memory a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.channels.c2.type = memory a1.channels.c2.capacity = 1000

a1.channels.c2.transactionCapacity = 100

# Bind the source and sink to the channel a1.sources.r1.channels = c1 c2 a1.sinks.k1.channel = c1 a1.sinks.k2.channel = c2

* 1. 创建 flume-flume-hdfs.conf

配置上级 Flume 输出的 Source，输出是到 HDFS 的Sink。编辑配置文件

[atguigu@hadoop102 group1]$ vim flume-flume-hdfs.conf

添加如下内容

# Name the components on this agent a2.sources = r1

a2.sinks = k1 a2.channels = c1

# Describe/configure the source # source 端的 avro 是一个数据接收服务

a2.sources.r1.type = avro a2.sources.r1.bind = hadoop102 a2.sources.r1.port = 4141

# Describe the sink a2.sinks.k1.type = hdfs

a2.sinks.k1.hdfs.path = hdfs://hadoop102:9820/flume2/%Y%m%d/%H

#上传文件的前缀

a2.sinks.k1.hdfs.filePrefix = flume2- #是否按照时间滚动文件夹 a2.sinks.k1.hdfs.round = true

#多少时间单位创建一个新的文件夹

a2.sinks.k1.hdfs.roundValue = 1 #重新定义时间单位 a2.sinks.k1.hdfs.roundUnit = hour #是否使用本地时间戳

a2.sinks.k1.hdfs.useLocalTimeStamp = true #积攒多少个 Event 才 flush 到 HDFS 一次 a2.sinks.k1.hdfs.batchSize = 100

#设置文件类型，可支持压缩

a2.sinks.k1.hdfs.fileType = DataStream #多久生成一个新的文件 a2.sinks.k1.hdfs.rollInterval = 30

#设置每个文件的滚动大小大概是 128M

a2.sinks.k1.hdfs.rollSize = 134217700 #文件的滚动与Event 数量无关 a2.sinks.k1.hdfs.rollCount = 0

# Describe the channel a2.channels.c1.type = memory a2.channels.c1.capacity = 1000

a2.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel a2.sources.r1.channels = c1 a2.sinks.k1.channel = c1

* 1. 创建 flume-flume-dir.conf

配置上级 Flume 输出的 Source，输出是到本地目录的 Sink。编辑配置文件

[atguigu@hadoop102 group1]$ vim flume-flume-dir.conf

添加如下内容

# Name the components on this agent a3.sources = r1

a3.sinks = k1 a3.channels = c2

# Describe/configure the source a3.sources.r1.type = avro a3.sources.r1.bind = hadoop102 a3.sources.r1.port = 4142

# Describe the sink a3.sinks.k1.type = file\_roll

a3.sinks.k1.sink.directory = /opt/module/data/flume3

# Describe the channel a3.channels.c2.type = memory a3.channels.c2.capacity = 1000

a3.channels.c2.transactionCapacity = 100

### 尚硅谷大数据技术之 Flume

—————————————————————————————

# Bind the source and sink to the channel a3.sources.r1.channels = c2 a3.sinks.k1.channel = c2

提示：输出的本地目录必须是已经存在的目录，如果该目录不存在，并不会创建新的目录。

* 1. 执行配置文件

分别启动对应的 flume 进程：flume-flume-dir，flume-flume-hdfs，flume-file-flume。

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a3 --conf-file job/group1/flume-flume-dir.conf

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a2 --conf-file job/group1/flume-flume-hdfs.conf

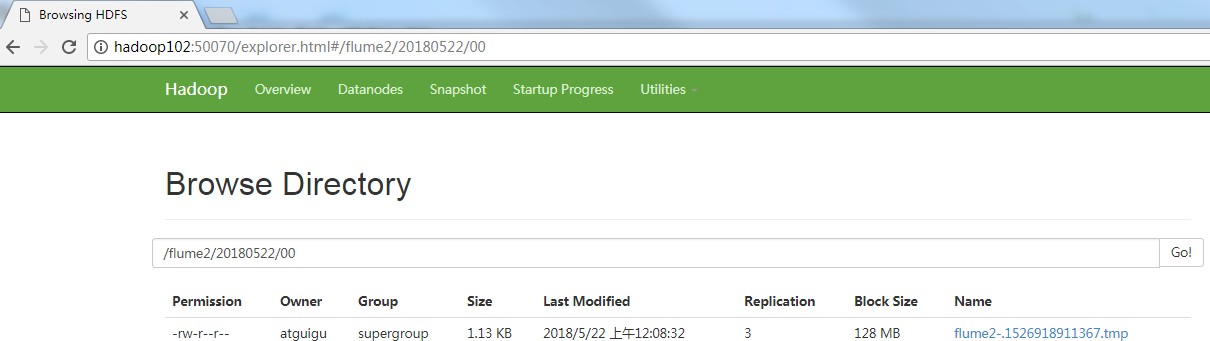
[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a1 --conf-file job/group1/flume-file-flume.conf

* 1. 启动Hadoop 和 Hive

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/start-dfs.sh [atguigu@hadoop103 hadoop-2.7.2]$ sbin/start-yarn.sh

[atguigu@hadoop102 hive]$ bin/hive hive (default)>

* 1. 检查 HDFS 上数据



* 1. 检查/opt/module/datas/flume3 目录中数据

[atguigu@hadoop102 flume3]$ ll

总用量 8

-rw-rw-r--. 1 atguigu atguigu 5942 5 月 22 00:09 1526918887550-3

## 负载均衡和故障转移

1. 案例需求

使用 Flume1 监控一个端口，其 sink 组中的 sink 分别对接 Flume2 和 Flume3，采用 FailoverSinkProcessor，实现故障转移的功能。

1. 需求分析



故障转移案例

Flume-1

Flume-2

Avro Source logger Sink

nc localhost 44444

flume-flume-console1

netcat Source

Avro Sink1

Memory Channel

Memory Channel flume-file-flume监控文件

Flume-3

Avro Sink2

FailoverSinkProcessor

Hello Atguigu Hive

Avro Source logger Sink flume-flume-console2

Memory Channel

控制台

1. 实现步骤
   1. 准备工作

在/opt/module/flume/job 目录下创建 group2 文件夹

[atguigu@hadoop102 job]$ cd group2/

* 1. 创建 flume-netcat-flume.conf

配置 1 个 netcat source 和 1 个 channel、1 个 sink group（2 个 sink），分别输送给 flume-flume-console1 和 flume-flume-console2。

编辑配置文件

[atguigu@hadoop102 group2]$ vim flume-netcat-flume.conf

添加如下内容

# Name the components on this agent a1.sources = r1

a1.channels = c1 a1.sinkgroups = g1 a1.sinks = k1 k2

# Describe/configure the source a1.sources.r1.type = netcat a1.sources.r1.bind = localhost a1.sources.r1.port = 44444

a1.sinkgroups.g1.processor.type = failover a1.sinkgroups.g1.processor.priority.k1 = 5

a1.sinkgroups.g1.processor.priority.k2 = 10

a1.sinkgroups.g1.processor.maxpenalty = 10000

# Describe the sink a1.sinks.k1.type = avro a1.sinks.k1.hostname = hadoop102 a1.sinks.k1.port = 4141

a1.sinks.k2.type = avro a1.sinks.k2.hostname = hadoop102 a1.sinks.k2.port = 4142

# Describe the channel 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.sinkgroups.g1.sinks = k1 k2 a1.sinks.k1.channel = c1 a1.sinks.k2.channel = c1

* 1. 创建 flume-flume-console1.conf

配置上级 Flume 输出的 Source，输出是到本地控制台。编辑配置文件

[atguigu@hadoop102 group2]$ vim flume-flume-console1.conf

添加如下内容

# Name the components on this agent a2.sources = r1

a2.sinks = k1 a2.channels = c1

# Describe/configure the source a2.sources.r1.type = avro a2.sources.r1.bind = hadoop102 a2.sources.r1.port = 4141

# Describe the sink a2.sinks.k1.type = logger

# Describe the channel a2.channels.c1.type = memory a2.channels.c1.capacity = 1000

a2.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel a2.sources.r1.channels = c1 a2.sinks.k1.channel = c1

* 1. 创建 flume-flume-console2.conf

配置上级 Flume 输出的 Source，输出是到本地控制台。编辑配置文件

[atguigu@hadoop102 group2]$ vim flume-flume-console2.conf

添加如下内容

# Name the components on this agent a3.sources = r1

a3.sinks = k1 a3.channels = c2

# Describe/configure the source a3.sources.r1.type = avro a3.sources.r1.bind = hadoop102 a3.sources.r1.port = 4142

# Describe the sink a3.sinks.k1.type = logger

# Describe the channel a3.channels.c2.type = memory a3.channels.c2.capacity = 1000

a3.channels.c2.transactionCapacity = 100

# Bind the source and sink to the channel a3.sources.r1.channels = c2 a3.sinks.k1.channel = c2

* 1. 执行配置文件

分别开启对应配置文件： flume-flume-console2 ， flume-flume-console1 ， flume- netcat-flume。

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a3 --conf-file job/group2/flume-flume-console2.conf -

Dflume.root.logger=INFO,console

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a2 --conf-file job/group2/flume-flume-console1.conf -

Dflume.root.logger=INFO,console

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a1 --conf-file job/group2/flume-netcat-flume.conf

* 1. 使用 netcat 工具向本机的 44444 端口发送内容

$ nc localhost 44444

* 1. 查看Flume2 及 Flume3 的控制台打印日志
  2. 将 Flume2 kill，观察 Flume3 的控制台打印情况。注：使用 jps -ml 查看 Flume 进程。

## 聚合

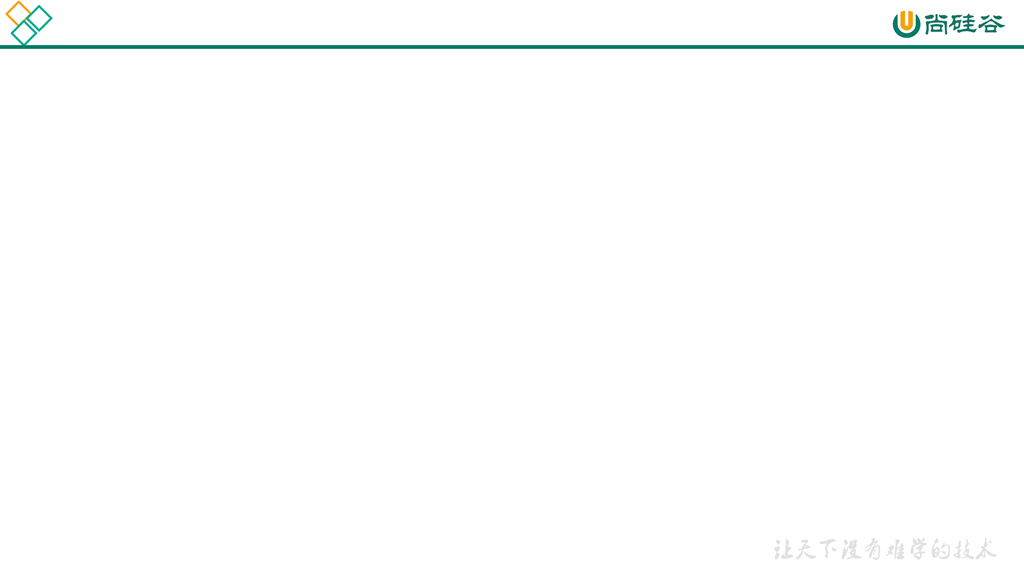
1. 案例需求：

hadoop102 上的Flume-1 监控文件/opt/module/group.log，

hadoop103 上的Flume-2 监控某一个端口的数据流，

Flume-1 与 Flume-2 将数据发送给 hadoop104 上的 Flume-3，Flume-3 将最终数据打印到控制台。

1. 需求分析



Exec Source

Avro Sink

Flume1配置文件

Memory Channel

Netcat Source

Avro Sink

Flume2配置文件

Memory Channel

多数据源汇总案例

Flume-1

实时更新日志

Flume-3

Avro Source logger Sink

Flume-2

Flume3配置文件

Memory Channel

通过telnet向44444端

口发送数据

主机44444端口

控制台

日志文件

/opt/module/group.log

1. 实现步骤：
   1. 准备工作分发 Flume

[atguigu@hadoop102 module]$ xsync flume

在 hadoop102、hadoop103 以及 hadoop104 的/opt/module/flume/job 目录下创建一个

group3 文件夹。

[atguigu@hadoop102 job]$ mkdir group3 [atguigu@hadoop103 job]$ mkdir group3 [atguigu@hadoop104 job]$ mkdir group3

* 1. 创建 flume1-logger-flume.conf

配置 Source 用于监控 hive.log 文件，配置 Sink 输出数据到下一级Flume。在 hadoop102 上编辑配置文件

[atguigu@hadoop102 group3]$ vim flume1-logger-flume.conf

添加如下内容

# Name the components on this agent a1.sources = r1

a1.sinks = k1 a1.channels = c1

# Describe/configure the source a1.sources.r1.type = exec

a1.sources.r1.command = tail -F /opt/module/group.log a1.sources.r1.shell = /bin/bash -c

# Describe the sink a1.sinks.k1.type = avro a1.sinks.k1.hostname = hadoop104 a1.sinks.k1.port = 4141

# Describe the channel 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

* 1. 创建 flume2-netcat-flume.conf

配置 Source 监控端口 44444 数据流，配置Sink 数据到下一级Flume：在 hadoop103 上编辑配置文件

[atguigu@hadoop102 group3]$ vim flume2-netcat-flume.conf

添加如下内容

# Name the components on this agent a2.sources = r1

a2.sinks = k1 a2.channels = c1

# Describe/configure the source a2.sources.r1.type = netcat a2.sources.r1.bind = hadoop103 a2.sources.r1.port = 44444

# Describe the sink a2.sinks.k1.type = avro a2.sinks.k1.hostname = hadoop104 a2.sinks.k1.port = 4141

# Use a channel which buffers events in memory a2.channels.c1.type = memory a2.channels.c1.capacity = 1000

a2.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel a2.sources.r1.channels = c1 a2.sinks.k1.channel = c1

* 1. 创建 flume3-flume-logger.conf

配置 source 用于接收 flume1 与 flume2 发送过来的数据流，最终合并后 sink 到控制台。

在 hadoop104 上编辑配置文件

[atguigu@hadoop104 group3]$ touch flume3-flume-logger.conf [atguigu@hadoop104 group3]$ vim flume3-flume-logger.conf

添加如下内容

# Name the components on this agent a3.sources = r1

a3.sinks = k1 a3.channels = c1

# Describe/configure the source a3.sources.r1.type = avro

a3.sources.r1.bind = hadoop104 a3.sources.r1.port = 4141

# Describe the sink # Describe the sink

a3.sinks.k1.type = logger

# Describe the channel a3.channels.c1.type = memory a3.channels.c1.capacity = 1000

a3.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel a3.sources.r1.channels = c1 a3.sinks.k1.channel = c1

* 1. 执行配置文件

分别开启对应配置文件：flume3-flume-logger.conf，flume2-netcat-flume.conf，

flume1-logger-flume.conf。

[atguigu@hadoop104 flume]$ bin/flume-ng agent --conf conf/ --name a3 --conf-file job/group3/flume3-flume-logger.conf -

Dflume.root.logger=INFO,console

[atguigu@hadoop102 flume]$ bin/flume-ng agent --conf conf/ --name a2 --conf-file job/group3/flume1-logger-flume.conf

[atguigu@hadoop103 flume]$ bin/flume-ng agent --conf conf/ --name a1 --conf-file job/group3/flume2-netcat-flume.conf

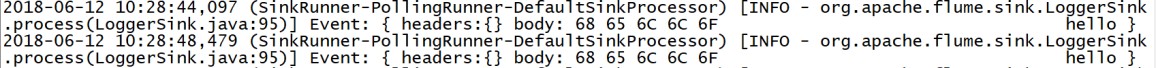
* 1. 在 hadoop103 上向/opt/module 目录下的 group.log 追加内容

[atguigu@hadoop103 module]$ echo 'hello' > group.log

* 1. 在 hadoop102 上向 44444 端口发送数据

[atguigu@hadoop102 flume]$ telnet hadoop102 44444

* 1. 检查 hadoop104 上数据



# 自定义Interceptor

1. 案例需求

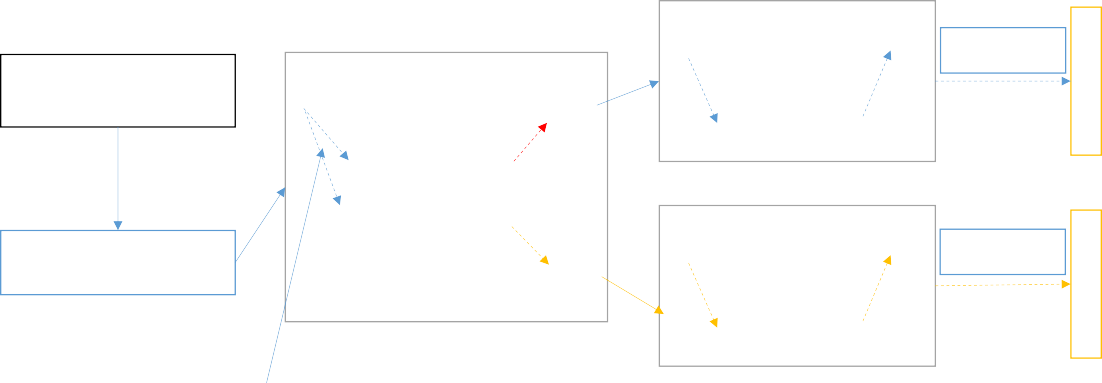
使用 Flume 采集服务器本地日志，需要按照日志类型的不同，将不同种类的日志发往不同的分析系统。

1. 需求分析

在实际的开发中，一台服务器产生的日志类型可能有很多种，不同类型的日志可能需要发送到不同的分析系统。此时会用到 Flume 拓扑结构中的 Multiplexing 结构，Multiplexing的原理是，根据 event 中 Header 的某个 key 的值，将不同的 event 发送到不同的 Channel

中，所以我们需要自定义一个 Interceptor，为不同类型的 event 的 Header 中的 key 赋予不同的值。

在该案例中，我们以端口数据模拟日志，以是否包含”atguigu”模拟不同类型的日志，我们需要自定义 interceptor 区分数据中是否包含”atguigu”，将其分别发往不同的分析系统（Channel）。



Interceptor和Multiplexing ChannelSelector案例

Flume-2

Flume-1

Avro Source logger Sink

atguigu

控制台实时输入数据

flume-flume-console1

netcat Source

Avro Sink1

控制台

Memory Channel

Memory Channel

Memory Channel

Flume-3

atguigu

flume

Avro Source logger Sink

flume-flume-console2

flume

Avro Sink2

控制台

Memory Channel

Multiplexing Channel Selector

1. 实现步骤
   1. 创建一个 maven 项目，并引入以下依赖。

<dependency>

<groupId>org.apache.flume</groupId>

<artifactId>flume-ng-core</artifactId>

<version>1.9.0</version>

</dependency>

* 1. 定义 CustomInterceptor 类并实现 Interceptor 接口。

package com.atguigu.interceptor;

import org.apache.flume.Context; import org.apache.flume.Event;

import org.apache.flume.interceptor.Interceptor;

import java.util.ArrayList; import java.util.List; import java.util.Map;

public class TypeInterceptor implements Interceptor {

//声明一个存放事件的集合

private List<Event> addHeaderEvents;

@Override

### 尚硅谷大数据技术之 Flume

—————————————————————————————

public void initialize() {

//初始化存放事件的集合

addHeaderEvents = new ArrayList<>();

}

//单个事件拦截 @Override

public Event intercept(Event event) {

//1.获取事件中的头信息

Map<String, String> headers = event.getHeaders();

//2.获取事件中的 body 信息

String body = new String(event.getBody());

//3.根据 body 中是否有"atguigu"来决定添加怎样的头信息

if (body.contains("atguigu")) {

//4.添加头信息

headers.put("type", "first");

} else {

//4.添加头信息

headers.put("type", "second");

}

return event;

}

//批量事件拦截 @Override

public List<Event> intercept(List<Event> events) {

//1.清空集合 addHeaderEvents.clear();

//2.遍历 events

for (Event event : events) {

//3.给每一个事件添加头信息

addHeaderEvents.add(intercept(event));

}

//4.返回结果

return addHeaderEvents;

}

@Override

public void close() {

}

public static class Builder implements Interceptor.Builder { @Override

public Interceptor build() { return new TypeInterceptor();

}

@Override

public void configure(Context context) {

}

}

}

* 1. 编辑 flume 配置文件

为 hadoop102 上的 Flume1 配置 1 个 netcat source，1 个 sink grou（p 2 个 avro sink），

并配置相应的ChannelSelector 和 interceptor。

# Name the components on this agent a1.sources = r1

a1.sinks = k1 k2 a1.channels = c1 c2

# Describe/configure the source a1.sources.r1.type = netcat a1.sources.r1.bind = localhost a1.sources.r1.port = 44444 a1.sources.r1.interceptors = i1 a1.sources.r1.interceptors.i1.type =

com.atguigu.flume.interceptor.CustomInterceptor$Builder a1.sources.r1.selector.type = multiplexing a1.sources.r1.selector.header = type a1.sources.r1.selector.mapping.first = c1 a1.sources.r1.selector.mapping.second = c2

# Describe the sink a1.sinks.k1.type = avro a1.sinks.k1.hostname = hadoop103 a1.sinks.k1.port = 4141

a1.sinks.k2.type=avro a1.sinks.k2.hostname = hadoop104 a1.sinks.k2.port = 4242

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

a1.channels.c1.transactionCapacity = 100

# Use a channel which buffers events in memory a1.channels.c2.type = memory a1.channels.c2.capacity = 1000

a1.channels.c2.transactionCapacity = 100

# Bind the source and sink to the channel a1.sources.r1.channels = c1 c2 a1.sinks.k1.channel = c1 a1.sinks.k2.channel = c2

为 hadoop103 上的 Flume4 配置一个 avro source 和一个 logger sink。

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

a1.sources.r1.type = avro

a1.sources.r1.bind = hadoop103 a1.sources.r1.port = 4141

a1.sinks.k1.type = logger

a1.channels.c1.type = memory a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

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

为 hadoop104 上的 Flume3 配置一个 avro source 和一个 logger sink。

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

a1.sources.r1.type = avro a1.sources.r1.bind = hadoop104 a1.sources.r1.port = 4242

a1.sinks.k1.type = logger

a1.channels.c1.type = memory a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

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

* 1. 分别在 hadoop102，hadoop103，hadoop104 上启动 flume 进程，注意先后顺序。
  2. 在 hadoop102 使用netcat 向 localhost:44444 发送字母和数字。
  3. 观察 hadoop103 和hadoop104 打印的日志。

# 自定义 Source

1. 介绍

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，包括 avro、thrift、exec、jms、spooling directory、netcat、sequence generator、syslog、http、legacy。官方提供的 source 类型已经很多，但是有时候并不能满足实际开发当中的需求，此时我们就需要根据实际需求自定义某些 source。

官方也提供了自定义 source 的接口：

<https://flume.apache.org/FlumeDeveloperGuide.html#source>根据官方说明自定义

MySource 需要继承 AbstractSource 类并实现 Configurable 和 PollableSource 接口。实现相应方法：

getBackOffSleepIncrement() //backoff 步长

getMaxBackOffSleepInterval()//backoff 最长时间

configure(Context context)//初始化 context（读取配置文件内容） process()//获取数据封装成 event 并写入 channel，这个方法将被循环调用。使用场景：读取 MySQL 数据或者其他文件系统。

1. 需求

使用 flume 接收数据，并给每条数据添加前缀，输出到控制台。前缀可从 flume 配置文件中配置。



自定义Source需求

MySource：代码中循环生产数据

1 编写自定义Source代码并打包到集群

1. 创建符合条件的flume配置文件
2. 执行配置文件，开启监控

4 查看控制台数据

Console

Logger Sink

Memory Channel

1. 分析



自定义Source需求分析

AbstractSource

Configurable

PollableSource

MySource

configure(Context context)：读取配置文件(XX.conf)中的配置信息 process()：接收数据，将数据封装成一个个的Event，写入Channel。使用for循环模拟数据生成。for(int i=0;i<5;i++) getBackOffSleepIncrement()：暂不用

getMaxBackOffSleepInterval()：暂不用

1. 编码
   1. 导入 pom 依赖

<dependencies>

<dependency>

<groupId>org.apache.flume</groupId>

<artifactId>flume-ng-core</artifactId>

<version>1.9.0</version>

</dependency>

* 1. 编写代码

package com.atguigu;

import org.apache.flume.Context;

import org.apache.flume.EventDeliveryException; import org.apache.flume.PollableSource;

import org.apache.flume.conf.Configurable; import org.apache.flume.event.SimpleEvent; import org.apache.flume.source.AbstractSource;

import java.util.HashMap;

public class MySource extends AbstractSource implements Configurable, PollableSource {

//定义配置文件将来要读取的字段 private Long delay; private String field;

//初始化配置信息 @Override

public void configure(Context context) { delay = context.getLong("delay");

field = context.getString("field", "Hello!");

}

@Override

public Status process() throws EventDeliveryException {

try {

//创建事件头信息

HashMap<String, String> hearderMap = new HashMap<>();

//创建事件

SimpleEvent event = new SimpleEvent();

//循环封装事件

for (int i = 0; i < 5; i++) {

//给事件设置头信息

event.setHeaders(hearderMap);

//给事件设置内容

event.setBody((field + i).getBytes());

//将事件写入 channel

getChannelProcessor().processEvent(event); Thread.sleep(delay);

}

} catch (Exception e) { e.printStackTrace();

return Status.BACKOFF;

}

return Status.READY;

}

@Override

public long getBackOffSleepIncrement() { return 0;

}

@Override

public long getMaxBackOffSleepInterval() { return 0;

}

}

1. 测试
2. 打包

将写好的代码打包，并放到 flume 的 lib 目录（/opt/module/flume）下。

1. 配置文件

# Name the components on this agent a1.sources = r1

a1.sinks = k1 a1.channels = c1

# Describe/configure the source a1.sources.r1.type = com.atguigu.MySource a1.sources.r1.delay = 1000 #a1.sources.r1.field = atguigu

# Describe the sink a1.sinks.k1.type = logger

# 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

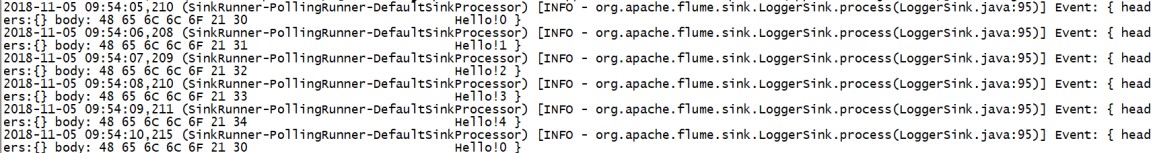
1. 开启任务

[atguigu@hadoop102 flume]$ pwd

/opt/module/flume

[atguigu@hadoop102 flume]$ bin/flume-ng agent -c conf/ -f job/mysource.conf -n a1 -Dflume.root.logger=INFO,console

1. 结果展示



# 自定义 Sink

1. 介绍

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 组件目的地包括 hdfs、logger、avro、thrift、ipc、file、null、HBase、solr、自定义。官方提供的 Sink 类型已经很多，但是有时候并不能满足实际开发当中的需求，此时我们就需要根据实际需求自定义某些 Sink。

官方也提供了自定义 sink 的接口：

https://flume.apache.org/FlumeDeveloperGuide.html#sink 根据官方说明自定义

MySink 需要继承 AbstractSink 类并实现 Configurable 接口。实现相应方法：

configure(Context context)//初始化 context（读取配置文件内容） process()//从 Channel 读取获取数据（event），这个方法将被循环调用。使用场景：读取 Channel 数据写入 MySQL 或者其他文件系统。

1. 需求

使用 flume 接收数据，并在 Sink 端给每条数据添加前缀和后缀，输出到控制台。前后缀可在 flume 任务配置文件中配置。

流程分析：



1.编码

AbstractSink

Configurable

configure():读取任务配置文件中的配置信息。

MySink

process():从Channel中取数据，添加前后缀，写入日志。

2.打包到集群并编写任务配置文件

3.数据流

source

控制台输入：

atguigu

channel

sink

atguigu

hello:atguigu:hello

1. 编码

package com.atguigu;

import org.apache.flume.\*;

import org.apache.flume.conf.Configurable; import org.apache.flume.sink.AbstractSink; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class MySink extends AbstractSink implements Configurable

{

//创建 Logger 对象

private static final Logger LOG = LoggerFactory.getLogger(AbstractSink.class);

private String prefix; private String suffix;

@Override

public Status process() throws EventDeliveryException {

//声明返回值状态信息 Status status;

//获取当前 Sink 绑定的 Channel Channel ch = getChannel();

//获取事务

Transaction txn = ch.getTransaction();

//声明事件 Event event;

//开启事务 txn.begin();

1. 测试

//读取 Channel 中的事件，直到读取到事件结束循环

while (true) {

event = ch.take(); if (event != null) {

break;

}

}

try {

//处理事件（打印）

LOG.info(prefix + new String(event.getBody()) +

suffix);

//事务提交 txn.commit();

status = Status.READY;

} catch (Exception e) {

//遇到异常，事务回滚 txn.rollback();

status = Status.BACKOFF;

} finally {

//关闭事务 txn.close();

}

return status;

}

@Override

public void configure(Context context) {

//读取配置文件内容，有默认值

prefix = context.getString("prefix", "hello:");

//读取配置文件内容，无默认值

suffix = context.getString("suffix");

}

}

* 1. 打包

将写好的代码打包，并放到 flume 的 lib 目录（/opt/module/flume）下。

* 1. 配置文件

# Name the components on this agent a1.sources = r1

a1.sinks = k1 a1.channels = c1

# Describe/configure the source a1.sources.r1.type = netcat a1.sources.r1.bind = localhost a1.sources.r1.port = 44444

# Describe the sink

a1.sinks.k1.type = com.atguigu.MySink #a1.sinks.k1.prefix = atguigu: a1.sinks.k1.suffix = :atguigu

# 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

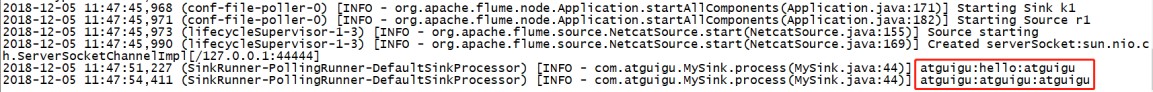
* 1. 开启任务

[atguigu@hadoop102 flume]$ bin/flume-ng agent -c conf/ -f job/mysink.conf -n a1 -Dflume.root.logger=INFO,console [atguigu@hadoop102 ~]$ nc localhost 44444

hello OK

atguigu OK

* 1. 结果展示



# Flume 数据流监控

* + 1. **Ganglia** 的安装与部署

Ganglia 由 gmond、gmetad 和 gweb 三部分组成。

gmond（Ganglia Monitoring Daemon）是一种轻量级服务，安装在每台需要收集指标数据的节点主机上。使用 gmond，你可以很容易收集很多系统指标数据，如 CPU、内存、磁盘、网络和活跃进程的数据等。

gmetad（Ganglia Meta Daemon）整合所有信息，并将其以 RRD 格式存储至磁盘的服务。

gweb（Ganglia Web）Ganglia 可视化工具，gweb 是一种利用浏览器显示gmetad 所存储数据的 PHP 前端。在 Web 界面中以图表方式展现集群的运行状态下收集的多种不同指标数据。

1. 安装 ganglia
   1. 规划

hadoop102: hadoop103: hadoop104:

web gmetad gmod gmod

gmod

* 1. 在 102 103 104 分别安装 epel-release

[atguigu@hadoop102 flume]$ sudo yum -y install epel-release

* 1. 在 102 安装

[atguigu@hadoop102 flume]$ sudo yum -y install ganglia-gmetad [atguigu@hadoop102 flume]$ sudo yum -y install ganglia-web [atguigu@hadoop102 flume]$ sudo yum -y install ganglia-gmond

（4）在 103 和 104 安装

[atguigu@hadoop102 flume]$ sudo yum -y install ganglia-gmond

1. 在 102 修改配置文件/etc/httpd/conf.d/ganglia.conf

[atguigu@hadoop102 flume]$ sudo vim

/etc/httpd/conf.d/ganglia.conf

修改为红颜色的配置：

# Ganglia monitoring system php web frontend #

Alias /ganglia /usr/share/ganglia

<Location /ganglia> # Require local

# 通过 windows 访问 ganglia,需要配置 Linux 对应的主机(windows)ip 地址

Require ip 192.168.9.1 # Require ip 10.1.2.3

# Require host example.org

</Location>

1. 在 102 修改配置文件/etc/ganglia/gmetad.conf

[atguigu@hadoop102 flume]$ sudo vim /etc/ganglia/gmetad.conf

修改为：

data\_source "my cluster" hadoop102

1. 在 102 103 104 修改配置文件/etc/ganglia/gmond.conf

[atguigu@hadoop102 flume]$ sudo vim /etc/ganglia/gmond.conf

修改为：

cluster {

name = "my cluster" owner = "unspecified" latlong = "unspecified" url = "unspecified"

}

udp\_send\_channel {

#bind\_hostname = yes # Highly recommended, soon to be default. # This option tells gmond to use a source

address

# that resolves to the machine's hostname.

Without

# this, the metrics may appear to come from

any

# interface and the DNS names associated with # those IPs will be used to create the RRDs.

# mcast\_join = 239.2.11.71 # 数据发送给 hadoop102

host = hadoop102 port = 8649

ttl = 1

}

udp\_recv\_channel {

# mcast\_join = 239.2.11.71 port = 8649

# 接收来自任意连接的数据

bind = 0.0.0.0 retry\_bind = true

# Size of the UDP buffer. If you are handling lots of metrics you really

# should bump it up to e.g. 10MB or even higher. # buffer = 10485760

}

1. 在 102 修改配置文件/etc/selinux/config

[atguigu@hadoop102 flume]$ sudo vim /etc/selinux/config

修改为：

# This file controls the state of SELinux on the system. # SELINUX= can take one of these three values:

# enforcing - SELinux security policy is enforced.

# permissive - SELinux prints warnings instead of enforcing. # disabled - No SELinux policy is loaded.

SELINUX=disabled

# SELINUXTYPE= can take one of these two values: # targeted - Targeted processes are protected, # mls - Multi Level Security protection.

SELINUXTYPE=targeted

尖叫提示：selinux 生效需要重启，如果此时不想重启，可以临时生效之：

[atguigu@hadoop102 flume]$ sudo setenforce 0

1. 启动 ganglia

（1）在 102 103 104 启动

[atguigu@hadoop102 flume]$ sudo systemctl start gmond

（2）在 102 启动

[atguigu@hadoop102 flume]$ sudo systemctl start httpd [atguigu@hadoop102 flume]$ sudo systemctl start gmetad

1. 打开网页浏览 ganglia 页面

<http://hadoop102/ganglia>

尖叫提示：如果完成以上操作依然出现权限不足错误，请修改/var/lib/ganglia 目录的权限：

[atguigu@hadoop102 flume]$ sudo chmod -R 777 /var/lib/ganglia

* + 1. 操作 **Flume** 测试监控

1. 启动 Flume 任务

[atguigu@hadoop102 flume]$ bin/flume-ng agent \

-c conf/ \

-n a1 \

-f job/flume-netcat-logger.conf \

-Dflume.root.logger=INFO,console \

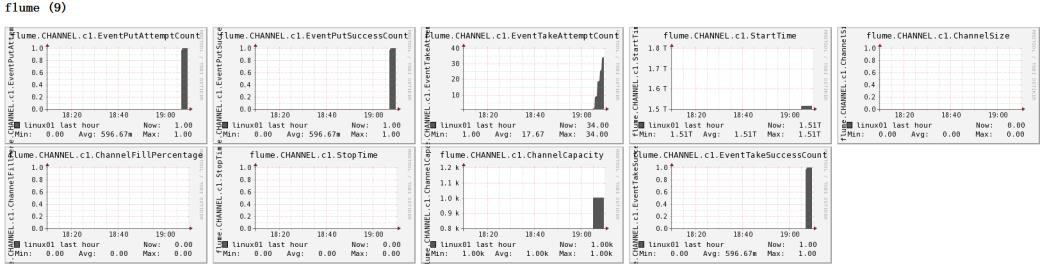
-Dflume.monitoring.type=ganglia \

-Dflume.monitoring.hosts=hadoop102:8649

1. 发送数据观察 ganglia 监测图

[atguigu@hadoop102 flume]$ nc localhost 44444

样式如图：



图例说明：

|  |  |
| --- | --- |
| 字段（图表名称） | 字段含义 |
| EventPutAttemptCount | source 尝试写入 channel 的事件总数量 |
| EventPutSuccessCount | 成功写入 channel 且提交的事件总数量 |
| EventTakeAttemptCount | sink 尝试从 channel 拉取事件的总数量。 |
| EventTakeSuccessCount | sink 成功读取的事件的总数量 |
| StartTime | channel 启动的时间（毫秒） |
| StopTime | channel 停止的时间（毫秒） |
| ChannelSize | 目前 channel 中事件的总数量 |
| ChannelFillPercentage | channel 占用百分比 |
| ChannelCapacity | channel 的容量 |

**第 4 章 企业真实面试题（重点）**

* 1. 你是如何实现 **Flume** 数据传输的监控的

使用第三方框架 Ganglia 实时监控 Flume。

* 1. **Flume** 的 **Source**，**Sink**，**Channel** 的作用？你们 **Source** 是什么类型？

1. 作用
   1. 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
   2. Channel 组件对采集到的数据进行缓存，可以存放在 Memory 或 File 中。
   3. 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、Hbase、solr、自定义。
2. 我公司采用的 Source 类型为：
3. 监控后台日志：exec
4. 监控后台产生日志的端口：netcat

# Flume 的 Channel Selectors



Flume Channel Selectors

数据源1

channel1

数据源2

channel2

Channel Selectors，可以让不同的项目日志通过不同的Channel到不同的Sink中去。

官方文档上Channel Selectors 有两种类型:Replicating Channel Selector (default) 和 Multiplexing Channel Selector

这两种Selector的区别是:Replicating 会将source过来的events发往所有channel, 而

Multiplexing可以选择该发往哪些Channel。

Sink2(Logger)

source

Sink1(Hdfs)

* 1. **Flume** 参数调优

1. Source

增加 Source 个（使用 Tair Dir Source 时可增加 FileGroups 个数）可以增大 Source的读取数据的能力。例如：当某一个目录产生的文件过多时需要将这个文件目录拆分成多个文件目录，同时配置好多个 Source 以保证 Source 有足够的能力获取到新产生的数据。

batchSize 参数决定 Source 一次批量运输到 Channel 的 event 条数，适当调大这个参数可以提高 Source 搬运Event 到 Channel 时的性能。

1. Channel

type 选择 memory 时 Channel 的性能最好，但是如果 Flume 进程意外挂掉可能会丢失数据。type 选择 file 时Channel 的容错性更好，但是性能上会比 memory channel 差。

使用 file Channel 时 dataDirs 配置多个不同盘下的目录可以提高性能。

Capacity 参数决定 Channel 可容纳最大的 event 条数。transactionCapacity 参数决定每次 Source 往 channel 里面写的最大 event 条数和每次 Sink 从 channel 里面读的最大 event 条数。transactionCapacity 需要大于 Source 和 Sink 的 batchSize 参数。 3）Sink

增加 Sink 的个数可以增加 Sink 消费 event 的能力。Sink 也不是越多越好够用就行，

过多的 Sink 会占用系统资源，造成系统资源不必要的浪费。

batchSize 参数决定 Sink 一次批量从 Channel 读取的 event 条数，适当调大这个参数可以提高 Sink 从 Channel 搬出 event 的性能。

* 1. **Flume** 的事务机制

Flume 的事务机制（类似数据库的事务机制）：Flume 使用两个独立的事务分别负责从

Soucrce 到 Channel，以及从 Channel 到 Sink 的事件传递。

比如 spooling directory source 为文件的每一行创建一个事件，一旦事务中所有的事件全部传递到 Channel 且提交成功，那么Soucrce 就将该文件标记为完成。

同理，事务以类似的方式处理从 Channel 到 Sink 的传递过程，如果因为某种原因使得事件无法记录，那么事务将会回滚。且所有的事件都会保持到 Channel 中，等待重新传递。

* 1. **Flume** 采集数据会丢失吗**?**

根据 Flume 的架构原理，Flume 是不可能丢失数据的，其内部有完善的事务机制， Source 到 Channel 是事务性的，Channel 到 Sink 是事务性的，因此这两个环节不会出现数据的丢失，唯一可能丢失数据的情况是 Channel 采用 memoryChannel，agent 宕机导致数据丢失，或者 Channel 存储数据已满，导致 Source 不再写入，未写入的数据丢失。

Flume 不会丢失数据，但是有可能造成数据的重复，例如数据已经成功由 Sink 发出，但是没有接收到响应，Sink 会再次发送数据，此时可能会导致数据的重复。