法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象学院
  - 新浪微博：小象AI学院

大纲

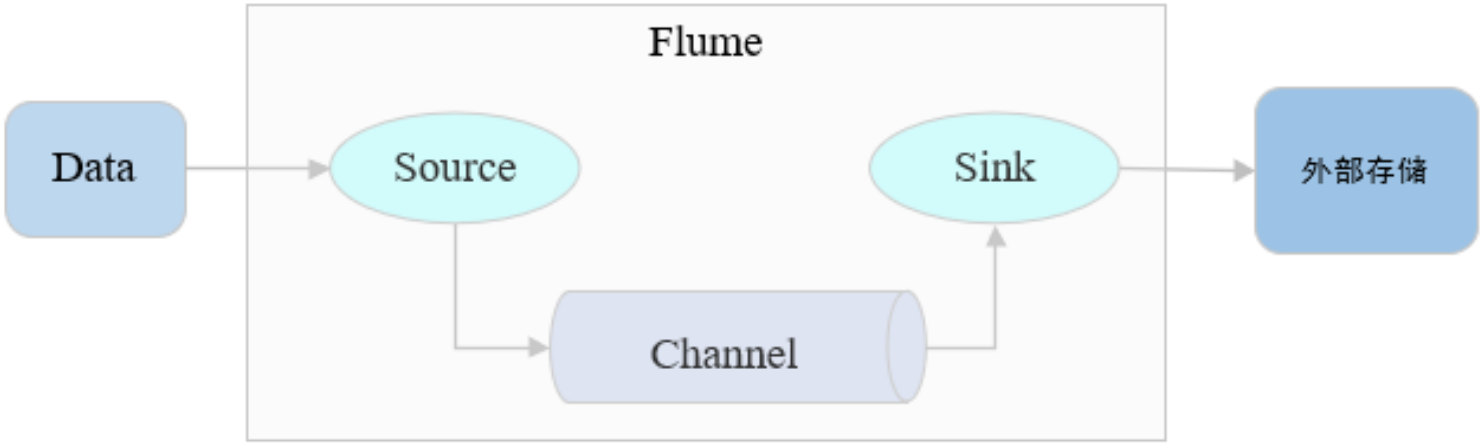
- Flume基本原理
- Flume环境搭建
- Flume组件详解与数据流
- Flume构建数据收集系统的设计与实现

大纲

Flume基本原理

Flume简介

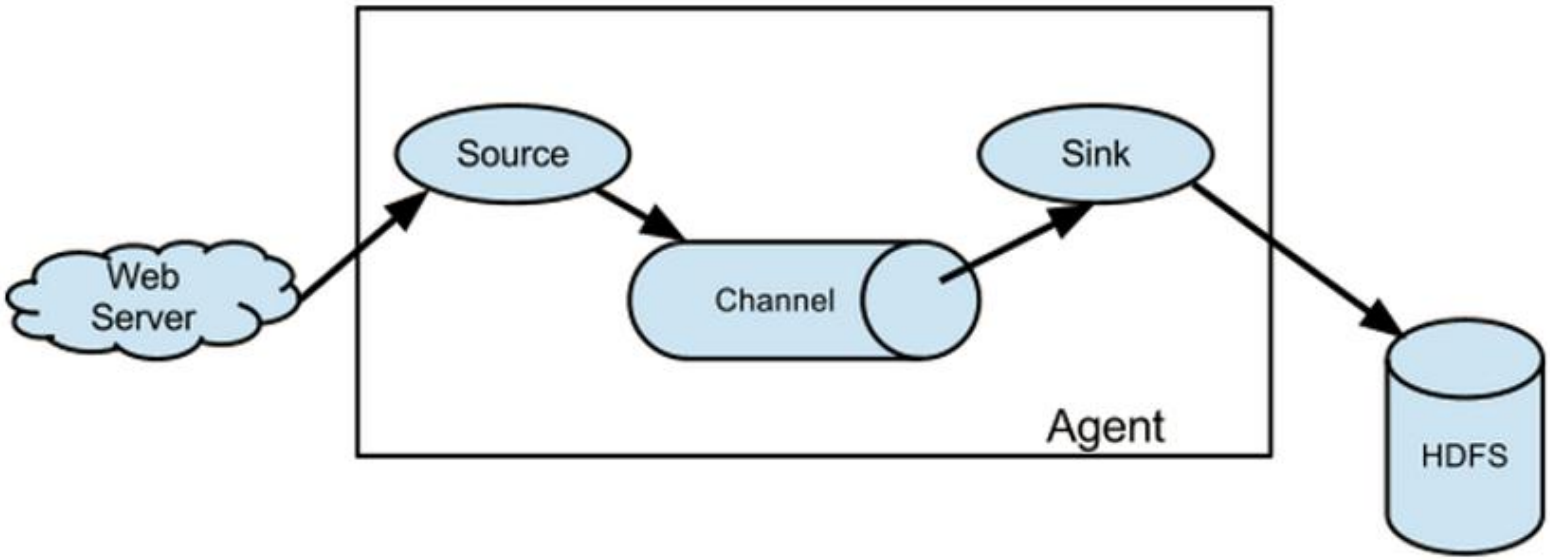
- Flume是一个分布式的、可靠的、高可用的海量日志采集、聚合和传输的系统
- 数据流模型：Source-Channel-Sink
- 事务机制保证消息传递的可靠性 。
- 内置丰富插件，轻松与其他系统集成
- Java实现，优秀的系统框架设计，模块分明，易于开发



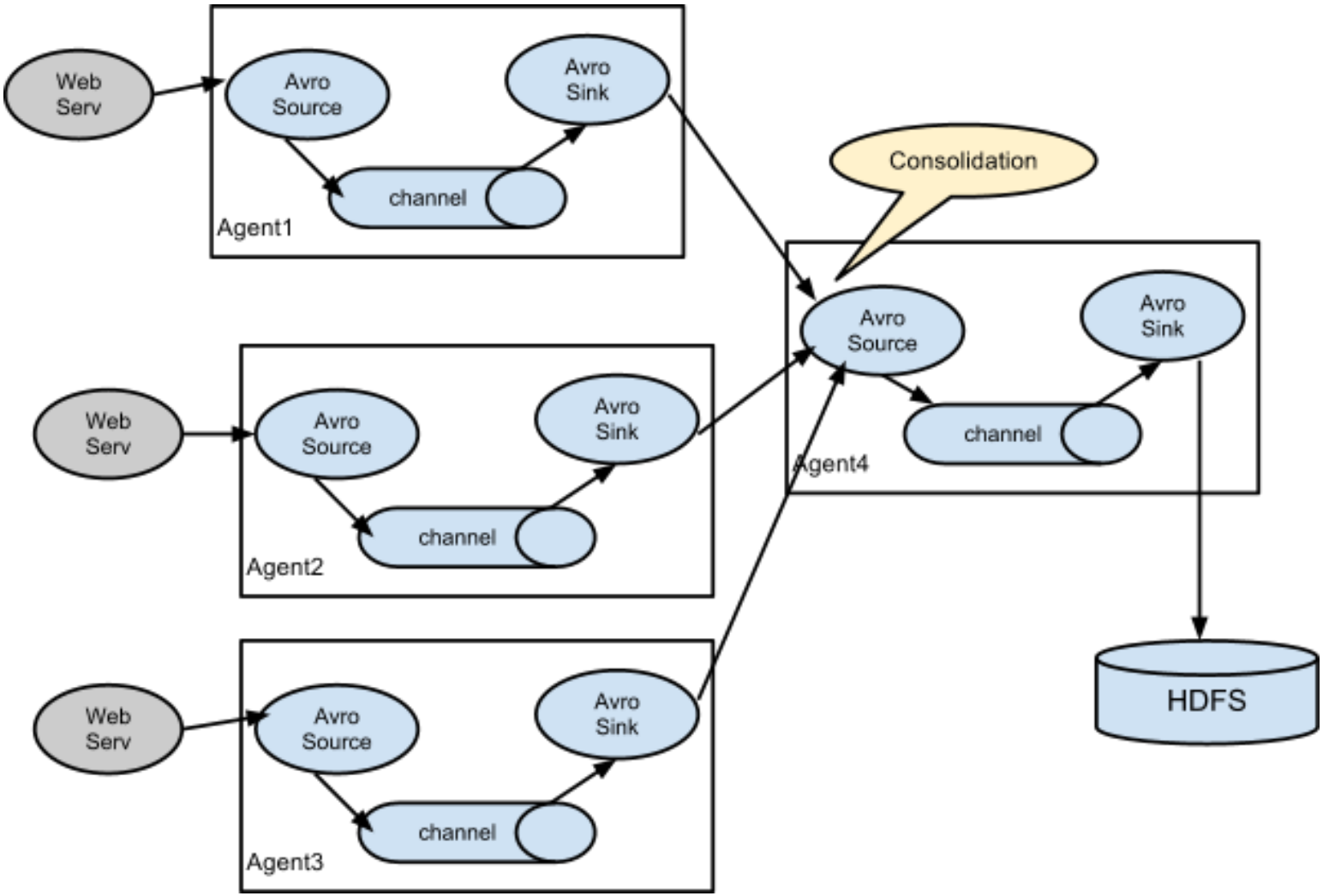
Flume基本组件

- Event：消息的基本单位，有header和body组成
- Agent：JVM进程，负责将一端外部来源产生的消息转发到另一端外部的目的地
  - Source：从外部来源读入event，并写入channel
  - Channel：event暂存组件，source写入后，event将会一直保存,直到被sink成功消费
  - Sink：从channel读入event，并写入目的地

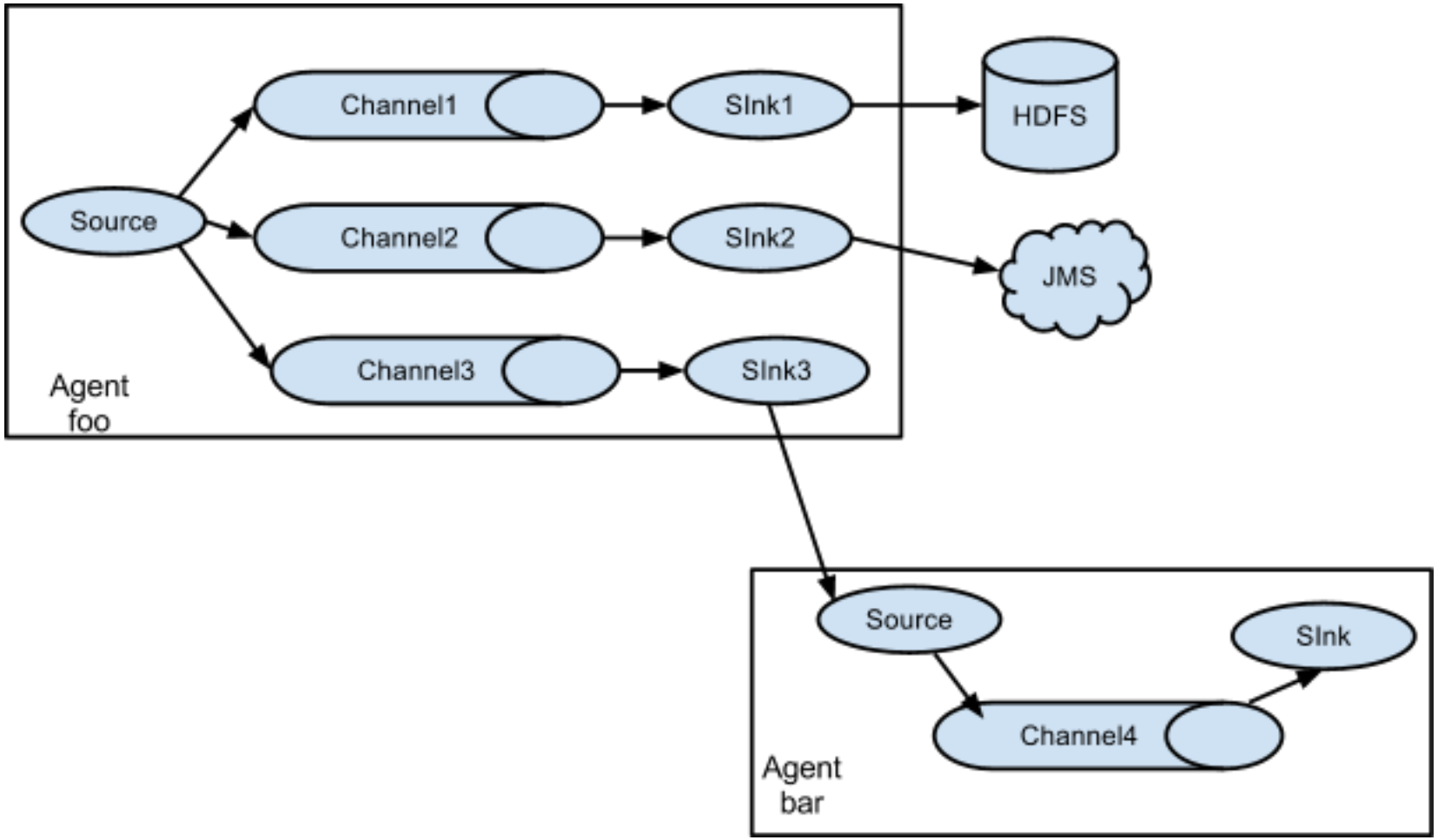
Flume数据流



Flume数据流


Web Serv
Avro Source
Avro Sink
channel
Agent1
Web Serv
Avro Source
Avro Sink
channel
Agent2
Web Serv
Avro Source
Avro Sink
channel
Agent3
Consolidation
Avro Source
Avro Sink
channel
Agent4
HDFS

Flume数据流


Channel1
Sink1
HDFS
Source
Channel2
Sink2
JMS
Channel3
Sink3
Agent
foo
Source
Sink
Channel4
Agent
bar

大纲

Flume环境搭建

Flume搭建

- 下载官方编译好的二进制压缩包

  http://flume.apache.org/download.html

- 可以使用maven自己编译源码生成安装包

  gihub源码地址：git@github.com:apache/flume.git

  maven编译：mvn clean install -DskipTests -Phadoop-2

Flume搭建

- 一个简单的例子
  - 创建conf/example.conf

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444
a1.sources.r1.channels = c1
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100
a1.sinks.k1.type = logger
a1.sinks.k1.channel = c1
```

  - 启动agent

bin/flume-ng agent --conf conf --conf-file conf/example.conf --name a1 -Dflume.root.logger=INFO,console

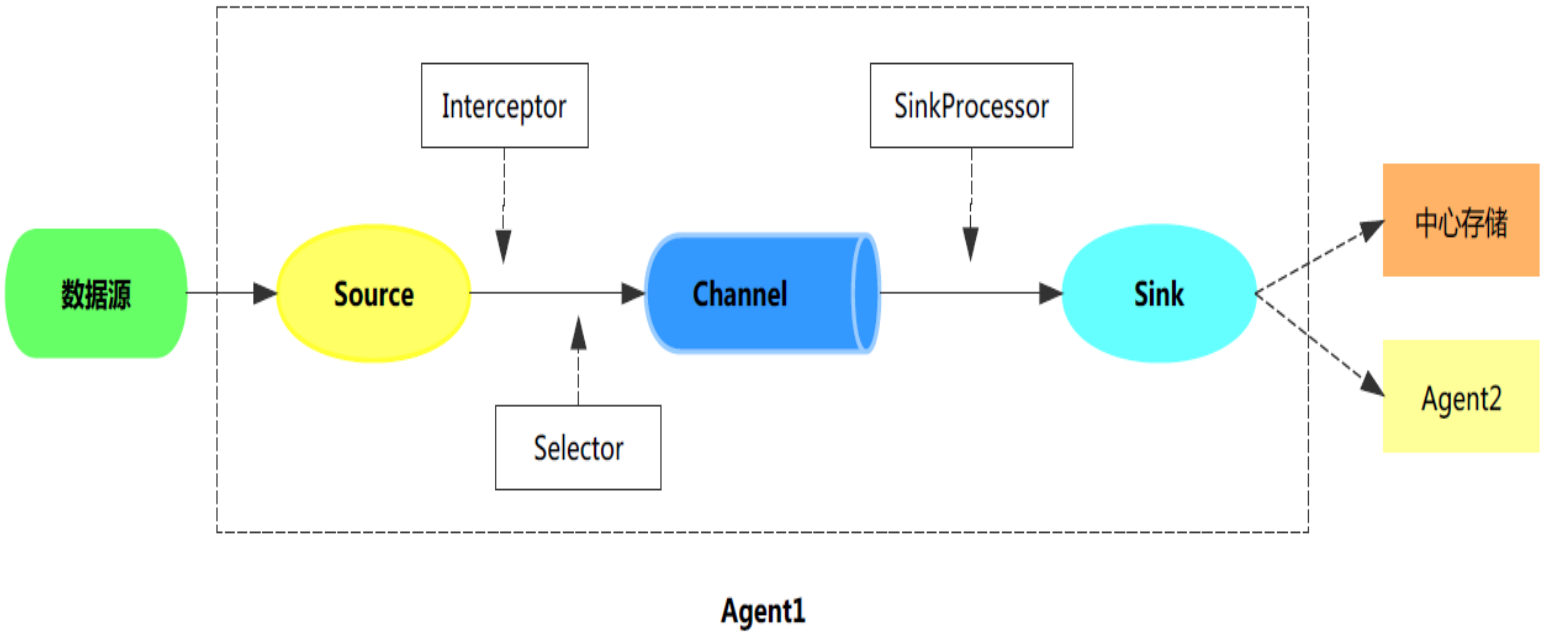
大纲

Flume组件详解与数据流

Event完整的传输处理流程

一个event在一个agent中的传输处理流程如下：

source--interceptor--selector->channel->sink processor--sink->中心存储/下一级agent



Source组件

- Source：对接各种外部数据源，将收集到的事件发送到Channel中，一个source可以向多个channel发送event，Flume内置非常丰富的Source，同时用户可以自定义Source

Source类型	Type	用途
Avro Source	avro	启动一个Avro Server，可与上一级Agent连接
HTTP Source	http	启动一个HttpServer
Exec Source	exec	执行unix command，获取标准输出，如tail -f
Taildir Source	TAILDIR	监听目录或文件
Spooling Directory Source	spooldir	监听目录下的新增文件
Kafka Source	org.apache.flume.source.kafka.KafkaSource	读取Kafka数据
JMS Source	jms	从JMS源读取数据

Source组件- Avro Source

- Avro Source：支持 Avro协议，接收 RPC事件请求。Avro Source通过监听Avro端口接收外部Avro客户端流事件（event），在Flume的多层架构中经常被使用接收上游Avro Sink发送的event
- 关键参数说明

- type：类型名称avro
- bind ：绑定的IP
- port ：监听的端口
- threads：接收请求的线程数，当需要接收多个avro客户端的数据流时要设置合适的线程数，否则会造成avro客户端数据流积压
- compression-type：是否使用压缩，如果使用压缩设则值为“deflate”，avro source一般用于多个Agent组成的数据流，接收来自avro sink的event，如果avro source设置了压缩，name上一阶段的avro sink也要设置压缩。默认值none
- channels：Source对接的Channel名称

Avro Source示例

- Agent名称为avroagent，Source名称为r1，类型为avro ，绑定192.168.183.100 ，监听端口8888。 avro source启动接收客户端数据流的最大线程数为3 。channel名称为c1 ，类型为memory channel，Sink名称为k1，类型为logger，将消费的event输出到console控制台

```
avroagent.sources = r1
avroagent.channels = c1
avroagent.sinks = k1
avroagent.sources.r1.type = avro
avroagent.sources.r1.bind = 192.168.183.100
avroagent.sources.r1.port = 8888
avroagent.sources.r1.threads= 3
avroagent.sources.r1.channels = c1
avroagent.channels.c1.type = memory
avroagent.channels.c1.capacity = 10000
avroagent.channels.c1.transactionCapacity = 1000
avroagent.sinks.k1.type = logger
avroagent.sinks.k1.channel = c1
```

Source组件- Exec Source

- Exec Source：支持Linux命令，收集标准输出数据或者通过tail -f file的方式监听指定文件。
- Exec Source可以实现实时的消息传输，但是它并不记录已经读取文件的位置，不支持断点续传，当Exec Source重启或者挂掉都会造成后续增加的消息丢失，一般在测试环境使用
- 关键参数说明
  - type ： source类型为exec
  - command ： Linux命令
  - channels ： Source对接的Channel名称。

Exec Source示例

- Agent名称为execagent，Source名称为r1，Sink名称为k1，channel名称为c1，source类型为exec，r1将event发送到channel c1，r1通过tail -F命令监听exectest.log文件，k1类型为avro，将消费的event以RPC的方式发送到192.168.183.100的8888端口

```
execagent.sources = r1
execagent.channels = c1
execagent.sinks = k1
execagent.sources.r1.type = exec
execagent.sources.r1.command = tail -F /home/hadoop/apps/flume/execsource/exectest.log
execagent.sources.r1.channels = c1
execagent.channels.c1.type = memory
execagent.channels.c1.capacity = 10000
execagent.channels.c1.transactionCapacity = 1000
execagent.sinks.k1.type = avro
execagent.sinks.k1.channel = c1
execagent.sinks.k1.hostname = 192.168.183.100
execagent.sinks.k1.port = 8888
```

Source组件- Spooling Directory Source

- Spooling Directory Source：监听一个文件夹，收集文件夹下文件数据，收集完文件数据会将文件名称的后缀改为.COMPLETED
- 缺点不支持已存在文件新增数据的收集，且不能够对嵌套文件夹递归监听
- 关键参数说明
  - type ： source类型为spooldir
  - spoolDir：source监听的文件夹
  - fileHeader ： 是否添加文件的绝对路径到event的header中，默认值false
  - fileHeaderKey：添加到event header中文件绝对路径的键值，默认值file
  - fileSuffix：收集完新文件数据给文件添加的后缀名称，默认值：.COMPLETED
  - channels ： Source对接的Channel名称

SpoolDir Source示例

- Agent名称为a1，Source名称为r1，Sink名称为k1，channel名称为c1，source类型为spooldir，r1监听的文件夹路径 /home/hadoop/apps/flume/spoolDir ，r1将event发送到channel c1，并在event头信息中添加文件绝对路径信息，k1类型为logger，将消费的event输出到console控制台

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = spooldir
a1.sources.r1.channels = c1
a1.sources.r1.spoolDir = /home/hadoop/apps/flume/spoolDir
a1.sources.r1.fileHeader = true
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 1000
a1.sinks.k1.type = logger
a1.sinks.k1.channel = c1
```

Source组件- Kafka Source

- Kafka Source：对接分布式消息队列kafka，作为kafka的消费者持续从kafka中拉取数据，如果多个kafka source同时消费kafka中同一个主题（topic），则kafka source的kafka.consumer.group.id应该设置成相同的组id，多个kafka source之间不会消费重复的数据，每一个source都会拉取topic下的不同数据
- 关键参数说明
  - type：类型设置为kafksouce的类路径，org.apache.flume.source.kafka.KafkaSource
  - channels：Source对接的Channel名称
  - kafka.bootstrap.servers ：Kafka broker列表，格式为ip1:port1, ip2:port2…，建议配置多个值提高容错能力，多个值之间用逗号隔开
  - kafka.topics：消费的topic名称
  - kafka.consumer.group.id：kafka source所属组id，默认值flume
  - batchSize ：批量写入channel的最大消息数，默认值1000
  - batchDurationMillis ：等待批量写入channel的最长时间，这个参数和batchSize 两个参数只要有一个先满足都会触发批量写入channel操作，默认值1000毫秒

Kafka Source示例

- Agent名称为a1 ， Source名称为r1，Sink名称为k1，channel名称为c1，source类型为kafkasource，r1对接的channel名称为c1，r1批量写入c1的最大消息数为1000，r1等待批量写入c1的最长时间为2秒，r1消费的主题名称为flumetopictest1 ， r1所属的consumer group id为flumekafkagroupid ， sink k1类型为logger，将消费的event输出到console控制台。

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.r1.channels = c1
a1.sources.r1.batchSize = 1000
a1.sources.r1.batchDurationMillis = 2000
a1.sources.r1.kafka.bootstrap.servers = 192.168.1.1:9092,192.168.1.2:9092
a1.sources.r1.kafka.topics = flumetopictest1
a1.sources.r1.kafka.consumer.group.id = flumekafkagroupid
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 1000
a1.sinks.k1.type = logger
a1.sinks.k1.channel = c1
```

Source组件- Taildir Source

- Taildir Source：监听一个文件夹或者文件，通过正则表达式匹配需要监听的数据源文件，Taildir Source通过将监听的文件位置写入到文件中来实现断点续传，并且能够保证没有重复数据的读取
- 关键参数说明
  - type：source类型TAILDIR
  - positionFile：保存监听文件读取位置的文件路径
  - idleTimeout：关闭空闲文件延迟时间，如果有新的记录添加到已关闭的空闲文件taildir srouce将继续打开该空闲文件，默认值120000毫秒(2分钟)
  - writePosInterval：向保存读取位置文件中写入读取文件位置的时间间隔，默认值3000毫秒
  - batchSize：批量写入channel最大event数，默认值100
  - maxBackoffSleep：每次最后一次尝试没有获取到监听文件最新数据的最大延迟时间，默认值5000毫秒

Source组件- Taildir Source

- 关键参数说明
  - cachePatternMatching：对于监听的文件夹下通过正则表达式匹配的文件可能数量会很多，将匹配成功的监听文件列表和读取文件列表的顺序都添加到缓存中，可以提高性能，默认值true
  - fileHeader ：是否添加文件的绝对路径到event的header中，默认值false
  - fileHeaderKey：添加到event header中文件绝对路径的键值，默认值file
  - filegroups：监听的文件组列表，taildirsource通过文件组监听多个目录或文件
  - filegroups.<filegroupName>：文件正则表达式路径或者监听指定文件路径
  - channels：Source对接的Channel名称

Taildir Source示例

- Agent名称为a1，Source名称为r1，Sink名称为k1，Channel名称为c1，source类型为TAILDIR，r1对接的channel名称为c1，设置保存监听文件读取位置信息的文件路径，监听文件列表包含两个监听文件组f1、f2，f1监听指定example.log，f2通过正则表达式匹配指定路径下包含log关键字的所有文件。sink k1类型为logger，将消费的event输出到console控制台。

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = TAILDIR
a1.sources.r1.positionFile = /home/hadoop/apps/flume/taildir/position/taildir_position.json
a1.sources.r1.filegroups = f1 f2
a1.sources.r1.filegroups.f1 = /home/hadoop/apps/flume/taildir/test1/example.log
a1.sources.r1.filegroups.f2 = /home/hadoop/apps/flume/taildir/test2/.*log.*
a1.sources.r1.channels = c1
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 1000
a1.sinks.k1.type = logger
a1.sinks.k1.channel = c1
```

Channel组件

- Channel：Channel被设计为event中转暂存区，存储Source收集并且没有被Sink消费的event，为了平衡Source收集和Sink读取数据的速度，可视为Flume内部的消息队列。
- Channel是线程安全的并且具有事务性，支持source写失败重复写和sink读失败重复读等操作
- 常用的Channel类型有：Memory Channel、File Channel、Kafka Channel、JDBC Channel等

Channel组件- Memory Channel

- Memory Channel：使用内存作为Channel，Memory Channel读写速度快，但是存储数据量小，Flume进程挂掉、服务器停机或者重启都会导致数据丢失。部署Flume Agent的线上服务器内存资源充足、不关心数据丢失的场景下可以使用
- 关键参数说明
  - type ： channel类型memory
  - capacity ： channel中存储的最大event数，默认值100
  - transactionCapacity ： 一次事务中写入和读取的event最大数，默认值100。
  - keep-alive：在Channel中写入或读取event等待完成的超时时间，默认值3秒
  - byteCapacityBufferPercentage：缓冲空间占Channel容量（byteCapacity）的百分比，为event中的头信息保留了空间，默认值20（单位百分比）
  - byteCapacity ： Channel占用内存的最大容量，默认值为Flume堆内存的80%

Channel组件- File Channel

- File Channel：将event写入到磁盘文件中，与Memory Channel相比存储容量大，无数据丢失风险。
- File Channle数据存储路径可以配置多磁盘文件路径，提高写入文件性能
- Flume将Event顺序写入到File Channel文件的末尾，在配置文件中通过设置maxFileSize参数设置数据文件大小上限
- 当一个已关闭的只读数据文件中的Event被完全读取完成，并且Sink已经提交读取完成的事务，则Flume将删除存储该数据文件
- 通过设置检查点和备份检查点在Agent重启之后能够快速将File Channle中的数据按顺序回放到内存中

Channel组件- File Channel

关键参数说明：

- type：channel类型为file
- checkpointDir：检查点目录，默认在启动flume用户目录下创建，建议单独配置磁盘路径
- useDualCheckpoints：是否开启备份检查点，默认false，建议设置为true开启备份检查点，备份检查点的作用是当Agent意外出错导致写入检查点文件异常，在重新启动File Channel时通过备份检查点将数据回放到内存中，如果不开启备份检查点，在数据回放的过程中发现检查点文件异常会对所有数据进行全回放，全回放的过程相当耗时
- backupCheckpointDir：备份检查点目录，最好不要和检查点目录在同一块磁盘上
- checkpointInterval：每次写检查点的时间间隔，默认值30000毫秒

Channel组件- File Channel

关键参数说明：

- dataDirs：数据文件磁盘存储路径，建议配置多块盘的多个路径，通过磁盘的并行写入来提高file channel性能，多个磁盘路径用逗号隔开
- transactionCapacity：一次事务中写入和读取的event最大数，默认值10000
- maxFileSize：每个数据文件的最大大小，默认值：2146435071字节
- minimumRequiredSpace：磁盘路径最小剩余空间，如果磁盘剩余空间小于设置值，则不再写入数据
- capacity：file channel可容纳的最大event数
- keep-alive：在Channel中写入或读取event等待完成的超时时间，默认值3秒

File Channel示例

- Agent名称为a1，Source名称为r1，Sink名称为k1，Channel名称为c1，channel类型为file，检查点路径 /home/hadoop/apps/flume/file_channnel_checkpoint ，数据存放路径 /home/hadoop/apps/flume/file_channnel_data，开启备份检查点，备份检查点路径 /home/hadoop/apps/flume/file_channnel_backup，sink k1类型为logger，将消费的event输出到console控制台。

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444
a1.sources.r1.channels = c1
a1.channels.c1.type = file
a1.channels.c1.dataDirs = /home/hadoop/apps/flume/filechannel/data
a1.channels.c1.checkpointDir = /home/hadoop/apps/flume/filechannel/checkpoint
a1.channels.c1.useDualCheckpoints = true
a1.channels.c1.backupCheckpointDir = /home/hadoop/apps/flume/filechannel/backup
a1.sinks.k1.type = logger
a1.sinks.k1.channel = c1
```

Channel组件- Kafka Channel

- Kafka Channel：将分布式消息队列kafka作为channel
- 相对于Memory Channel和File Channel存储容量更大、容错能力更强，弥补了其他两种Channel的短板，如果合理利用Kafka的性能，能够达到事半功倍的效果

Channel组件- Kafka Channel

关键参数说明：

- type：Kafka Channel类型org.apache.flume.channel.kafka.KafkaChannel
- kafka.bootstrap.servers：Kafka broker列表，格式为ip1:port1, ip2:port2…，建议配置多个值提高容错能力，多个值之间用逗号隔开
- kafka.topic：topic名称，默认值“flume-channel”
- kafka.consumer.group.id：Consumer Group Id，全局唯一
- parseAsFlumeEvent：是否以Avro FlumeEvent模式写入到Kafka Channel中，默认值true，event的header信息与event body都写入到kafka中
- pollTimeout：轮询超时时间，默认值500毫秒
- kafka.consumer.auto.offset.reset：earliest表示从最早的偏移量开始拉取，latest表示从最新的偏移量开始拉取，none表示如果没有发现该Consumer组之前拉取的偏移量则抛异常

Kafka Channel示例

- Agent名称为a1，Source名称为r1，Sink名称为k1，Channel名称为c1，channel类型为kafkachannel，使用的topic名称flumechannel，Consumer Group ID 为flumecg1

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444
a1.sources.r1.channels = c1
a1.channels.c1.type = org.apache.flume.channel.kafka.KafkaChannel
a1.channels.c1.kafka.bootstrap.servers =
192.168.183.102:9092,192.168.183.103:9092,192.168.183.104:9092
a1.channels.c1.kafka.topic = flumechannel
a1.channels.c1.kafka.consumer.group.id = flumecg1
a1.sinks.k1.type = logger
a1.sinks.k1.channel = c1
```

Sink组件

- Sink：从Channel消费event，输出到外部存储，或者输出到下一个阶段的agent
- 一个Sink只能从一个Channel中消费event
- 当Sink写出event成功后，就会向Channel提交事务。Sink事务提交成功，处理完成的event将会被Channel删除。否则Channel会等待Sink重新消费处理失败的event
- Flume提供了丰富的Sink组件，如Avro Sink、HDFS Sink、Kafka Sink、File Roll Sink、HTTP Sink等

Sink组件- Avro Sink

- Avro Sink常用于对接下一层的Avro Source，通过发送RPC请求将Event发送到下一层的Avro Source
- 为了减少Event传输占用大量的网络资源， Avro Sink提供了端到端的批量压缩数据传输
- 关键参数说明
  - type：Sink类型为avro。
  - hostname：绑定的目标Avro Souce主机名称或者IP
  - port：绑定的目标Avro Souce端口号
  - batch-size：批量发送Event数，默认值100
  - compression-type：是否使用压缩，如果使用压缩设则值为“deflate”， Avro Sink设置了压缩那么Avro Source也应设置相同的压缩格式，目前支持zlib压缩，默认值none
  - compression-level：压缩级别，0表示不压缩，从1到9数字越大压缩效果越好，默认值6

Sink组件- HDFS Sink

- HDFS Sink将Event写入到HDFS中持久化存储
- HDFS Sink提供了强大的时间戳转义功能，根据Event头信息中的timestamp时间戳信息转义成日期格式，在HDFS中以日期目录分层存储
- 关键参数说明
  - type：Sink类型为hdfs。
  - hdfs.path：HDFS存储路径，支持按日期时间分区。
  - hdfs.filePrefix：Event输出到HDFS的文件名前缀，默认前缀FlumeData
  - hdfs.fileSuffix：Event输出到HDFS的文件名后缀
  - hdfs.inUsePrefix：临时文件名前缀
  - hdfs.inUseSuffix：临时文件名后缀，默认值.tmp
  - hdfs.rollInterval：HDFS文件滚动生成时间间隔，默认值30秒，该值设置为0表示文件不根据时间滚动生成

Sink组件- HDFS Sink

- 关键参数说明
  - hdfs.rollSize：临时文件滚动生成大小，默认值1024B，该值设置为0表示文件不根据文件大小滚动生成。
  - hdfs.rollCount：临时文件滚动生成的Event数，默认值10，该值设置为0表示文件不根据Event数滚动生成。
  - hdfs.idleTimeout：临时文件等待Event写入的超时时间，达到超时时间临时文件自动关闭重命名为目标文件名称，默认值0秒，该值设置为0表示禁用此功能，不自动关闭临时文件。
  - hdfs.batchSize：Flume批量写入HDFS的Event数量，默认值100
  - hdfs.callTimeout：操作HDFS文件的超时时间，如果需要写入HDFS文件的Event数比较大或者发生了打开、写入、刷新、关闭文件超时的问题，可以根据实际情况适当的增大超时时间。默认值10000毫秒
  - hdfs.writeFormat：写出到hdfs文件格式，目前可以选择Text或者Writable两种格式，默认值Writable

Sink组件- HDFS Sink

- 关键参数说明
  - hdfs.round：用于HDFS文件按照时间分区，时间戳向下取整，默认值false
  - hdfs.roundValue：当round设置为true，配合roundUnit时间单位一起使用，例如roundUnit值为minute，该值设置为1则表示一分钟之内的数据写到一个文件中，相当于每一分钟生成一个文件。默认值1
  - hdfs.roundUnit：按时间分区使用的时间单位，可以选择second秒、minute分钟、hour小时三种粒度的时间单位，默认值second秒
  - hdfs.timeZone：写入HDFS文件使用的时区，默认值Local Time本地时间
  - hdfs.useLocalTimeStamp：是否使用本地时间替换Event头信息中的时间戳，默认值false
  - hdfs.codeC：文件压缩格式，目前支持的压缩格式有gzip、bzip2、lzo、lzop、snappy，默认不采用压缩
  - hdfs.fileType：文件类型，DataStream则输出的文件不会进行压缩，CompressedStream则对输出的文件进行压缩需要设置hdfs.codeC指定压缩格式。默认值SequenceFile

HDFS Sink示例

- Source r1使用timestamp拦截器，在event header中添加timestamp时间戳信息。使用HDFS Sink将event写入HDFS的/data/flume路径下，并且按照年月日分区，写入到HDFS的Text文件以hdfssink-开头，每一分钟生成一个文件，时间向下取整，HDFS操作超时时间为1分钟

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
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 = timestamp
a1.sources.r1.interceptors.i1.preserveExisting = false
a1.sources.r1.channels = c1
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 1000
a1.sinks.k1.type = hdfs
a1.sinks.k1.channel = c1
a1.sinks.k1.hdfs.path = /data/flume/%Y%m%d
a1.sinks.k1.hdfs.filePrefix = hdfssink
a1.sinks.k1.hdfs.fileType = DataStream
a1.sinks.k1.hdfs.writeFormat = Text
a1.sinks.k1.hdfs.round = true
a1.sinks.k1.hdfs.roundValue = 1
a1.sinks.k1.hdfs.roundUnit = minute
a1.sinks.k1.hdfs.callTimeout = 60000
```

Sink组件- Kafka Sink

- Flume通过KafkaSink将Event写入到Kafka指定的主题中
- 关键参数说明
  - type：Sink类型，值为KafkaSink类路径org.apache.flume.sink.kafka.KafkaSink。
  - kafka.bootstrap.servers：Broker列表，定义格式host:port，多个Broker之间用逗号隔开，可以配置一个也可以配置多个，用于Producer发现集群中的Broker，建议配置多个，防止当个Broker出现问题连接失败。
  - kafka.topic：Kafka中Topic主题名称，默认值flume-topic。
  - flumeBatchSize：Producer端单次批量发送的消息条数，该值应该根据实际环境适当调整，增大批量发送消息的条数能够在一定程度上提高性能，但是同时也增加了延迟和Producer端数据丢失的风险。默认值100。

Sink组件- Kafka Sink

- 关键参数说明
  - kafka.producer.acks：设置Producer端发送消息到Borker是否等待接收Broker返回成功送达信号。0表示Producer发送消息到Broker之后不需要等待Broker返回成功送达的信号，这种方式吞吐量高，但是存在数据丢失的风险。1表示Broker接收到消息成功写入本地log文件后向Producer返回成功接收的信号，不需要等待所有的Follower全部同步完消息后再做回应，这种方式在数据丢失风险和吞吐量之间做了平衡。all（或者-1）表示Broker接收到Producer的消息成功写入本地log并且等待所有的Follower成功写入本地log后向Producer返回成功接收的信号，这种方式能够保证消息不丢失，但是性能最差。默认值1。
  - useFlumeEventFormat：默认值false，Kafka Sink只会将Event body内容发送到Kafka Topic中。如果设置为true，Producer发送到Kafka Topic中的Event将能够保留Producer端头信息

Kafka Sink示例

- 使用Kafka Sink向“FlumeKafkaSinkTopic”主题批量发送消息，批量发送的消息数量为100

```
a1.sources = r1
a1.channels = c1
a1.sinks = k1
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444
a1.sources.r1.channels = c1
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 1000
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.channel = c1
a1.sinks.k1.kafka.topic = FlumeKafkaSinkTopic1
a1.sinks.k1.kafka.bootstrap.servers = 192.168.183.102:9092,192.168.183.103:9092
a1.sinks.k1.kafka.flumeBatchSize = 100
a1.sinks.k1.kafka.producer.acks = 1
```

Interceptor拦截器

- Source将event写入到Channel之前调用拦截器
- Source和Channel之间可以有多个拦截器，不同的拦截器使用不同的规则处理Event
- 可选、轻量级、可插拔的插件
- 通过实现Interceptor接口实现自定义的拦截器
- 内置拦截器：Timestamp Interceptor、Host Interceptor、UUID Interceptor、Static Interceptor、Regex Filtering Interceptor等

Timestamp Interceptor

- Flume使用时间戳拦截器在event头信息中添加时间戳信息， Key为timestamp，Value为拦截器拦截Event时的时间戳
- 头信息时间戳的作用，比如HDFS存储的数据采用时间分区存储，Sink可以根据Event头信息中的时间戳将Event按照时间分区写入到HDFS
- 关键参数说明：
  - type:拦截器类型为timestamp
  - preserveExisting：如果头信息中存在timestamp时间戳信息是否保留原来的时间戳信息，true保留，false使用新的时间戳替换已经存在的时间戳，默认值为false

Host Interceptor

- Flume使用主机戳拦截器在Event头信息中添加主机名称或者IP
- 主机拦截器的作用：比如Source将Event按照主机名称写入到不同的Channel中便于后续的Sink对不同Channnel中的数据分开处理
- 关键参数说明：
  - type:拦截器类型为host
  - preserveExisting：如果头信息中存在timestamp时间戳信息是否保留原来的时间戳信息，true保留，false使用新的时间戳替换已经存在的时间戳，默认值为false
  - useIP：是否使用IP作为主机信息写入都信息，默认值为false
  - hostHeader：设置头信息中主机信息的Key，默认值为host

Static Interceptor

- Flume使用static interceptor静态拦截器在evetn头信息添加静态信息
- 关键参数说明：
  - type:拦截器类型为static
  - preserveExisting：如果头信息中存在timestamp时间戳信息是否保留原来的时间戳信息，true保留，false使用新的时间戳替换已经存在的时间戳，默认值为false
  - key：头信息中的键
  - value：头信息中键对应的值

Selector选择器

- Source将event写入到Channel之前调用拦截器，如果配置了Interceptor拦截器，则Selector在拦截器全部处理完之后调用。通过selector决定event写入Channel的方式
- 内置Replicating Channel Selector复制Channel选择器、 Multiplexing Channel Selector复用Channel选择器

Replicating Channel Selector

- 如果Channel选择器没有指定，默认是Replicating Channel Selector。即一个Source以复制的方式将一个event同时写入到多个Channel中，不同的Sink可以从不同的Channel中获取相同的event。
- 关键参数说明：
  - selector.type：Channel选择器类型为replicating
  - selector.optional：定义可选Channel，当写入event到可选Channel失败时，不会向Source抛出异常，继续执行。多个可选Channel之间用空格隔开

```
#设置选择器
a1.sources.r1.selector.type = replicating
#设置required channel
a1.sources.r1.channels = c1
#设置optional channel
a1.sources.r1.selector.optional = c2
```

Multiplexing Channel Selector

- Multiplexing Channel Selector多路复用选择器根据event的头信息中不同键值数据来判断Event应该被写入到哪个Channel中
- 三种级别的Channel，分别是必选channle、可选channel、默认channel
- 关键参数说明：
  - selector.type：Channel选择器类型为multiplexing
  - selector.header：设置头信息中用于检测的headerName
  - selector.default：默认写入的Channel列表
  - selector.mapping.*：headerName对应的不同值映射的不同Channel列表
  - selector.optional：可选写入的Channel列表

Sink Processor

- Sink Processor协调多个sink间进行load balance和fail over
- Default Sink Processor只有一个sink，无需创建Sink Processor
- Sink Group：将多个sink放到一个组内，要求组内一个sink消费channel
- Load-Balancing Sink Processor（负载均衡处理器）round_robin(默认)或random
- Failover Sink Processor（容错处理器）可定义一个sink优先级列表，根据优先级选择使用的sink

Load-Balancing Sink Processor

- 关键参数说明：
  - sinks：sink组内的子Sink，多个子sink之间用空格隔开
  - processor.type：设置负载均衡类型load_balance
  - processor.backoff：设置为true时，如果在系统运行过程中执行的Sink失败，会将失败的Sink放进一个冷却池中。默认值false
  - processor.selector.maxTimeOut：失败sink在冷却池中最大驻留时间，默认值30000ms
  - processor.selector：负载均衡选择算法，可以使用轮询“round_robin”、随机“random”或者是继承AbstractSinkSelector类的自定义负载均衡实现类

```
agent1.sinkgroups = g1
agent1.sinkgroups.g1.sinks = k1 k2
agent1.sinkgroups.g1.processor.type = load_balance
agent1.sinkgroups.g1.processor.backoff = true
agent1.sinkgroups.g1.processor.selector = random
```

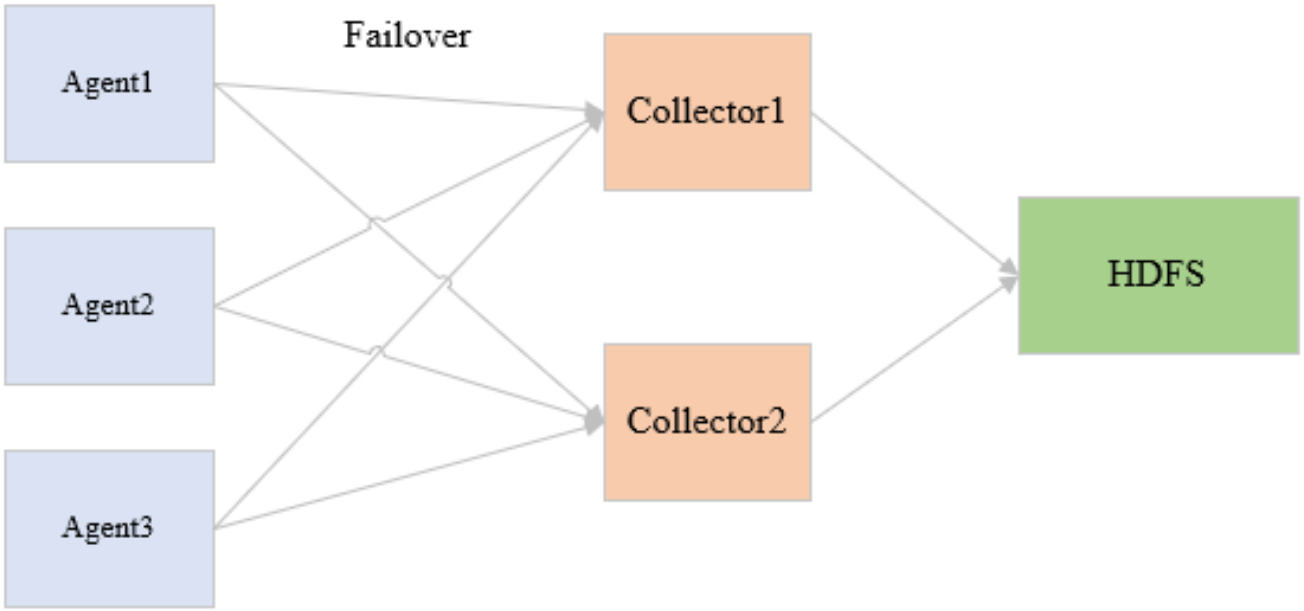
Failover Sink Processor

- 关键参数说明：
  - sinks：sink组内的子Sink，多个子sink之间用空格隔开
  - processor.type：设置故障转移类型“failover”
  - processor.priority.<sinkName>：指定Sink组内各子Sink的优先级别，优先级从高到低，数值越大优先级越高
  - processor.maxpenalty：等待失败的Sink恢复的最长时间，默认值30000毫秒

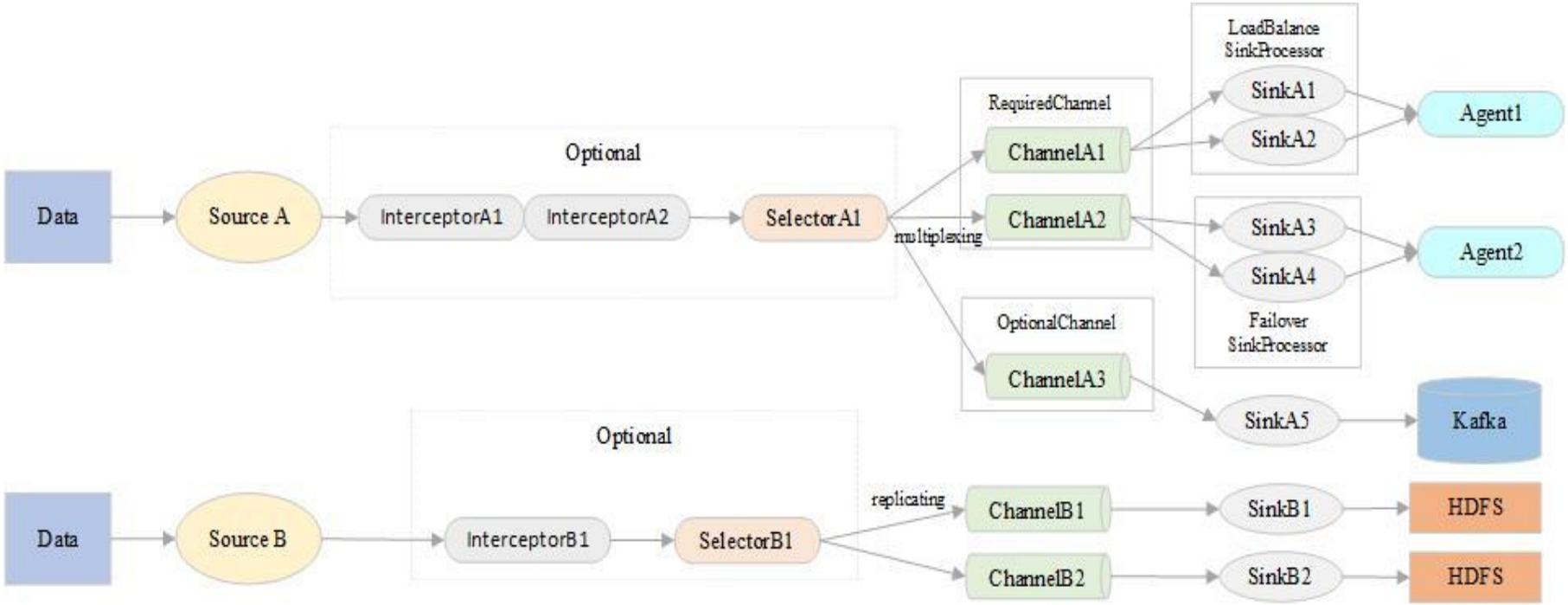
```
agent1.sinkgroups = g1
agent1.sinkgroups.g1.sinks = k1 k2
agent1.sinkgroups.g1.processor.type = failover
agent1.sinkgroups.g1.processor.priority.k1 = 5
agent1.sinkgroups.g1.processor.priority.k2 = 10
agent1.sinkgroups.g1.processor.maxpenalty = 5000
```

Failover应用场景

- 分布式日志收集场景
  - 多个agent收集不同机器上相同类型的日志数据，为了保障高可用，采用分层部署，日志收集层Collector部署两个甚至多个，Agent通过Failover Sink Processor实现其中任何一个collector挂掉不影响系统的日志收集服务



组件数据流知识点总结

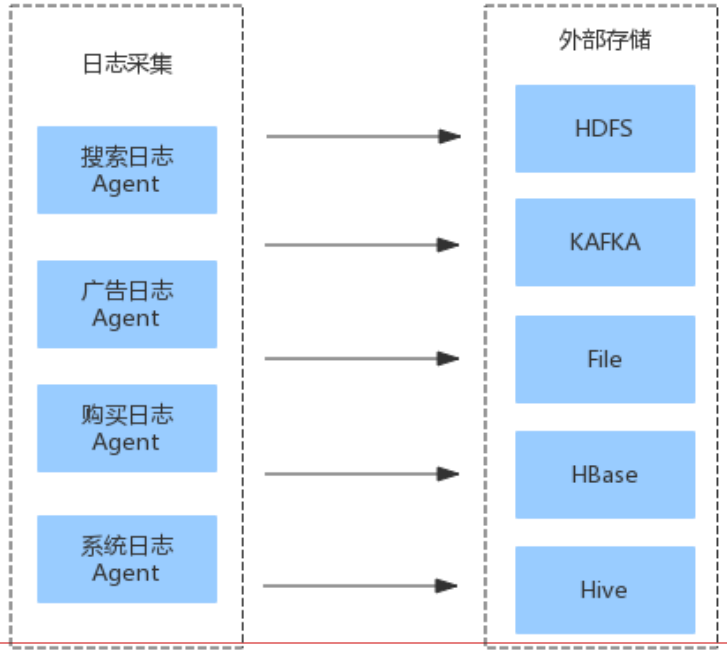

Data
Source A
Optional
InterceptorA1
InterceptorA2
SelectorA1
multiplexing
RequiredChannel
ChannelA1
ChannelA2
OptionalChannel
ChannelA3
LoadBalance
SinkProcessor
SinkA1
SinkA2
Agent1
SinkA3
SinkA4
Agent2
Failover
SinkProcessor
SinkA5
Kafka
Data
Source B
Optional
InterceptorB1
SelectorB1
replicating
ChannelB1
ChannelB2
SinkB1
SinkB2
HDFS
HDFS

大纲

Flume构建数据收集系统的设计与实现

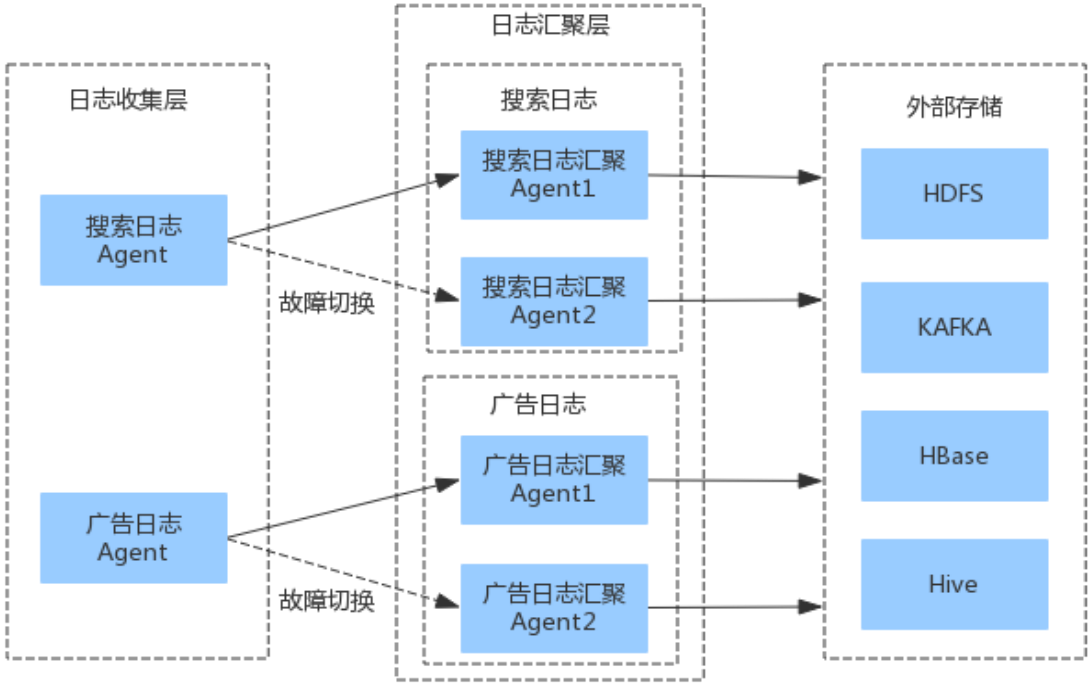
单层日志收集架构

- 优点：架构简单、使用方便
- 缺点：
  - 如果采集的数据源或者Agent比较多，将event写入到hdfs会产生很多小文件
  - 外部存储升级维护或者出现系统故障需要对所有采集层Agent做处理，人力成本高，系统稳定性差
  - 系统安全性差
  - 数据源管理混乱



分层日志收集架构

- 优点：
  - 各种类型的日志数据分层整合处理架构清晰，运维高效，降低人工误操作风险
  - 避免过多小文件产生，提高系统稳定性和处理能力
  - 不会对外部暴露关键系统的系统信息，降低被攻击的风险，大大提供系统安全性
  - 各关联系统易升级
- 缺点：相对于单一日志收集架构部署相对复杂，需要占用的机器资源更多



疑问

- 小象问答官网
  - http://wenda.chinahadoop.cn

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象学院
- 新浪微博：小象AI学院