Flume DirectoryMonitorSource 使用说明文档

版本	日期	作者	操作
V-0.1	2014-1-20	何鹏	创建文档

术语约定

术语	说明
HDFS	Hadoop distributed filesystem 一个基于 master-slave 架构的分布式文件系统，具有高可用、高吞吐、易扩展等特性；
Flume	一个数据收集工具，支持多种数据源端和数据汇聚段；可以由用于根据自己网路特点，灵活的组织、设定适合自己应用的拓扑结构；
Source	Flume 的数据源端，常见如：syslogtcp、syslogudp、netcat（socket）、scribe 等；
Sink	Flume 的数据汇聚端，常见如：logger（日志数据）、file（本地文件）、hdfs（输出到 hdfs）；
Channel	Flume 中连接数据源和数据汇聚的管道，数据的可靠性由 channel 保证；
Derby	基于 apache licence 的 sql 型轻量级数据库，支持嵌入式（embeded）应用，可以非常灵活的接入轻量级的应用；使用 derby 一般数据大小不要超过 50G；
Event	一条消息，是 flume 中传输的基本单元，通常而言，包含 header 和 body；其中 header 一般存放 meta 数据（比如 host、uuid、timestamp 等），而 body 是一个 byte 数组；在绝大部分的日志收集中，body 是日志文件中的一个记录（通常是一行数据，以换行符`\n`结束）；
record	在 DirectoryMonitorSource 一个 record 称之为一条完整的日志记录，可能包括多行数据（multiple lines）；
FQCN	Full qualified class name

简介

Flume 是一个开源数据收集工具，具有灵活的架构，易于用户扩展。由 cloudera 开源，目前已经贡献给 apache software，成为 apache 的项目。

Flume 更加倾向于一个工具而不是一个系统，在 flume 的设计中，数据流（data flow）是一个很重要的概念，根据数据流的整个生命周期，flume 将系统拆分为三个大的模块：

Source

Channel

Sink

从 Flume 的角度看来，数据由 Source 产生，经过 Channel 分发到不同的 Sink 端，其一个典型的数据流图如下：

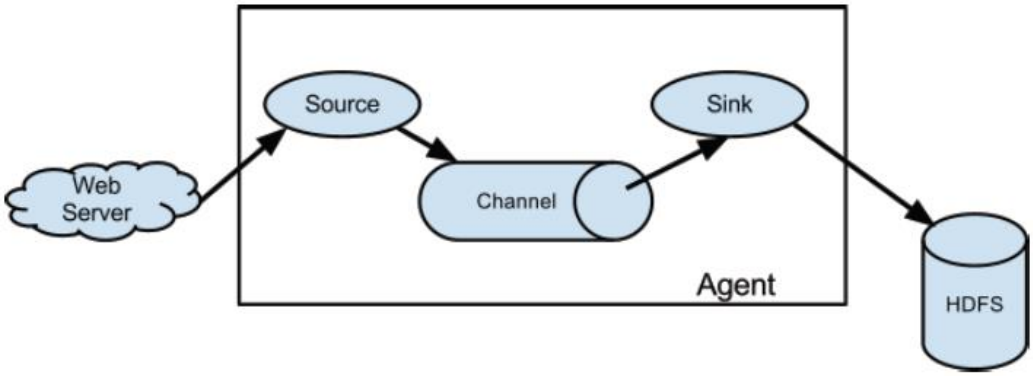


图 1 flume data flow

Flume 默认提供了较多的 Source 和 Sink 实现，简单介绍如下：

Source type	说明
avro	基于 apache avro rpc 的 server 数据源； 一般只会在两个 fume agent 之间中转数据时，其中的下游节点可以配置为 source 为 avro（对应的上游需要配置 sink 为 avro）；
thrift	同上，但 rpc 框架是基于 thrift 实现；
exec	可以执行用户指定的命令，并从命令的 stdout 读取数据作为数据源，一种典型的应用如： exec tail –f xxxoooo.data
jms	数据源来自于 java　jms（消息系统）；
spooldir	设置某一个目录为 spooldir，然后此 agent 会将目录下所有的文件作为数据源端进行传输，目前市面上大多采用此方案，但此方案对部署有如下硬性要求： spooldir 对应目录下的文件，必须是一个完结的文件（不会有应用程序继续想文件中写数据），agent 传输完毕文件后，可以选择将文件删除或者将文件 rename（作为一个标记，标记此文件已经传输）。 因此对应的缺点为： 1、 此 source 在部署是，是一种侵入式，必须对已有应用的日志切分（rotate strategy）进行修改，比如修改为每 5 分钟切换一次； 2、 需要一个外部进程（或者命令），将应用切分后，完整的一个日志片段 mv 到 spooldir 下（应用不能直接向 spooldir 中写数据，否则导致程序死掉）； 3、 传输做不到实时，因为我们不可能让应用每五秒、十秒就进行日志切换； 4、 不支持断点续传，比如 agent 挂掉后，如果某个文件已经传输一半，则可能出现重复传输已经传输过的数据（同时意味着后续对其运维的起停过程难以实现）；
netcat	tcp socket server；
seq	测试使用，自动产生序列数据；
syslogtcp	根据 syslog 产生 flume 的事件（数据）；
syslogudp	同上，基于 udp；

表 1 Flume 常见的 source 功能描述

Sink type	说明
hdfs	以 hdfs 作为数据汇聚段，支持按照时间或者大小等多种参数进行切换；支持 text 和 sequencefile（binary format）以及 meta&text 等多种方式写数据文件；
file_roll	类似于 hdfs，不过是写到了本地文件系统中；
logger	一般用于测试，将数据写到一个 looger 上；
avro	一般用于中转数据，与 avro source 配合使用；
thrift	同上，基于 thrift rpc、serialize，一般和 thrift sink 配合使用；
hbase	将数据写到 hbase 中，需要设置 columnfamily（类似于 sql 的 table）等；
MorphlineSolrSink	配合 Morphine 使用，简单讲就是将 event 经过一些列由 morphline 定义好的 handler 进行处理；在使用 cloudera search 作为 hadoop 的检索引擎时，一般会使用 morphline 进行一些处理；
ElasticSearchSink	将数据送给 elasticsearch 平台；

表 2 Flume 常见的 sink 及功能描述

应用场景说明

目前，在民生的应用场景中，应用已经在正常的运行，push 应用方修改日志的切分方式既难以实现，同时后续运维会存在更多的问题，比如切分力度过小，会出现大量的文件，极端状况耗尽 linux inode 导致服务不可用；同时过于频繁的切换，log 模块的性能会大幅度降低。

因此在我们的场景中，是不能直接使用 spooldir 作为数据源端，同时 spooldir 还存在如下的问题：

1、 不支持多行数据作为同一个处理单元—record（上层应用要求必须是这样）；
2、 需要一个外部进程（或者命令），将应用切分后，完整的一个日志片段 mv 到 spooldir 下；
3、 传输做不到实时，因为我们不可能让应用每五秒、十秒就进行日志切换；
4、 不支持断点续传，比如 agent 挂掉后，如果某个文件已经传输一半，则可能出现重复传输已经传输过的数据（同时意味着后续对其运维的起停过程难以实现）；

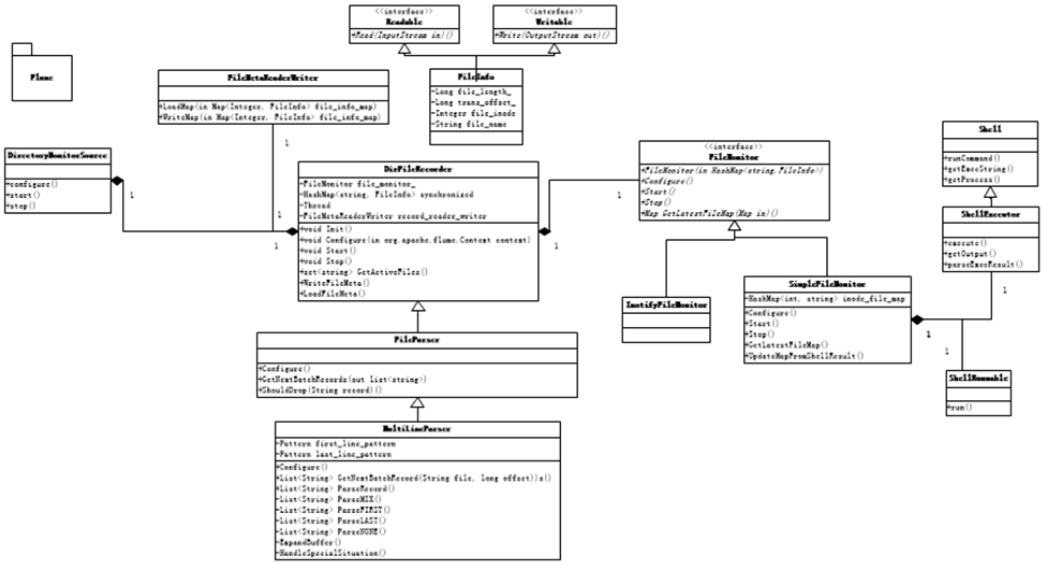
综上，我们需要开发一个适合我们自己、且通用的应用场景；

开发分析说明

根据上面介绍和分析，开发一个使用当前应用场景的通用 source 端，是必须的。根据需要整理，整理开发 Source 功能如下：

1、 可以轻量级的进行部署（不需要对运行中的应用进行日志调整）；
2、 支持断点续传；
3、 可以直接指定应用输出的目录作为数据源端，不需要引入额外的 daemon 来进行日志移动；
4、 支持多行数据作为一个 record 进行文件解析；
5、 支持 record 的黑名单和白名单过滤（可以删除确定不需要的数据）；

设计 DirectoryMonitorSource 结构如下：



注：在当前 0.1 版本中，为了实现快速开发以及部署的轻量级，未采用 java-inotiyf 模式，金实现了 SimpleFileMonitory。

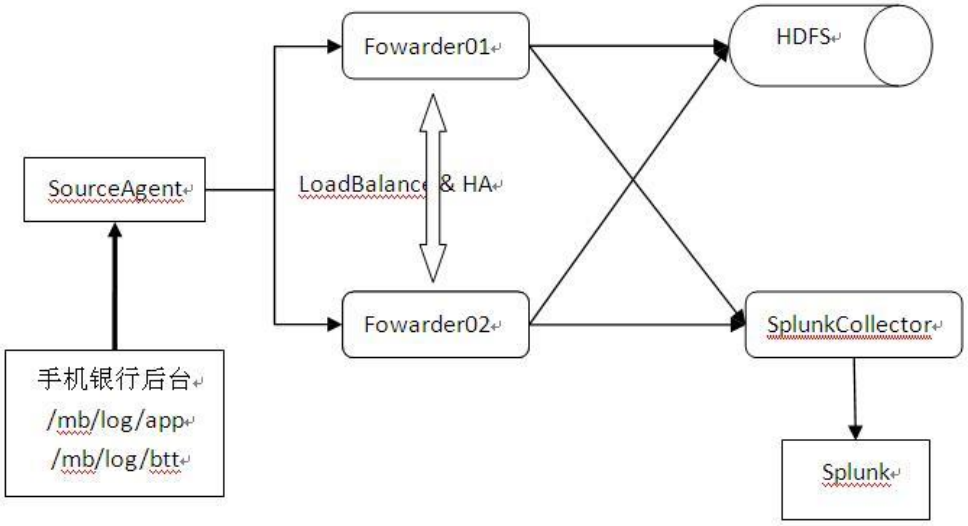
配置说明

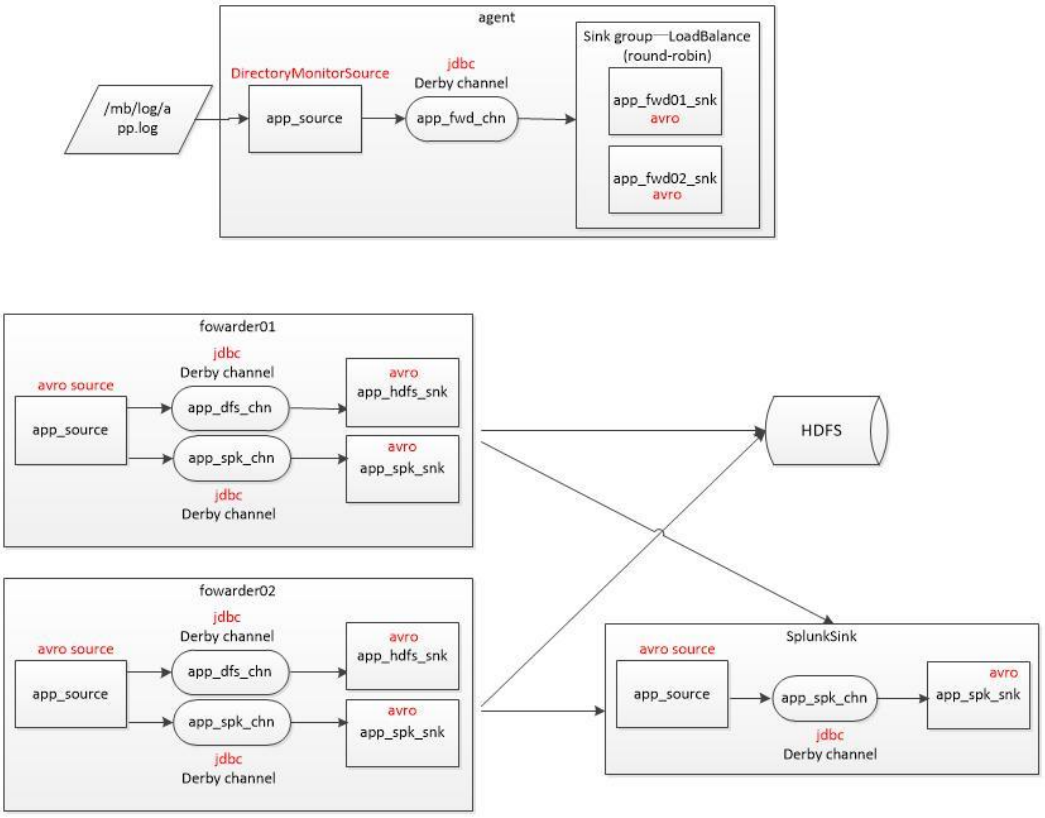
DirectoryMonitorSource 作为 flume 的 source 端，可配置参数如下：

参数	取值	说明
type	com.minsheng.flume.source.MonitorDirectorySource	必须写成 FQCN 这种形式；
monitor_dir	监控目录	注：不支持目录嵌套递归监控，会自动跳过目录下的子目录；
meta_store_dir	meta 存储目录	支持断点续传，需要记录 meta 信息；轻量级信息，每个文件大约 100 个字节的 meta 信息；
file_name_include_pattern	目录下文件名的白名单正则表达式；默认为 “.*”，表示所有文件（隐藏文件除外），均包含；	比如：app.log.*
file_name_exclude_pattern	目录下文件名的黑名单正则表达式； 默认： “^[.].*” 以’.’开头的文件加入黑名单	
first_line_pattern	一个 record 的第一行的正则表达式 默认：无	注：**这两个参数，必须至少设定一个，否则没有办法去区分 record 的边界；** **如果两个参数均不制定，责默认文**
last_line_pattern	一个 record 的最后一行的正	

	则表达式 默认：无	**件每一行数据为一个 record；**
file_content_include_pattern	一个 record（可能是多行数据）的白名单正则表达式 默认：”[\\s\\S]*”	**注：由于我们日志文件中存在\t,\n等字符，通常的正则表达式”.*”是不能匹配这些字符的，**因此白名单中，如果是所有内容都接受，必须写成如下的正则表达式： “[\\s\\S]*” 正则表达式的具体语法请 google 之。
file_content_exclude_pattern	一个 record（可能是多行数据）的黑名单正则表达式 默认：无	
file_check_interval_sec	多长时间对目录下 默认：5 秒	
file_send_interval_sec	多长时间发送一次数据 默认：3 秒	

针对如下的数据流向图：





其 sourceAgent 端对应的配置为：

```
agent.sources = app_source btt_source
agent.channels = app_fwd_chn btt_fwd_chn
agent.sinks = app_fwd01_snk    app_fwd02_snk btt_fwd01_snk btt_fwd02_snk

agent.singroups = app_group btt_group

agent.singroups.app_group.sinks = app_fwd01_snk app_fwd02_snk
agent.singroups.app_group.processor.type = load_balance
agent.singroups.app_group.processor.backoff = true
agent.singroups.app_group.processor.selector = round_robin

agent.singroups.btt_group.sinks = btt_fwd01_snk btt_fwd02_snk
agent.singroups.btt_group.processor.type = load_balance
agent.singroups.btt_group.processor.backoff = true
agent.singroups.btt_group.processor.selector = round_robin

# For each one of the sources, the type is defined
agent.sources.app_source.type = com.minsheng.flume.source.DirectoryMonitorSource
agent.sources.app_source.channels = app_fwd_chn
agent.sources.app_source.monitor_dir = /mb/log/app/
agent.sources.app_source.first_line_pattern= ^\\[\\[[\\s\\S]*
```

```
agent.sources.app_source.file_name_include_pattern = app.log.*
agent.sources.app_source.file_check_interval_sec = 1
agent.sources.app_source.file_send_interval_sec = 2
agent.sources.app_source.file_content_include_pattern = [\\s\\S]*

agent.sources.app_source.interceptors = i1 i2
agent.sources.app_source.interceptors.i1.type =
org.apache.flume.sink.solr.morphline.UUIDInterceptor$Builder
agent.sources.app_source.interceptors.i1.headerName = id
agent.sources.app_source.interceptors.i1.preserveExisting = true

agent.sources.app_source.interceptors.i2.type = host
agent.sources.app_source.interceptors.i2.hostHeader = ip
agent.sources.app_source.interceptors.i2.preserveExisting = false
agent.sources.app_source.interceptors.i2.useIP = true

# btt_source
agent.sources.btt_source.type = com.minsheng.flume.source.DirectoryMonitorSource
agent.sources.btt_source.channels = btt_fwd_chn
agent.sources.btt_source.monitor_dir = /mb/log/btt
agent.sources.btt_source.first_line_pattern= ^\\[[A-Z]{4,5}[\\s\\S]*
agent.sources.btt_source.file_name_include_pattern = btt.log.*
agent.sources.btt_source.file_check_interval_sec = 1
agent.sources.btt_source.file_send_interval_sec = 2
agent.sources.btt_source.file_content_include_pattern = [\\s\\S]*

agent.sources.btt_source.interceptors = i1 i2
agent.sources.btt_source.interceptors.i1.type =
org.apache.flume.sink.solr.morphline.UUIDInterceptor$Builder
agent.sources.btt_source.interceptors.i1.headerName = id
agent.sources.btt_source.interceptors.i1.preserveExisting = true

agent.sources.btt_source.interceptors.i2.type = host
agent.sources.btt_source.interceptors.i2.hostHeader = ip
agent.sources.btt_source.interceptors.i2.preserveExisting = false
agent.sources.btt_source.interceptors.i2.useIP = true

# Each sink's type must be defined
agent.sinks.app_fwd01_snk.type = avro
agent.sinks.app_fwd01_snk.channel = app_fwd_chn
agent.sinks.app_fwd01_snk.hostname = 197.3.187.88
agent.sinks.app_fwd01_snk.port = 11121

# Each sink's type must be defined
```

```
agent.sinks.app_fwd02_snk.type = avro
agent.sinks.app_fwd02_snk.channel = app_fwd_chn
agent.sinks.app_fwd02_snk.hostname = 197.3.187.89
agent.sinks.app_fwd02_snk.port = 11121

# Each sink's type must be defined
agent.sinks.btt_fwd01_snk.type = avro
agent.sinks.btt_fwd01_snk.channel = btt_fwd_chn
agent.sinks.btt_fwd01_snk.hostname = 197.3.187.88
agent.sinks.btt_fwd01_snk.port = 11131

# Each sink's type must be defined
agent.sinks.btt_fwd02_snk.type = avro
agent.sinks.btt_fwd02_snk.channel = btt_fwd_chn
agent.sinks.btt_fwd02_snk.hostname = 197.3.187.89
agent.sinks.btt_fwd02_snk.port = 11131

# Each channel's type is defined.
agent.channels.app_fwd_chn.type = memory
agent.channels.app_fwd_chn.capacity = 1000
agent.channels.app_fwd_chn.transactionCapacity = 1000

agent.channels.btt_fwd_chn.type = memory
agent.channels.btt_fwd_chn.capacity = 1000
agent.channels.btt_fwd_chn.transactionCapacity = 1000

agent.channels.weblogic_fwd_chn.type = memory
agent.channels.weblogic_fwd_chn.capacity = 1000
agent.channels.weblogic_fwd_chn.transactionCapacity = 1000
```