2018-4-18 databaseSink

小書匠

目录

背景及功能说明

Apache Flume

> flume的说明不赘述，官方文档：http://flume.apache.org/FlumeDeveloperGuide.html
> 支持版本：flume1.7.0及以上
> 支持环境：依赖flume `TailDirsource` 组件的环境限制，目前只支持linux环境测试及使用
> 使用flume组件：`TailDirSource` 、 `memory/file channel`

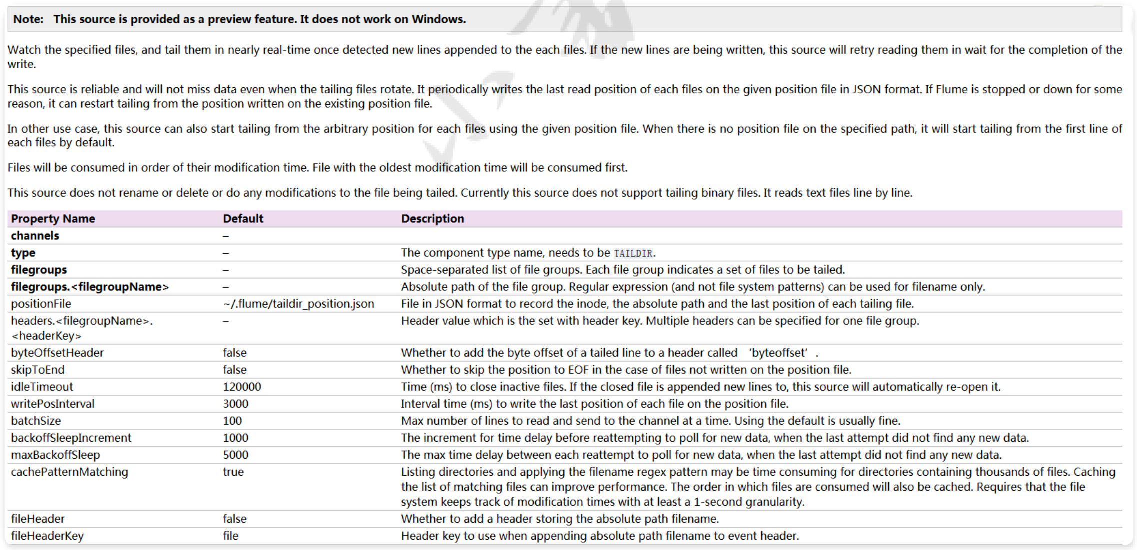
使用背景

> - 本模块核心参考官方开发者文档：http://flume.apache.org/FlumeDeveloperGuide.html
> - 对分布式服务器上的日志内容进行收集，最后汇集到一个集中处理点，并对数据进行持久化、清洗、汇总等消费操作。本自定义sink则使用mysql作为最后的数据消费者，且为了保证吞吐量，不做任何清洗、统计等操作，直接作为一个数据收集者，mysql作为数据收集存放处。

使用流程

1.flume模块source组件的选用和配置`TailDirSource`

> 该source的完整配置说明如下（截自flume官网）：



Note: This source is provided as a preview feature. It does not work on Windows.

Watch the specified files, and tail them in nearly real-time once detected new lines appended to the each files. If the new lines are being written, this source will retry reading them in wait for the completion of the write.

This source is reliable and will not miss data even when the tailing files rotate. It periodically writes the last read position of each files on the given position file in JSON format. If Flume is stopped or down for some reason, it can restart tailing from the position written on the existing position file.

In other use case, this source can also start tailing from the arbitrary position for each files using the given position file. When there is no position file on the specified path, it will start tailing from the first line of each files by default.

Files will be consumed in order of their modification time. File with the oldest modification time will be consumed first.

This source does not rename or delete or do any modifications to the file being tailed. Currently this source does not support tailing binary files. It reads text files line by line.

Property Name	Default	Description
channels	–	
type	–	The component type name, needs to be `TAILDIR`.
filegroups	–	Space-separated list of file groups. Each file group indicates a set of files to be tailed.
filegroups.<filegroupName>	–	Absolute path of the file group. Regular expression (and not file system patterns) can be used for filename only.
positionFile	~/.flume/taildir_position.json	File in JSON format to record the inode, the absolute path and the last position of each tailing file.
headers.<filegroupName>.<headerKey>	–	Header value which is the set with header key. Multiple headers can be specified for one file group.
byteOffsetHeader	false	Whether to add the byte offset of a tailed line to a header called ‘byteoffset’ .
skipToEnd	false	Whether to skip the position to EOF in the case of files not written on the position file.
idleTimeout	120000	Time (ms) to close inactive files. If the closed file is appended new lines to, this source will automatically re-open it.
writePosInterval	3000	Interval time (ms) to write the last position of each file on the position file.
batchSize	100	Max number of lines to read and send to the channel at a time. Using the default is usually fine.
backoffSleepIncrement	1000	The increment for time delay before reattempting to poll for new data, when the last attempt did not find any new data.
maxBackoffSleep	5000	The max time delay between each reattempt to poll for new data, when the last attempt did not find any new data.
cachePatternMatching	true	Listing directories and applying the filename regex pattern may be time consuming for directories containing thousands of files. Caching the list of matching files can improve performance. The order in which files are consumed will also be cached. Requires that the file system keeps track of modification times with at least a 1-second granularity.
fileHeader	false	Whether to add a header storing the absolute path filename.
fileHeaderKey	file	Header key to use when appending absolute path filename to event header.

TailDirSource

> 官方example

```
a1.sources = r1
a1.channels = c1
a1.sources.r1.type = TAILDIR
a1.sources.r1.channels = c1
a1.sources.r1.positionFile = /var/log/flume/taildir_position.json
a1.sources.r1.filegroups = f1 f2
a1.sources.r1.filegroups.f1 = /var/log/test1/example.log
a1.sources.r1.headers.f1.headerKey1 = value1
```

```
a1.sources.r1.filegroups.f2 = /var/log/test2/.log.
a1.sources.r1.headers.f2.headerKey1 = value2
a1.sources.r1.headers.f2.headerKey2 = value2-2
a1.sources.r1.fileHeader = true
```

- TailDirSource的配置特殊约定两个headkey-value
  - `processor`：是一个headKey。为该文件组配置一个sink的处理器。也可理解为标识将被`processor`对应的value代表的处理器处理
  - `signal`：是一个headKey。该文件在分布式中的唯一标识。用于断行续接。因为log4J等日志组件开启缓存打印日志时可能打印出来的日志是不完整的一行。

> 一个典型的`TailDir`配置：

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

```
agent1.sources.source1.type = TAILDIR
agent1.sources.source1.channels = channel1
agent1.sources.source1.channels.skipToEnd = False
agent1.sources.source1.positionFile = ./taildir_position.json
agent1.sources.source1.filegroups = f1 f2 f3 f4
agent1.sources.source1.filegroups.f1 = /usr/local/apache-tomcat-7.0.85/logs/statistic/gem_accountChange*.log
agent1.sources.source1.headers.f1.processor = accountChange
agent1.sources.source1.headers.f1.signal = accountChange_1
agent1.sources.source1.filegroups.f2 = /usr/local/apache-tomcat-7.0.85/logs/statistic/gem_flumeTest*.log
agent1.sources.source1.headers.f2.processor = flumeTest
agent1.sources.source1.headers.f2.signal = flumeTest_1
agent1.sources.source1.filegroups.f3 = /usr/local/apache-tomcat-7.0.85/logs/statistic/gem_roleChange*.log
agent1.sources.source1.headers.f3.processor = roleChange
agent1.sources.source1.headers.f3.signal = roleChange_1
agent1.sources.source1.filegroups.f4 = /usr/local/apache-tomcat-7.0.85/logs/statistic/gem_roomInOut*.log
agent1.sources.source1.headers.f4.processor = roomInOut
agent1.sources.source1.headers.f4.signal = roomInOut_1
agent1.sources.source1.fileHeader = true
```

2.flume模块的channel组件的选用和配置

> channel的选用依照需求，本自定义组件无依赖

3.flume模块的sink组件选用和配置-本项目自定义sink

> 需要配置自定义的sink
> 自定义sink的配置官方文档说明：
> A custom sink is your own implementation of the Sink interface. A custom sink's class and its dependencies must be included in the agent's classpath when starting the Flume agent. The type of the custom sink is its FQCN. Required properties are in bold.

Property Name	Default	Description
channel	-	
type	-	The component type name, needs to be your FQCN

官方案例：Example for agent named a1:

a1.channels = c1

a1.sinks = k1
a1.sinks.k1.type = org.example.MySink
a1.sinks.k1.channel = c1

- 本自定义sink的配置说明如下:

Property Name	Default	Description
channel	-	
type	-	The component type name, needs to be your FQCN like: com.xs.fun.sink.BaseAdminMysqlSink
jdbcUrl	-	jdbc url like：jdbc:mysql://localhost:3306/table_name
driverClass Name	com.mysql.jdbc. Driver	
username	-	database connect name
password	-	database connect passward
batchSize	100	batch process event size
initialSize		database pool initialSize
maxActive		database pool maxActive
maxIdle		database pool maxIdle
minIdle		database pool minIdle
maxWait		database pool minIdle

- 一个典型的配置案例 其中 account-change-sink是sink名

> agent1.sinks.account-change-sink.type=com.xs.fun.sink.BaseAdminMysqlSink
> agent1.sinks.account-change-sink.channel=channel1
> agent1.sinks.account-change-sink.batchSize=100
> agent1.sinks.account-change-sink.jdbcUrl=jdbc:mysql://localhost:3306/g01_admin
> agent1.sinks.account-change-sink.driverClassName=com.mysql.jdbc.Driver
> agent1.sinks.account-change-sink.username=root
> agent1.sinks.account-change-sink.password=123456
> agent1.sinks.account-change-sink.initialSize=
> agent1.sinks.account-change-sink.maxActive=
> agent1.sinks.account-change-sink.maxIdle=
> agent1.sinks.account-change-sink.minIdle=
> agent1.sinks.account-change-sink.maxWait=

> 以上三个步骤就配置完了flume节点相关配置

4.两种方式编写processor

> 采集的数据需要指定processor来处理，并存放到指定数据表。此时有两种方法快速开发来处理数据。

- 新建自己的项目，然后把本项目打好的jar包放到buildPath中，然后进行开发
- 直接把个人的项目工程maven导入，然后开始快速开发

开发流程

- 采集的数据是json格式数据且每一行每个属性key都对应数据表中的column名称，那么可以直接配置使用默认的processor，不用编写任何代码。（建议）

- 采集的数据需要进行一些字段或内容过滤，则可以自定义一个processor，继承指定processor即可。

项目源码中至少有一个自定义的processor的案例，可以参考

- 两个配置文件
    - `defaultProcessorConfig.properties`：使用默认processor的配置。
        - 配置使用哪个默认的processsor，并指定数据表名等基本属性

> 典型的配置案例：
> processor.roomInOut.tableName=t_remote_temp_room_duration
> processor.roomInOut.tableIdName=id
> processor.roomInOut.methodType=insert

> 说明：
> `processor.roomInOut` 对应 `TailDir source` 中的 `headKey.headValue`。
> 三个属性：`tableName` 声明数据存储的表名、`tableIdName` 声明表中的唯一ID列名、`methodType` 声明使用processor的哪种操作方法（将会开放crud方法，目前只insert开放）

-
    - `customerProcessorConfig.properties`：自定义processor的配置。
        - 每个processor都需要配置。方能在flume启动时加载processor（日后将会优化）。

> 典型的配置内容案例：
> processor.accountChange=com.xs.fun.processor.AccountChangeEventProcessor
> 说明：
> `processor.accountChange` 对应 `tailDir source` 中的 `headkey.headValue`，`com.xs.fun.processor.AccountChangeEventProcessor` 声明该processor的全路径

> 以上则是自定义sink的过程

5.部署

> 自定sink配置好了后，打成jar包。该jar是运行在数据持久化一段的flume节点上。放到flume指定目录，官方建议：
> The `plugins.d` directory is located at `$FLUME_HOME/plugins.d`. At startup time, the flume-ng start script looks in the `plugins.d` directory for plugins that conform to the below format and includes them in proper paths when starting up java.
> Directory layout for plugins

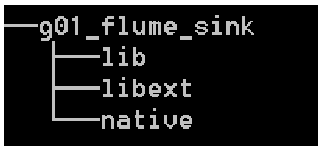
Each plugin (subdirectory) within plugins.d can have up to three sub-directories:

```
lib - the plugin’s jar(s)
libext - the plugin’s dependency jar(s)
native - any required native libraries, such as .so files
```

> Example of two plugins within the plugins.d directory:
> plugins.d/
> plugins.d/custom-source-1/
> plugins.d/custom-source-1/lib/my-source.jar
> plugins.d/custom-source-1/libext/spring-core-2.5.6.jar
> plugins.d/custom-source-2/
> plugins.d/custom-source-2/lib/custom.jar
> plugins.d/custom-source-2/native/gettext.so

- 说明：官方建议是在flume安装目录下新建一个plugins.d的文件夹，然后在该文件夹新建一个sink分类如：test。然后在test中的目录结构：lib（存放自己打的jar）、libext（自定义的jar依赖的jar包）、native（自定义的jar依赖的本地库）

- 一个典型的目录：



```
──g01_flume_sink
    ├──lib
    ├──libext
    └──native
```

6.启动

- 单flume节点启动：flume采集和处理在同一台服务器。则直接启动，可以看到日志打印初始化processor的信息，并且对使用默认processor的配置进行数据库表的配置检测
- 多节点flume启动：flume采集位置和处理位置不在同一台服务器。则先启动处理位置上的flume，启动成功后再启动采集位置上的flume

7.重启

> 当需要添加processor时等情况需要停机flume重启flume时，停机或重启采集网络上的任意一flume服务，都不会造成数据丢失。这是由flume的自身特性决定。