**Spark Streaming 基于拉方式处理 Flume-ng 数据源(Pull-based Approach using a Custom Sink)**

目录

[**Spark Streaming 基于拉方式处理 Flume-ng 数据源(Pull-based Approach using a Custom Sink)** 1](#_Toc433291608)

[1. 使用自定义Sink的拉方式 1](#_Toc433291609)

[2. 配置Spark Streaming程序 2](#_Toc433291610)

Flume-ng聚集和移动大量日志数据的分布式的，有效的服务。这里我们解释怎样配置Flume-ng 和Spark Streaming来从Flume获取数据。这里介绍第二种方法。

# 1. 使用自定义Sink的拉方式

不是Flume直接推送数据到SparkStreaming，这种方法运行了一个如下所示的Flume Sink。

1.  Flume将数据推送到Sink中，然后数据在此处缓存。

2. Spark Streaming使用一个可靠的Flume接收器和操作从Sink中拉取数据。只有在Spark Streaming接收到数据并且把数据复制后才认为操作成功。

这个方法比前面的方法提供了强大的可靠性和容错保证。然而，这需要配置Flume运行一个自定义Sink。下面是配置步骤。

1.1一般需求

选择一台在Flume代理中运行自定义Sink的机器。Flume其余的管道被配置为向那个代理发送数据。Spark集群中的机器都能连接到运行自定义Sink的那台机器上。

1.2配置Flume

在选定的机器上配置Flume需要如下的两步。

1. 添加如下的JAR包到要运行自定义Sink的机器中的Flume的classpath中（这里我把如下jar放在/usr/lib/flume-ng/lib/目录下）

spark-streaming-flume-sink\_2.10-1.3.0-cdh5.4.3.jar

commons-lang3-3.3.2.jar

B. 配置文件：在那台机器上，通过下面的配置文件配置Flume代理发送数据到一个Avro Sink中。

agent.sinks = spark

agent.sinks.spark.type = org.apache.spark.streaming.flume.sink.SparkSink

agent.sinks.spark.hostname = <hostname of the local machine>

agent.sinks.spark.port = 41444

agent.sinks.spark.channel = memoryChannel

# 2. 配置Spark Streaming程序

2.1编程：在流处理程序的代码中，引入FlumeUtils并如下创建一个输入DStream流。这里给出Spark Java 程序例子

**import** java.util.Arrays;

**import** org.apache.spark.SparkConf;

**import** org.apache.spark.api.java.function.FlatMapFunction;

**import** org.apache.spark.api.java.function.Function;

**import** org.apache.spark.examples.streaming.StreamingExamples;

**import** org.apache.spark.streaming.\*;

**import** org.apache.spark.streaming.api.java.\*;

**import** org.apache.spark.streaming.flume.FlumeUtils;

**import** org.apache.spark.streaming.flume.SparkFlumeEvent;

**import** org.apache.spark.storage.StorageLevel;

**public** **class** SparkStreamingFlume2 {

**public** SparkStreamingFlume2() {

}

**public** **static** **void** main(String[] args) {

**if** (args.length != 2) {

System.*err*.println("Usage: JavaFlumeEventCount1 <host> <port>");

System.*exit*(1);

}

StreamingExamples.*setStreamingLogLevels*();

String host = args[0];

**int** port = Integer.*parseInt*(args[1]);

Duration batchInterval = **new** Duration(2000);

SparkConf sparkConf = **new** SparkConf().setAppName("JavaFlumeEventCount");

JavaStreamingContext ssc = **new** JavaStreamingContext(sparkConf, batchInterval);

StorageLevel storagelevel = StorageLevel.*MEMORY\_ONLY*();

JavaReceiverInputDStream<SparkFlumeEvent> flumeStream = FlumeUtils.*createPollingStream*(ssc, host, port,storagelevel);

flumeStream.count();

flumeStream.count().map(**new** Function<Long, String>() {

@Override

**public** String call(Long in) {

**return** "Received " + in + " flume events.";

}

}).print();

ssc.start();

ssc.awaitTermination();

}

2.2启动Flume这里主要需要添加 spark-streaming-flume-sink\_2.10-1.3.0-cdh5.4.3.jar和commons-lang3-3.3.2.jar 到$FLUME\_HOME/lib目录下

Flume-ng agent –c /etc/flume-ng/conf –f /etc/flume-ng/conf/flume.conf –Dflume.root.logger=DEBUG,console –n agent02

2.3提交Spark，这里需要注意的添加必要的jar包，可以在提交的时候加上--jars来指定相关的jar包，也可以在sc中调用addJar()添加

spark-submit --master spark://udh-spark-test-04:7077 --class SparkStreamingFlume2 --jars /root/spark-streaming-flume-sink\_2.10-1.3.0-cdh5.4.3.jar,/usr/lib/flume-ng/lib/flume-ng-sdk-1.5.0-cdh5.4.3.jar,/usr/lib/spark/lib/spark-examples-1.3.0-cdh5.4.3-hadoop2.6.0-cdh5.4.3.jar,/usr/lib/spark/lib/spark-assembly-1.3.0-cdh5.4.3-hadoop2.6.0-cdh5.4.3.jar /root/flume-test02.jar udh-spark-test-03 41444