1. Flume on HDFS
2. Flume推送数据并实战
3. 原理绘图解析

如果数据是大规模或者持续不断的产生的话那么就使用kafka

kafka的吞吐力已经是无敌的了

有时候数据量大，有时候不产生数据，那么使用kafka就没有必要了

处理数据的节点和接收数据必须在一个节点上

*\*/***def** createStream (  
 ssc: StreamingContext,  
 hostname: String,  
 port: Int,  
 storageLevel: StorageLevel = StorageLevel.*MEMORY\_AND\_DISK\_SER\_2*

*//既在内存又在磁盘中同时序列化两份副本* ): ReceiverInputDStream[SparkFlumeEvent] = {  
 *createStream*(ssc, hostname, port, storageLevel, **false**)  
}

SparkStreaming太闲或者来不及消费

另外一方面，让SparkStreaming从flume拉数据，能消费多少就拉多少

面试题：sparkStreaming一直产生RDD，RDD都是对象，为什么要清理，什么时候清理，要维持状态怎么办？

SparkStreaming在每次运行完一个作业的时候，就会触发一个消息给他的事件循环体，在消息循环时，会不断的清理RDD,如果他发现你是要做window状态保存，他会提前做checkpoint,即使在内存中清理掉了，也会在磁盘上，这样如果你到时候需要状态，window或者时间操作的时候，还是可以聚合数据

上节课要求大家自己安装配置Flume，并且测试数据的传输。我昨天是要求传送的HDFS上。

文件配置：

~/.bashrc:

export FLUME\_HOME=/usr/local/flume/apache-flume-1.6.0-bin

export FLUME\_CONF\_DIR=$FLUME\_HOME/conf

PATH中增加：${FLUME\_HOME}/bin;

拷贝conf/flume-conf.properties.template，更名为conf/flume-cong.properties，只写以下内容：

agent1表示代理名称

agent1.sources=source1

agent1.sinks=sink1

agent1.channels=channel1

#配置source1

agent1.sources.source1.type=spooldir

agent1.sources.source1.spoolDir=/usr/local/flume/tmp/TestDir

agent1.sources.source1.channels=channel1

agent1.sources.source1.fileHeader = false

agent1.sources.source1.interceptors = i1

agent1.sources.source1.interceptors.i1.type = timestamp

#配置sink1

agent1.sinks.sink1.type=hdfs

agent1.sinks.sink1.hdfs.path=hdfs://master:9000/library/flume

agent1.sinks.sink1.hdfs.fileType=DataStream

agent1.sinks.sink1.hdfs.writeFormat=TEXT

agent1.sinks.sink1.hdfs.rollInterval=1

agent1.sinks.sink1.channel=channel1

agent1.sinks.sink1.hdfs.filePrefix=%Y-%m-%d

#agent1.sinks.sink1.type=avro

#agent1.sinks.sink1.channel=channel1

#agent1.sinks.sink1.hostname=Master

#agent1.sinks.sink1.port=9999

#配置channel1

agent1.channels.channel1.type=file

agent1.channels.channel1.checkpointDir=/usr/local/flume/tmp/checkpointDir

agent1.channels.channel1.dataDirs=/usr/local/flume/tmp/dataDirs

flume-env.sh配置:

# export JAVA\_HOME=/usr/lib/jvm/java-6-sun

export JAVA\_HOME=/usr/local/jdk/jdk1.8.0\_60

# Give Flume more memory and pre-allocate, enable remote monitoring via JMX

# export JAVA\_OPTS="-Xms100m -Xmx2000m -Dcom.sun.management.jmxremote"

export JAVA\_OPTS="-Xms100m -Xmx2000m -Dcom.sun.management.jmxremote"

建立文件夹 /usr/local/flume/tmp/TestDir。

在hdfs上建立/library/flume文件夹。

flume的bin文件夹下启动Flume:

./flume-ng agent -n agent1 -c conf -f /usr/local/flume/apache-flume-1.6.0-bin/conf/flume-conf.properties -Dflume.root.logger=DEBUG,console

在/usr/local/flume/tmp/TestDir下，拷入测试用的文件，比如：NOTICE

flume 控制台会有一些相关信息：

16/04/22 11:03:49 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /usr/local/flume/tmp/TestDir/NOTICE to /usr/local/flume/tmp/TestDir/NOTICE.COMPLETED

16/04/22 11:03:51 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false

16/04/22 11:03:51 INFO hdfs.BucketWriter: Creating hdfs://master:9000/library/flume/2016-04-22.1461294231806.tmp

16/04/22 11:03:52 INFO hdfs.BucketWriter: Closing hdfs://master:9000/library/flume/2016-04-22.1461294231806.tmp

16/04/22 11:03:52 INFO hdfs.BucketWriter: Renaming hdfs://master:9000/library/flume/2016-04-22.1461294231806.tmp to hdfs://master:9000/library/flume/2016-04-22.1461294231806

可以发现本地的NOTICE文件更名为NOTICE.COMPLETED。

用浏览器查询：[http://localhost:50070/explorer.html#/library/flume](http://localhost:50070/explorer.html" \l "/library/flume)，可看到Flume把NOTICE文件传送到 HDFS的/library/flume下，文件名为 2016-04-22.1461294231806。打开文件看内容可以验证。说明当Flume指定的源文件夹中有新文件时，Flume会自动将此文件，导入到Flume配置时指定的HDFS文件夹中。

一般正常业务情况下，应该是把Flume的数据放到Kafka中，然后让不同的数据消费者去消费数据。如果要在Flume和Kafka两者间做选择的话，则要看业务中数据是否持续不断的产生。如果是这样，则应该选择Kafka。如果产生的数据时大时小，甚至有些时间没有数据，则没必要用Kafka，可以节省资源。

2. Flume推送数据到Spark Streaming实战

现在我们不把Flume的数据导入到HDFS中，而是把数据推送到Spark Streaming中。

修改conf/flume-cong.properties文件，改导入到HDFS，变为推送到Spark Streaming。

#配置sink1

#agent1.sinks.sink1.type=hdfs

#agent1.sinks.sink1.hdfs.path=hdfs://master:9000/library/flume

#agent1.sinks.sink1.hdfs.fileType=DataStream

#agent1.sinks.sink1.hdfs.writeFormat=TEXT

#agent1.sinks.sink1.hdfs.rollInterval=1

#agent1.sinks.sink1.channel=channel1

#agent1.sinks.sink1.hdfs.filePrefix=%Y-%m-%d

agent1.sinks.sink1.type=avro

agent1.sinks.sink1.channel=channel1

agent1.sinks.sink1.hostname=Master

agent1.sinks.sink1.port=9999

编写Spark Streaming应用的Java程序：

public class FlumePushData2SparkStreaming {

    public static void main(String[] args) {

      SparkConf conf = new SparkConf().setMaster("local[4]").setAppName("FlumePushDate2SparkStreaming");

        JavaStreamingContext jsc = new JavaStreamingContext(conf, Durations.seconds(30));

        JavaReceiverInputDStream lines = FlumeUtils.createStream(jsc,"Master", 9999);

        // 注意此处输入的event类型是SparkFlumeEvent类型。

        JavaDStream<String> words = lines.flatMap(new FlatMapFunction<SparkFlumeEvent, String>() {

            @Override

            public Iterable<String> call(SparkFlumeEvent event) throws Exception {

                String line = new String(event.event().getBody().array());

                return Arrays.asList(line.split(" "));

            }

        });

        JavaPairDStream<String, Integer> pairs = words.mapToPair(new PairFunction<String, String, Integer>() {

            @Override

            public Tuple2<String, Integer> call(String word) throws Exception {

                return new Tuple2<String, Integer>(word, 1);

            }       });

        JavaPairDStream<String, Integer> wordsCount = pairs.reduceByKey(new Function2<Integer, Integer, Integer>() {

           @Override

            public Integer call(Integer v1, Integer v2) throws Exception {

                return v1 + v2;

            }

        });

      wordsCount.print();

        jsc.start();

        jsc.awaitTermination();

       jsc.close();

    }

}

代码中用到了FlumeUtils。我们剖析一下代码中用到的FlumeUtils

