Spark Streaming

Spark Streaming通过将流数据按指定的时间片累积成RDD，然后将每个RDD进行批处理，进而实现大规模的流处理，Storm处理数据方式是以条为单位，而Spark Streaming基于单位时间处理数据的。

Spark Streaming支持从多种数据源获取数据，包括Kafka、Flume、Twitter、ZeroMQ及TCP sockets等，从数据源获取数据之后，可以使用诸如map、Reduce、Join等高级函数进行复杂算法处理，最后把处理结果存储到文件系统，数据流图如下所示：



Spark Streaming在内部的处理机制是，接收实时流的数据，并根据一定的时间间隔拆分成一批批的数据，然后通过Spark Engine处理这些批数据，最终得到处理后的一批批结果数据，如下图所示：



与Spark批数据处理内核对应的RDD实例，对应流数据Dstream可以看做是一组RDDs，及RDD序列，在流数据分成批后，通过一个FIFO队列，然后通过Spark Engine从该队列中依次取出批数据，再把批数据封装成RDD，然后进行处理。

# SparkStreaming WordCount

在Example中有streaming WordCount程序，执行命令如下所示：

*$ ./bin/run-example streaming.NetworkWordCount localhost 9999*

上面的命令，本地端口9999，接收获取的字符串，然后进行处理，使用nc来传输，结果如下所示：

|  |  |
| --- | --- |
| *$ nc -lk 9999*  *test spark* | *Time: 1499251649000 ms*  *-------------------------------------------*  *(test,1)*  *(spark,1)*  *scheduler.JobScheduler: Finished job streaming job*  *(execution: 0.062 s)* |

# Spark Streaming Java API

测试程序如下所示：

*package com.fys.streaming;*

*import org.apache.spark.api.java.StorageLevels;*

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

*import org.apache.spark.api.java.function.Function2;*

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

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

*import org.apache.spark.SparkConf;*

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

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

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

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

*import scala.Tuple2;*

*import java.util.Arrays;*

*import java.util.Iterator;*

*import java.util.regex.Pattern;*

*public class JavaNetworkWordCount {*

*private static final Pattern SPACE = Pattern.compile(" ");*

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

*if (args.length < 2) {*

*System.err.println("Usage: JavaNetworkWordCount <hostname> <port>");*

*System.exit(1);*

*}*

*// Create the context with a 1 second batch size*

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

*JavaStreamingContext ssc = new JavaStreamingContext(sparkConf, Durations.seconds(1));*

*//Create a JavaReceiverInputStream on target ip: port*

*JavaReceiverInputDStream<String> lines = ssc.socketTextStream(*

*args[0], Integer.parseInt(args[1]) , StorageLevels.MEMORY\_AND\_DISK\_SER*

*);*

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

*@Override*

*public Iterator<String> call(String x) {*

*return Arrays.asList(SPACE.split(x)).iterator();*

*}*

*});*

*JavaPairDStream<String, Integer> wordCounts = words.mapToPair(*

*new PairFunction<String, String, Integer>() {*

*@Override*

*public Tuple2<String, Integer> call(String s) {*

*return new Tuple2<>(s, 1);*

*}*

*}).reduceByKey(new Function2<Integer, Integer, Integer>() {*

*@Override*

*public Integer call(Integer i1, Integer i2) {*

*return i1 + i2;*

*}*

*});*

*wordCounts.print();*

*ssc.start();*

*ssc.awaitTermination();*

*}*

*}*

编译后，执行命令如下：

*bin/spark-submit --class com.fys.streaming.JavaNetworkWordCount --master spark://fys1.cmss.com:7077 ~/sparkexamples-2.1.0.jar fys1.cmss.com 9999*

结果如下：

|  |  |
| --- | --- |
| *$ nc -lk 9999*  *hehe haha* | *Time: 1499332619000 ms*  *-------------------------------------------*  *(hehe,1)*  *(haha,1)*  *scheduler.JobScheduler: Finished job streaming job*  *(execution: 0.062 s)* |

http://spark.apache.org/docs/latest/streaming-programming-guide.html