**目录**

1、Eclipse配置Log4j2

2、启动Kafka

3、Kafka -> Spark Streaming

**1、Eclipse配置Log4j2**

**（1）pom.xml添加依赖**

|  |
| --- |
| <!-- log4j2 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter</artifactId>  <!--exclusions部分是为了将spring boot自带的log4j配置替换掉 是为了能够得到更为清晰的日志信息 -->  <exclusions>  <exclusion>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-logging</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-log4j2</artifactId>  </dependency>    <!-- kafka -->  <dependency>  <groupId>org.apache.kafka</groupId>  <artifactId>kafka-clients</artifactId>  <version>0.9.0.1</version>  </dependency> |

**（2）清除SpringBoot启动时在控制台显示的图标**

因为测试过SpringBoot的图标的日志级别好像非常高，所以就不用接下来的设定日志显示级别来过滤掉了，直接在这里清除它。

在application.java中：

|  |
| --- |
| @SpringBootApplication  public class Application {    public static void main(String[] args) {  //SpringApplication.run(Application.class, args);  //清除spring boot的图标  SpringApplication springApplication = new SpringApplication(Application.class);  springApplication.setBannerMode(Banner.Mode.OFF);  springApplication.run(args);  }  } |

**（3）加入log4j2的配置**

在src/main/resources下面添加log4j2-spring.xml文件（加载log4j2的时候，会自动到当前目录下搜索这个文件名，如果更改名字，则需去application.properties中配置）

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!--日志级别以及优先级排序: OFF > FATAL > ERROR > WARN > INFO > DEBUG > TRACE > ALL -->  <!--Configuration后面的status，这个用于设置log4j2自身内部的信息输出，可以不设置。当设置成trace时，会看到log4j2内部各种详细输出 -->  <!--monitorInterval：Log4j2能够自动检测修改配置 文件和重新配置本身，设置间隔秒数 -->  <configuration status="WARN" monitorInterval="30">    <!--先定义所有的appender -->  <appenders>  <!--输出控制台的配置 -->  <console name="Console" target="SYSTEM\_OUT">  <!--<JSONLayout compact="true" eventEol="true"/> -->  <!--输出日志的格式：只用了取具体日志的格式，去掉了多余信息的显示，例如时间和类 -->  <PatternLayout pattern="%m%n" />  </console>    <!--输出到Kafka -->  <Kafka name="KAFKA" topic="test01">  <ThresholdFilter level="warn" onMatch="ACCEPT" onMismatch="DENY" />  <!-- kafka 会自动换行，所以不需要加%n -->  <PatternLayout pattern="%m" />  <Property name="bootstrap.servers">virhost01:9092</Property>  </Kafka>  </appenders>    <!--然后定义logger，只有定义了logger并引入的appender，appender才会生效 -->  <loggers>  <!--过滤掉spring和mybatis的一些无用的warn级别以下的信息 -->  <!--因为接下来输出的日志信息设置为了warn级别，不想被以下的信息干扰 -->  <logger name="org.springframework" level="warn"></logger>  <logger name="org.mybatis" level="warn"></logger>  <!--这里设置的是全局的日志显示级别 -->  <root level="warn">  <appender-ref ref="Console" />  <appender-ref ref="KAFKA" />  </root>  </loggers>    </configuration> |

**（4）写日志信息**

在你想要输出日志的类中，加载日志信息配置，getLogger(类名.class)：

**private static final** org.slf4j.Logger ***log*** = org.slf4j.LoggerFactory.*getLogger*(UserController.**class**);

然后就可以在该类中用***log***.warn()来写日志了。

可以先测试一下，程序运行到这里有没有输出日志里面的值，如果报错，把之前有关的flume配置信息都注释掉。

**2、启动Kafka**

（1）启动zookeeper

zkServer.sh start

（2）启动Kafka服务

cd /opt/cdh/kafka\_2.11-2.3.1

bin/kafka-server-start.sh config/server.properties

（3）创建Kafka主题（topic）

查看现有topic

bin/kafka-topics.sh --zookeeper virhost01:2181 --list

若没有topic，则创建一个topic

bin/kafka-topics.sh --zookeeper virhost01:2181 --create --replication-factor 1 --partitions 3 --topic test01

（4）启动消费者客户端

cd /opt/cdh/kafka\_2.11-2.3.1

bin/kafka-console-consumer.sh --bootstrap-server virhost01:9092 --topic test01

注意：打印在Kafka中的每一条日志后面会有一个空行，但在使用Spark Streaming 统计时不会有影响。

**3、Kafka -> Spark Streaming**

（1）IDEA创建Maven项目

New P roject 
Java 
Java EX 
Android 
IntelliJ platform plugin 
Maven 
Gradle 
Groovy 
Scala 
Kotlin 
Empty Project 
> 1.8 (java version 'I .8.0 141 
Project SDK: 
Create from archetype 
com.atlassian.maven.archetypesbamboo-plugin-archetype 
com.atlassian.maven.archetypesronfiuence-plugin-archetype 
com_atlassIan.maven.archetypesjira-plugin-archetype 
com.rfc.rnaven.archetypesjpa-maven-archetype 
de.akquinet.jbossccjbosscc-seam-archetype 
net.databinder:data-app 
net.liftweb:lift-archetype-basic 
net.liftweb:lift-archetype-blank 
net.sf.maven-har: 
ma har 
net.sf.maven •sar:maven •archetype-sar 
org.apache.camel.archetypesramel-archetype-activemq 
org.apache.camel.archetypescamel-archetype-component 
org apache. c amel. arc -archetype-java 
org.apache.camel.archetypes:canlelarchetypescala 
org-apache.camel.archerypesramel-archetype-spring 
org.apache.cocoon:cocoon-æ-archetype-block 
eates a new Camel project using Scala DSL 
Next 
New„ 
Add Archetype... 
Cancel 
Help 

（2）删除自动生成的文件

计算机生成了可选文字:
酽MyRouteBuilder
MyRouteMa•n

（3）pom.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 <http://maven.apache.org/maven-v4_0_0.xsd>">    <modelVersion>4.0.0</modelVersion>    <groupId>com.qrsx</groupId>  <artifactId>PlatFormAnalyse</artifactId>  <packaging>jar</packaging>  <version>1.0-SNAPSHOT</version>    <properties>  <scala.version>2.11.12</scala.version>  <kafka.version>0.8.2.1</kafka.version>  <spark.version>2.2.0</spark.version>  <hadoop.version>2.9.0</hadoop.version>  <hbase.version>1.4.4</hbase.version>  </properties>    <repositories>  <repository>  <id>scala-tools.org</id>  <name>Scala-Tools Maven2 Repository</name>  <url>http://scala-tools.org/repo-releases</url>  </repository>  </repositories>    <pluginRepositories>  <pluginRepository>  <id>scala-tools.org</id>  <name>Scala-Tools Maven2 Repository</name>  <url>http://scala-tools.org/repo-releases</url>  </pluginRepository>  </pluginRepositories>    <dependencies>  <dependency>  <groupId>org.scala-lang</groupId>  <artifactId>scala-library</artifactId>  <version>${scala.version}</version>  </dependency>    <dependency>  <groupId>org.apache.kafka</groupId>  <artifactId>kafka\_2.11</artifactId>  <version>${kafka.version}</version>  </dependency>    <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-client</artifactId>  <version>${hadoop.version}</version>  <exclusions>  <exclusion>  <artifactId>servlet-api</artifactId>  <groupId>javax.servlet</groupId>  </exclusion>  </exclusions>  </dependency>    <!--<dependency>-->  <!--<groupId>org.apache.hbase</groupId>-->  <!--<artifactId>hbase-clinet</artifactId>-->  <!--<version>${hbase.version}</version>-->  <!--</dependency>-->    <!--<dependency>-->  <!--<groupId>org.apache.hbase</groupId>-->  <!--<artifactId>hbase-server</artifactId>-->  <!--<version>${hbase.version}</version>-->  <!--</dependency>-->    <!--spark-->  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-streaming\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-sql\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-streaming-flume\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-streaming-flume-sink\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-streaming-kafka-0-8\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>  <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-core\_2.11</artifactId>  <version>2.2.0</version>  </dependency>    <dependency>  <groupId>net.jpountz.lz4</groupId>  <artifactId>lz4</artifactId>  <version>1.3.0</version>  </dependency>    <!--MySQL数据库驱动-->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.31</version>  </dependency>    <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  <version>3.5</version>  </dependency>    <dependency>  <groupId>commons-dbcp</groupId>  <artifactId>commons-dbcp</artifactId>  <version>1.4</version>  </dependency>  </dependencies>    <build>  <sourceDirectory>src/main/scala</sourceDirectory>  <testSourceDirectory>src/test/scala</testSourceDirectory>  <plugins>  <plugin>  <groupId>org.scala-tools</groupId>  <artifactId>maven-scala-plugin</artifactId>  <version>2.15.2</version>  <executions>  <execution>  <id>scala-compile</id>  <goals>  <goal>compile</goal>  </goals>  <configuration>  <!--includes是一个数组，包含要编译的code-->  <includes>  <include>\*\*/\*.scala</include>  </includes>  </configuration>  </execution>  <execution>  <id>scala-test-compile</id>  <goals>  <goal>testCompile</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build>  <reporting>  <plugins>  <plugin>  <groupId>org.scala-tools</groupId>  <artifactId>maven-scala-plugin</artifactId>  <configuration>  <scalaVersion>${scala.version}</scalaVersion>  </configuration>  </plugin>  </plugins>  </reporting>    </project> |

（4）scala

|  |
| --- |
| import java.util.Properties    import kafka.serializer.StringDecoder  import org.apache.spark.SparkConf  import org.apache.spark.sql.{DataFrame, SaveMode, SparkSession}  import org.apache.spark.streaming.kafka.\_  import org.apache.spark.streaming.{Durations, Seconds, StreamingContext}    object KafkaAndSparkStreaming {  def main(args: Array[String]): Unit = {  //判断args的配置是否符合标准  if (args.length != 2) {  System.err.println("Usage: KafkaDirect <ip:port> <topics>")  System.exit(1)  }    //将参数args读入到数组中  val Array(brokers, topics) = args    val sparkConf = new SparkConf().setMaster("local[2]").setAppName("KafkaAndSparkStreaming")  // 用n秒批间隔创建上下文  val ssc = new StreamingContext(sparkConf, Seconds(2))    //设置日志级别  ssc.sparkContext.setLogLevel("WARN")  ssc.checkpoint("./sscheckpoint")    // 创建kafka流与brokers和topic  val topicsSet = topics.split(" ").toSet  val kafkaParams = Map[String, String]("metadata.broker.list" -> brokers)  val messages = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder](ssc, kafkaParams, topicsSet)    //打印获取到的数据：messages.map(\_.\_2)  val mapDStream = messages.map(\_.\_2).map((\_, 1)).map(\_.\_2).map(("人数", \_))    //优化的窗口函数  val result = mapDStream.reduceByKeyAndWindow((v1: Int, v2: Int) => {  v1 + v2  }, (v1: Int, v2: Int) => {  v1 - v2  }, Durations.seconds(10), Durations.seconds(10))  result.print()    result.foreachRDD(  rdd => {  //创建一个spark Session对象  val sparkSession = SparkSession.builder().config(rdd.sparkContext.getConf).getOrCreate()  //将RDD转换为DataFrame，并且过滤掉了人数为0的时间段  import sparkSession.implicits.\_  //rdd:(人数，n)  //toDF("active\_count")：将rdd中的代表活跃人数的映射为表的active\_count列  val frame: DataFrame = rdd.filter(\_.\_2 > 0).map(\_.\_2).toDF("active\_count")  //封装用户名和口令  val properties = new Properties  properties.setProperty("user", "root")  properties.setProperty("password", "123456")  //数据库和表会自动创建  frame.write.mode(SaveMode.Append).jdbc("jdbc:mysql://virhost01:3306/shop\_platform\_analyse?characterEncoding=UTF-8", "active\_count", properties)  }  )    // 开始计算  ssc.start()  ssc.awaitTermination()  }  } |

（5）运行

Run -> Edit Configurations -> Application -> 项目名 -> Program arguments中输入：ip:port topic

Run/Debug Configurations 
Application 
KafkaAndPrintInSpark 
TestActuaI 
Templates 
Name: KafkaAndPrintlnSpark 
CJ Share through VCS 
Allow parallel run 
Configuration 
Main class: 
MM options: 
Code Coverage Logs 
Program arguments: 
Working directory: 
Environment variables: 
Redirect input from: 
Use classpath of module: 
IRE: 
Shorten command line: 
com.qrsx.KafkaDirect 
; 9092 
PlatFormAnalyse 
Include dependencies With 'Provided" scope 
(1,8 SDK of 'PlatFormAnalyse' module) 
user-local default: none - java [options] classname [args) 
o 
[3 Enable capturing form snapshots 
• Efore launch: Build, Activate tool window 
Build 
[2 Show this page [Z Activate tool window 
OK 
Cancel 

（6）打包运行

启动Spark

cd /opt/modules/spark-2.4.4-bin-hadoop2.6

sbin/start-all.sh

访问WebUI： <http://virhost01:8080>

以下语句都未成功

bin/spark-submit \

--master local[2] \

--class com.qrsx.KafkaAndSparkStreaming \

--name KafkaAndSparkStreaming \

--packages org.apache.spark:spark-streaming-kafka-0-8\_2.11:2.2.0 \

/opt/datas/\_.jar/KafkaAndSparkStreaming.jar \

virhost01:9092 test01

bin/spark-submit \

--master spark://master.virhost01:7077 \

--class com.qrsx.KafkaAndSparkStreaming \

/opt/datas/\_.jar/KafkaAndSparkStreaming.jar \

virhost01:9092 test01