**先上pom文件**

**[html]** [view plain](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992) [copy](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992)

1. **<dependencies>**
2. **<dependency>**
3. **<groupId>**org.apache.kafka**</groupId>**
4. **<artifactId>**kafka\_2.10**</artifactId>**
5. **<version>**0.10.0.0**</version>**
6. **</dependency>**
8. **<dependency>**
9. **<groupId>**org.apache.kafka**</groupId>**
10. **<artifactId>**kafka-clients**</artifactId>**
11. **<version>**0.10.0.0**</version>**
12. **</dependency>**
14. **<dependency>**
15. **<groupId>**org.apache.spark**</groupId>**
16. **<artifactId>**spark-core\_2.10**</artifactId>**
17. **<version>**1.6.0**</version>**
18. **</dependency>**
20. **<dependency>**
21. **<groupId>**org.apache.spark**</groupId>**
22. **<artifactId>**spark-streaming\_2.10**</artifactId>**
23. **<version>**1.6.0**</version>**
24. **</dependency>**
26. **<dependency>**
27. **<groupId>**org.apache.spark**</groupId>**
28. **<artifactId>**spark-streaming-kafka\_2.10**</artifactId>**
29. **<version>**1.6.0**</version>**
30. **</dependency>**
32. **<dependency>**
33. **<groupId>**org.apache.hbase**</groupId>**
34. **<artifactId>**hbase-client**</artifactId>**
35. **<version>**1.2.0**</version>**
36. **</dependency>**

模拟Kafka生产者

导包

**[html]** [view plain](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992) [copy](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992)

1. import org.apache.kafka.clients.producer.{KafkaProducer, ProducerRecord}

**[html]** [view plain](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992) [copy](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992)

1. import java.util.Properties

模拟生产者代码

**[html]** [view plain](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992) [copy](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992)

1. def main(args: Array[String]): Unit = {
2. val topic = "user\_events"
3. val brokers = "服务器名:9092"
4. val props = new Properties()
5. props.put("bootstrap.servers",brokers)
6. props.put("acks", "0")
7. props.put("retries",0.asInstanceOf[Integer])
8. props.put("batch.size", 16384.asInstanceOf[Integer])
9. props.put("linger.ms", 1.asInstanceOf[Integer])
10. props.put("buffer.memory", 33554432.asInstanceOf[Integer])
11. props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer")
12. props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer")
14. val producer = new KafkaProducer[String, String](props)
15. for(i **<-** 0 to 100){
16. producer.send(new ProducerRecord[String, String](topic, Integer.toString(i)))//参数中可以指定key值
17. }
18. producer.close()
19. }

SparkStreaming + HBase

导包

**[html]** [view plain](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992) [copy](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992)

1. import kafka.common.TopicAndPartition
2. import kafka.serializer.StringDecoder
3. import org.apache.spark.SparkConf
4. import org.apache.spark.streaming.{Seconds, StreamingContext}
5. import org.apache.hadoop.hbase.{HBaseConfiguration, HColumnDescriptor, HTableDescriptor, TableName}
6. import org.apache.hadoop.hbase.client.{Connection, ConnectionFactory, Put}
7. import org.apache.hadoop.hbase.util.Bytes

数据处理代码

**[html]** [view plain](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992) [copy](https://blog.csdn.net/Programmer_Xiaoming/article/details/78901992)

1. object KafkaSparkStreaming {
2. var hconf = HBaseConfiguration.create()
3. hconf.set("hbase.zookeeper.quorum", "rozntgtest8")
4. hconf.set("hbase.zookeeper.property.clientPort", "2181")
5. hconf.set("mapreduce.task.timeout", "120000000")
6. hconf.set("hbase.client.scanner.timeout.period", "600000000")
7. hconf.set("hbase.rpc.timeout", "600000000")
8. val conn:Connection = ConnectionFactory.createConnection(hconf)
9. def main(args: Array[String]): Unit = {
10. val conf = new SparkConf().setMaster("local[1]").setAppName("UserClickCountStat")
11. val ssc = new StreamingContext(conf, Seconds(5))
13. val topics = Set("user\_events")
14. val brokers = "主机名:9092"
15. val kafkaParams = Map[String, String](
16. "metadata.broker.list" -**>** brokers, "value.serializer"-**>** "org.apache.kafka.common.serialization.StringSerializer","key.serializer"-**>** "org.apache.kafka.common.serialization.StringSerializer")
18. val kafkaStream = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder](ssc, kafkaParams, topics)
19. kafkaStream.foreachRDD(rdd =**>** {
20. if (!conn.getAdmin.tableExists(TableName.valueOf("number"))) {
21. val descriptor = new HTableDescriptor(TableName.valueOf("number")).addFamily(new HColumnDescriptor("info"))
22. conn.getAdmin.createTable(descriptor)
23. }
24. rdd.foreachPartition(x =**>** {
25. x.foreach( a =**>** {
26. val put = new Put(Bytes.toBytes("rowkey"))
27. put.addColumn(Bytes.toBytes("info"), Bytes.toBytes(a.\_2), Bytes.toBytes(a.\_2))
28. val table = conn.getTable(TableName.valueOf("number"))
29. table.put(put)
30. })
31. })
32. })
33. ssc.start()
34. ssc.awaitTermination()
36. }

这中间有一个问题，conn必须在作为成员变量，我在测试的时候把它放在局部的时候无法序列化，报错:task not serializable。