SparkStreaming、窗口函数

Spark Streaming

离线计算、实时计算与流式处理介绍：

流处理、实时计算属于一类的，即计算在数据变化时，都是在数据的计算实时性要求比较高的场景，能够实时的响应结果，一般在秒级，Yahoo的S4,twiter的storm以及sparkStreaming都属于流处理和实时计算一类的。

离线计算一般是批量处理数据库的过程，比如利用[Hadoop](http://lib.csdn.net/base/hadoop" \o "Hadoop知识库" \t "http://blog.csdn.net/yuxin6866/article/details/_blank)的mapreduce；内存计算[Spark](http://lib.csdn.net/base/spark" \o "Apache Spark知识库" \t "http://blog.csdn.net/yuxin6866/article/details/_blank)类似于mapreduce， 不过是数据是放在在内存中，效率比较高。

场景举例：1.统计去年到今年一共收到了多少订单，收到了多少钱 ？（离线计算）

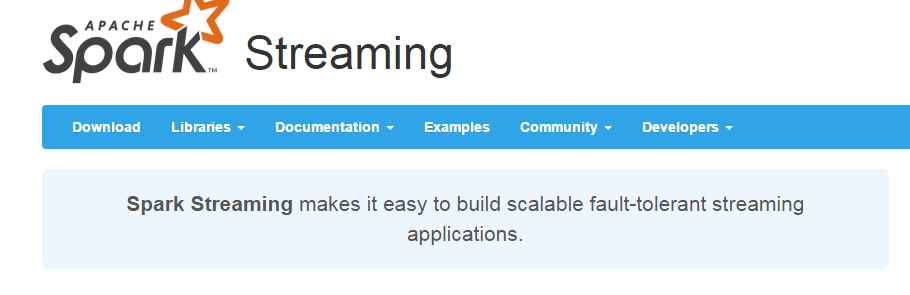
2.电商购物，先点击电脑，下个页面推荐电脑相关的商品 (实时计算)

3.互联网安全，根据防火墙交换机安全事件，实时计算得出结论并立即采取相应措施（实时计算）

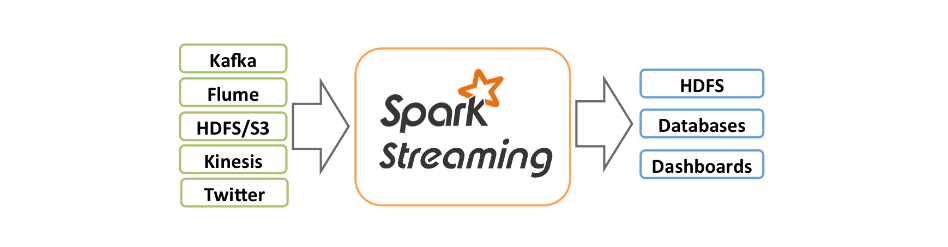
4. 无人驾驶车

猜测离线计算以及实时计算的设计

**什么是Spark Streaming**

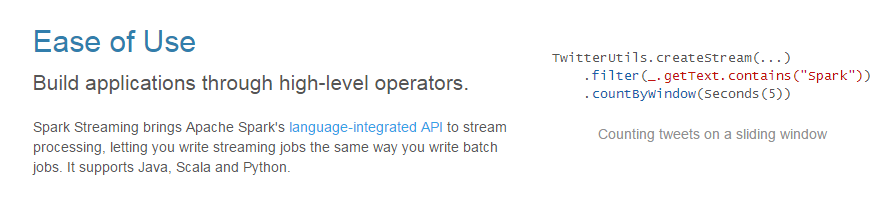


Spark Streaming类似于Apache Storm，用于流式数据的处理。根据其官方文档介绍，Spark Streaming有高吞吐量和容错能力强等特点。Spark Streaming支持的数据输入源很多，例如：Kafka、Flume、Twitter、ZeroMQ和简单的TCP套接字等等。数据输入后可以用Spark的高度抽象原语如：map、reduce、join、window等进行运算。而结果也能保存在很多地方，如HDFS，数据库等。另外Spark Streaming也能和MLlib（机器学习）以及Graphx完美融合。

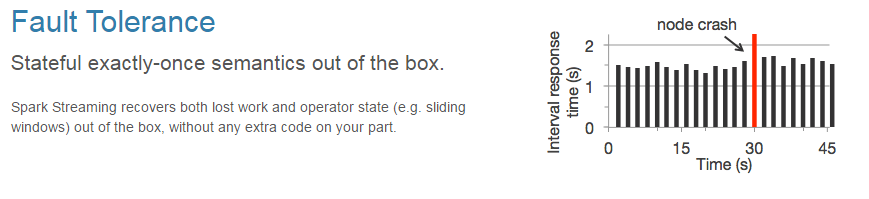


### 为什么要学习Spark Streaming

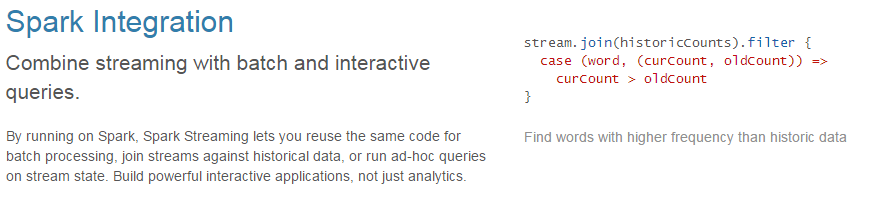
1. 易用



1. 容错



1. 易整合到spark体系



**DStream**

什么是DStream

Discretized Stream是Spark Streaming的基础抽象，代表持续性的数据流和经过各种Spark原语操作后的结果数据流。在内部实现上，DStream是一系列连续的RDD来表示。每个RDD含有一段时间间隔内的数据，如下图：



对数据的操作也是按照RDD为单位来进行的



计算过程由Spark engine来完成



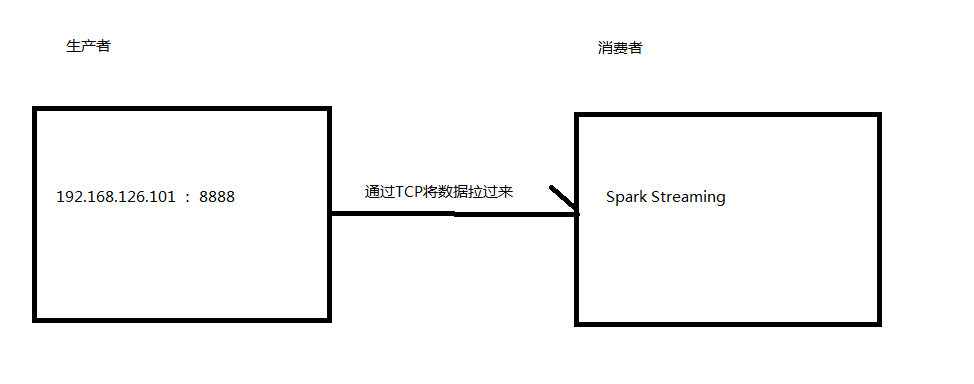
## DStream相关操作

DStream上的原语与RDD的类似，分为Transformations（转换）和Output Operations（输出）两种，此外转换操作中还有一些比较特殊的原语，如：updateStateByKey()、transform()以及各种Window相关的原语。

具体的方法说明参考：<http://spark.apache.org/docs/1.6.0/streaming-programming-guide.html>

Spark Streaming 实践

架构图：



1. 安装并启动生成者

首先在一台Linux（ip：192.168.126.101）上用YUM安装nc工具

yum install -y nc

启动一个服务端并监听8888端口

nc -lk 8888

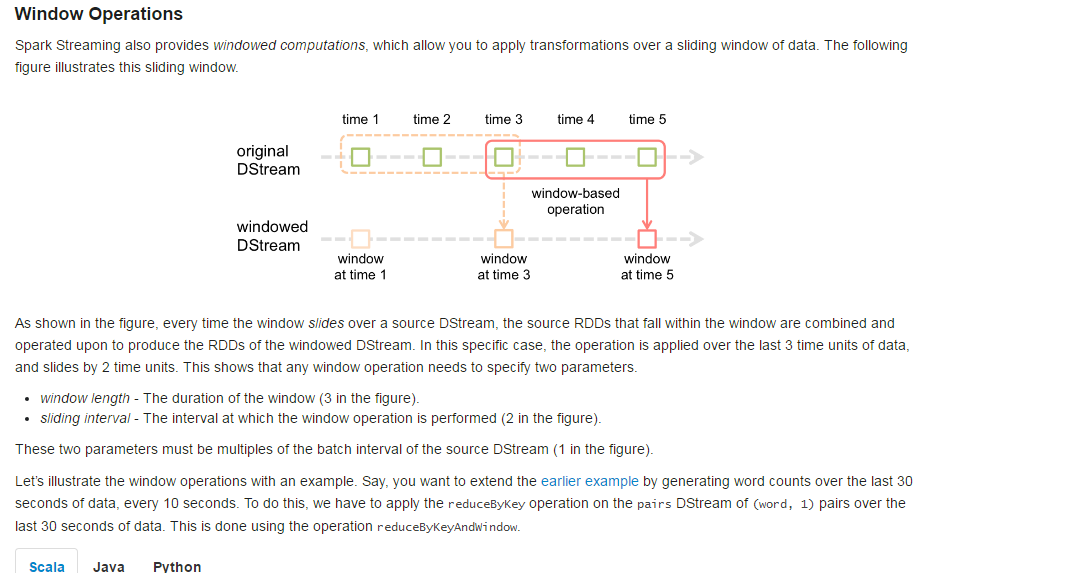
第一个程序：

def main(args: Array[String]): Unit = {  
 val conf = new SparkConf().setAppName("StreamingWordCount").setMaster("local[4]")  
 val sc = new SparkContext(conf)  
 // 创建streamContext对象  
 val streamContext = new StreamingContext(sc,Seconds(5) )  
 // 从IP为192.168.126.101 端口号为8888的应用那里拿数据  
 val dStream = streamContext.socketTextStream("192.168.126.101" , 8888)  
 // 对数据进行处理  
 val resultDStream = dStream.flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_)  
 resultDStream.print()  
 // 启动spark streaming  
 streamContext.start()  
 // 等待被停止  
 streamContext.awaitTermination()  
}

结果每次在Linux段输入的单词次数都被正确的统计出来，但是结果不能累加！如果需要累加需要使用updateStateByKey(func)来更新状态，下面给出一个例子：

def main(args: Array[String]): Unit = {  
 val conf = new SparkConf().setAppName("StreamingWordCount")  
 val sc = new SparkContext(conf)  
 // 创建streamContext对象  
 val streamContext = new StreamingContext(sc,Seconds(5) )  
 // 从IP为192.168.126.101 端口号为8888的应用那里拿数据  
 val dStream = streamContext.socketTextStream("192.168.126.101" , 8888)  
  
 streamContext.checkpoint("hdfs://bigdata01:9000/sparkstream")  
 // 对数据进行处理, 在原有基础上进行更新  
 val resultDStream = dStream.flatMap(\_.split(" ")).map((\_,1)).  
 updateStateByKey(updateFunc , new HashPartitioner(streamContext.sparkContext.defaultParallelism),true)  
 resultDStream.print()  
 // 启动spark streaming  
 streamContext.start()  
 // 等待被停止  
 streamContext.awaitTermination()  
}

窗口函数



def main(args: Array[String]): Unit = {  
 val conf = new SparkConf().setAppName("StreamingWordCountTest").setMaster("local[4]")  
 val sc = new SparkContext(conf)  
 val ssc = new StreamingContext(sc,Seconds(5))  
 ssc.checkpoint("hdfs://bigdata01:9000/sparkstream01")  
 val dStream = ssc.socketTextStream("192.168.126.101", 8888)  
  
 val dStream01 = dStream.flatMap(\_.split(" ")).countByWindow(Seconds(10),Seconds(5))  
 dStream01.print()  
  
 ssc.start()  
 ssc.awaitTermination()  
  
}

spark streaming Dstream、窗口函数

1. 将课堂上的场景全部操作一遍（demo部分先理解，再背着写出来，再总结；源码部分要点进去看）

2. 熟悉官方文档