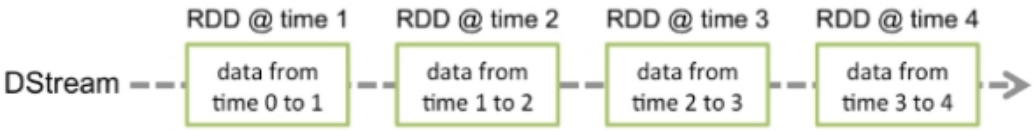
> 人的大脑倾向于做更容易的事情

SparkStreaming之Dstream入门

SparkStreaming概述

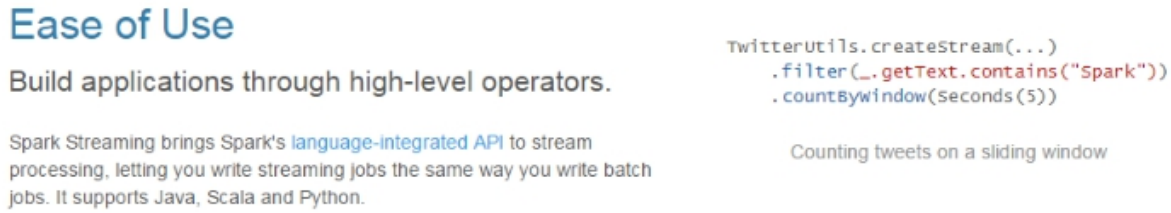
> Spark Streaming用于流式数据的处理。Spark Streaming支持的数据输入源很多，例如：Kafka、Flume、Twitter、ZeroMQ和简单的TCP套接字等等。数据输入后可以用Spark的高度抽象原语如：map、reduce、join、window等进行运算。而结果也能保存在很多地方，如HDFS，数据库等。



和Spark基于RDD的概念很相似，Spark Streaming使用离散化流(discretized stream)作为抽象表示，叫作DStream。**DStream 是随时间推移而收到的数据的序列**。在内部，每个时间区间收到的数据都作为 RDD 存在，而DStream是由这些RDD所组成的序列(因此得名“离散化”)。

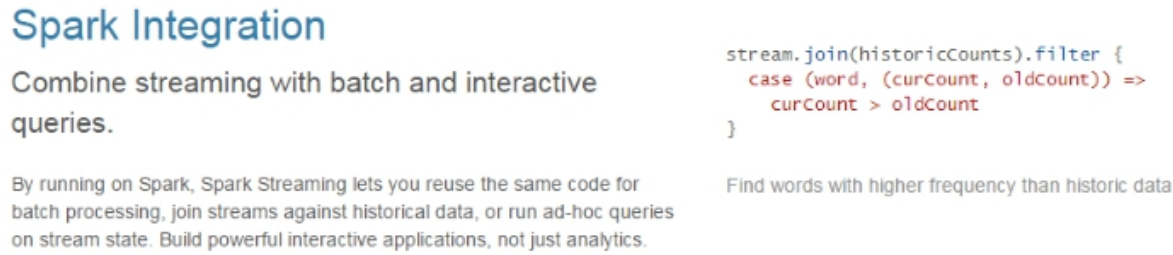
SparkStreaming特点

1.易用

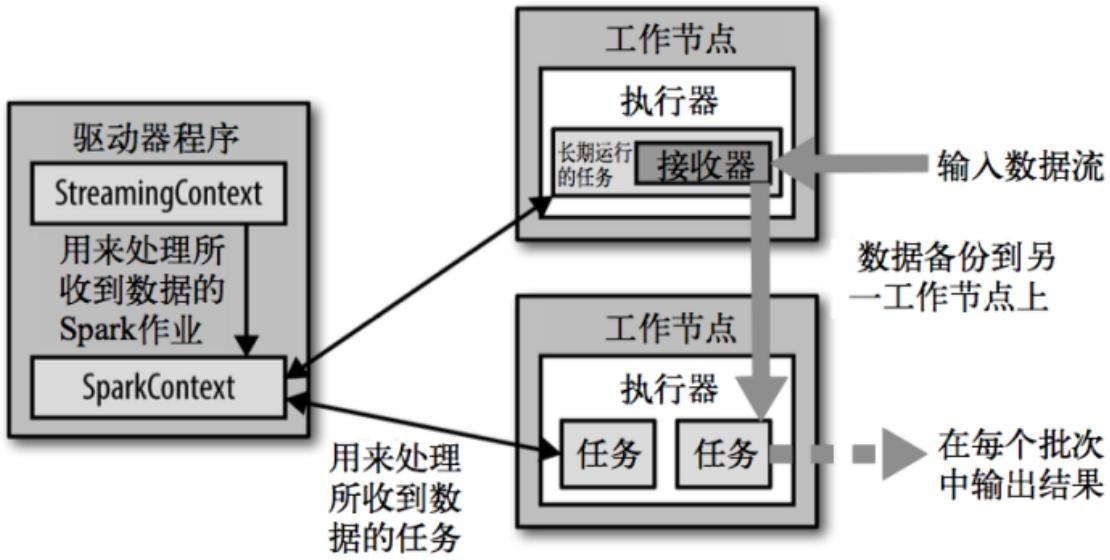


2.容错

3.易整合到Spark体系



SparkStreaming架构



背压机制

> Spark 1.5以前版本，用户如果要限制Receiver的数据接收速率，可以通过设置静态配制参数“spark.streaming.receiver.maxRate”的值来实现，此举虽然可以通过限制接收速率，来适配当前的处理能力，防止内存溢出，但也会引入其它问题。比如：producer数据生产高于maxRate，当前集群处理能力也高于maxRate，这就会造成**资源利用率下降**等问题。
>
> 为了更好的协调数据接收速率与资源处理能力，1.5版本开始Spark Streaming可以**动态控制数据接收速率**来适配集群数据处理能力。
>
> 背压机制（即Spark Streaming Backpressure）：根据JobScheduler反馈作业的执行信息来动态调整Receiver数据接收率。
>
> 通过属性“spark.streaming.backpressure.enabled”来控制是否启用backpressure机制，默认值false，即不启用。

Dstream基本操作

DStream 是随时间推移而收到的数据的序列

需求

> 使用netcat工具向9999端口不断的发送数据，通过SparkStreaming读取端口数据并统计不同单词出现的次数

maven依赖

```
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-streaming_2.11</artifactId>
    <version>2.1.1</version>
</dependency>
```

实现代码

```
import org.apache.spark.streaming.dstream.{DStream, ReceiverInputDStream}
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.SparkConf

object StreamWordCount {

  def main(args: Array[String]): Unit = {

    //1.初始化Spark配置信息
    val sparkConf = new
SparkConf().setMaster("local[*]").setAppName("StreamWordCount")

    //2.初始化SparkStreamingContext
    val ssc = new StreamingContext(sparkConf, Seconds(3))

    //3.通过监控端口创建DStream，读进来的数据为一行行
    val lineStreams = ssc.socketTextStream("hadoop102", 9999)

    //将每一行数据做切分，形成一个个单词
    val wordStreams = lineStreams.flatMap(_.split(" "))

    //将单词映射成元组（word,1）
    val wordAndOneStreams = wordStreams.map((_, 1))

    //将相同的单词次数做统计
    val wordAndCountStreams = wordAndOneStreams.reduceByKey(_+_)

    //打印
    wordAndCountStreams.print()

    //启动SparkStreamingContext
    ssc.start()
    ssc.awaitTermination()
  }
}
```

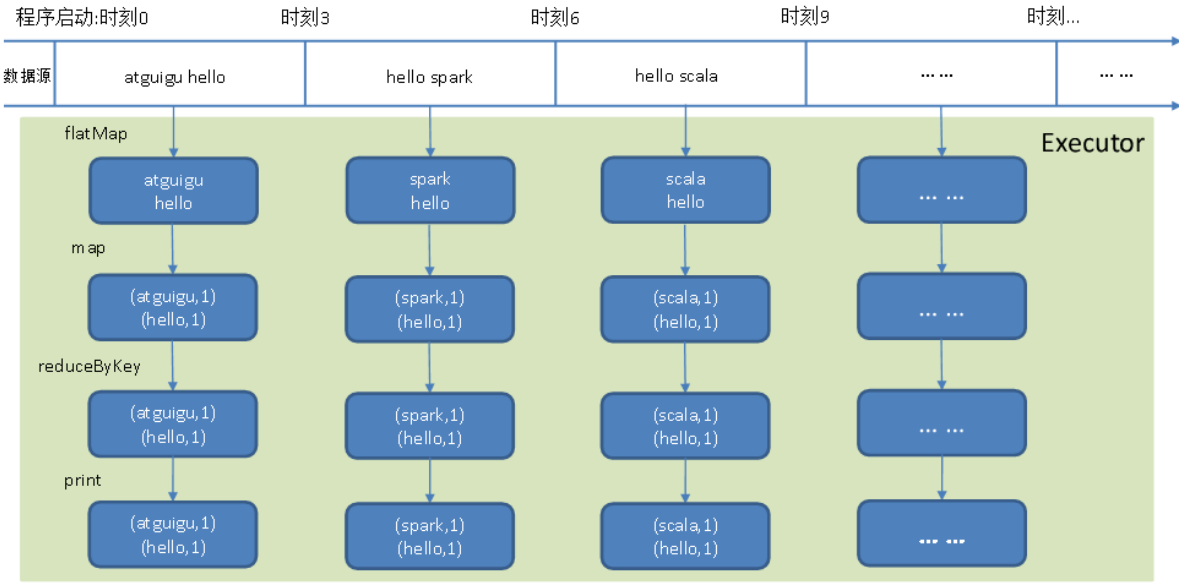
首先启动程序，然后通过NetCat想指定端口发送数据

```
[zhutiansama@hadoop102 spark]$ nc -lk 9999
```

注意：如果程序运行时，log日志太多，可以将spark conf目录下的log4j文件里面的日志级别改成WARN

WordCount解析

> 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：


程序启动:时刻0
时刻3
时刻6
时刻9
时刻...
数据源
atguigu hello
hello spark
hello scala
... ...
... ...
Executor
flatMap
atguigu
hello
spark
hello
scala
hello
... ...
map
(atguigu,1)
(hello,1)
(spark,1)
(hello,1)
(scala,1)
(hello,1)
... ...
reduceByKey
(atguigu,1)
(hello,1)
(spark,1)
(hello,1)
(scala,1)
(hello,1)
... ...
print
(atguigu,1)
(hello,1)
(spark,1)
(hello,1)
(scala,1)
(hello,1)
... ...