7.3 DStream操作概述

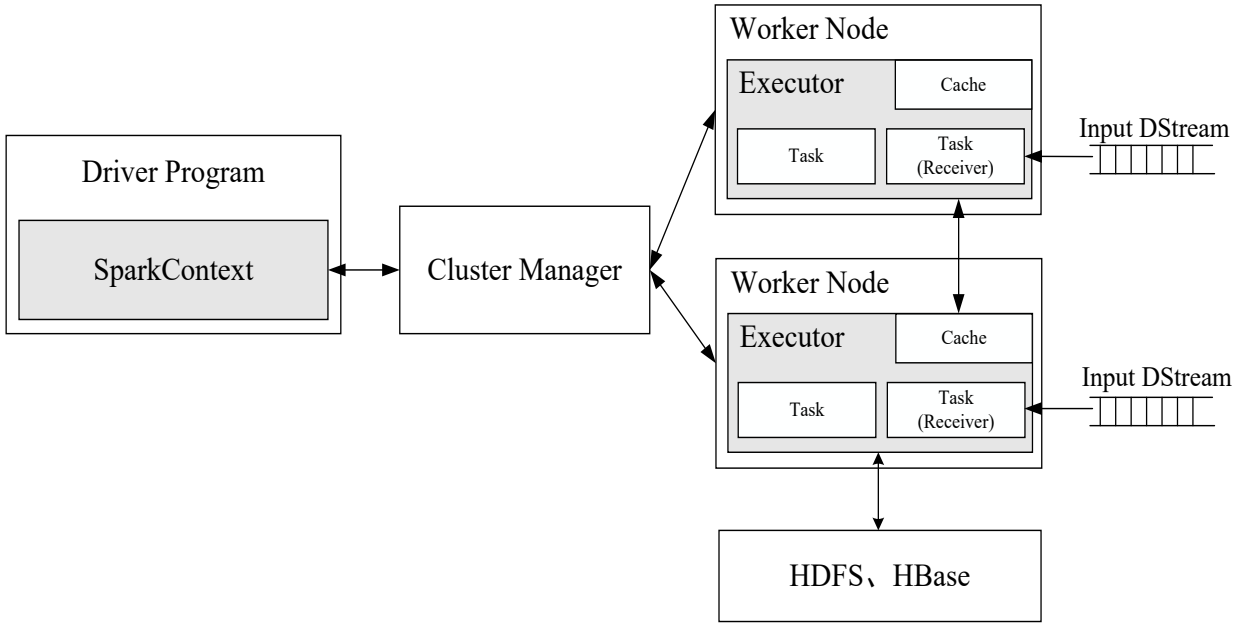
7.3.1 Spark Streaming工作机制

7.3.2 Spark Streaming程序的基本步骤

7.3.3 创建StreamingContext对象

7.3.1 Spark Streaming工作机制





- 在Spark Streaming中，会有一个组件Receiver，作为一个长期运行的task跑在一个Executor上
- 每个Receiver都会负责一个input DStream（比如从文件中读取数据的文件流，比如套接字流，或者从Kafka中读取的一个输入流等等）
- Spark Streaming通过input DStream与外部数据源进行连接，读取相关数据

7.3.2 Spark Streaming程序的基本步骤



编写Spark Streaming程序的基本步骤是：

- 1.通过创建输入DStream来定义输入源
- 2.通过对DStream应用转换操作和输出操作来定义流计算
- 3.用streamingContext.start()来开始接收数据和处理流程
- 4.通过streamingContext.awaitTermination()方法来等待处理结束（手动结束或因为错误而结束）
- 5.可以通过streamingContext.stop()来手动结束流计算进程

7.3.3 创建StreamingContext对象



- 如果要运行一个Spark Streaming程序，就需要首先生成一个StreamingContext对象，它是Spark Streaming程序的主入口
- 可以从一个SparkConf对象创建一个StreamingContext对象。
- 在pyspark中的创建方法：进入pyspark以后，就已经获得了一个默认的SparkConext对象，也就是sc。因此，可以采用如下方式来创建StreamingContext对象：

```
>>> from pyspark.streaming import StreamingContext
>>> ssc = StreamingContext(sc, 1)
```

7.3.3 创建StreamingContext对象



如果是编写一个独立的Spark Streaming程序，而不是在pyspark中运行，则需要通过如下方式创建StreamingContext对象：

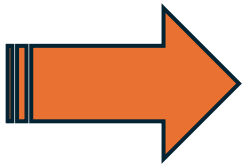
```
#coding:utf8
from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext
if __name__=='__main__':
    conf=SparkConf().setMaster("local[2]").setAppName("Ch07 TestDStream")
    sc=SparkContext(conf=conf)
    ssc=StreamingContext(sc,1)
    print(sc.version)
    sc.stop()
```

FutureWarning: DStream is deprecated as of Spark 3.4.0. Migrate to Structured Streaming.

为什么要迁移到Structured Streaming？



特性	Dstream(ch07)	Structured Streaming(ch08)
API	基于RDD（底层API）	基于DataFrame/Dataset（高层API）
事件时间处理	实现复杂	原生支持（如withWatermark）
性能	微批处理模式限制	优化后的执行引擎
维护性	已弃用（未来可能移除支持）	Spark官方主推的流处理方案



Stop here!!
Go to ch08