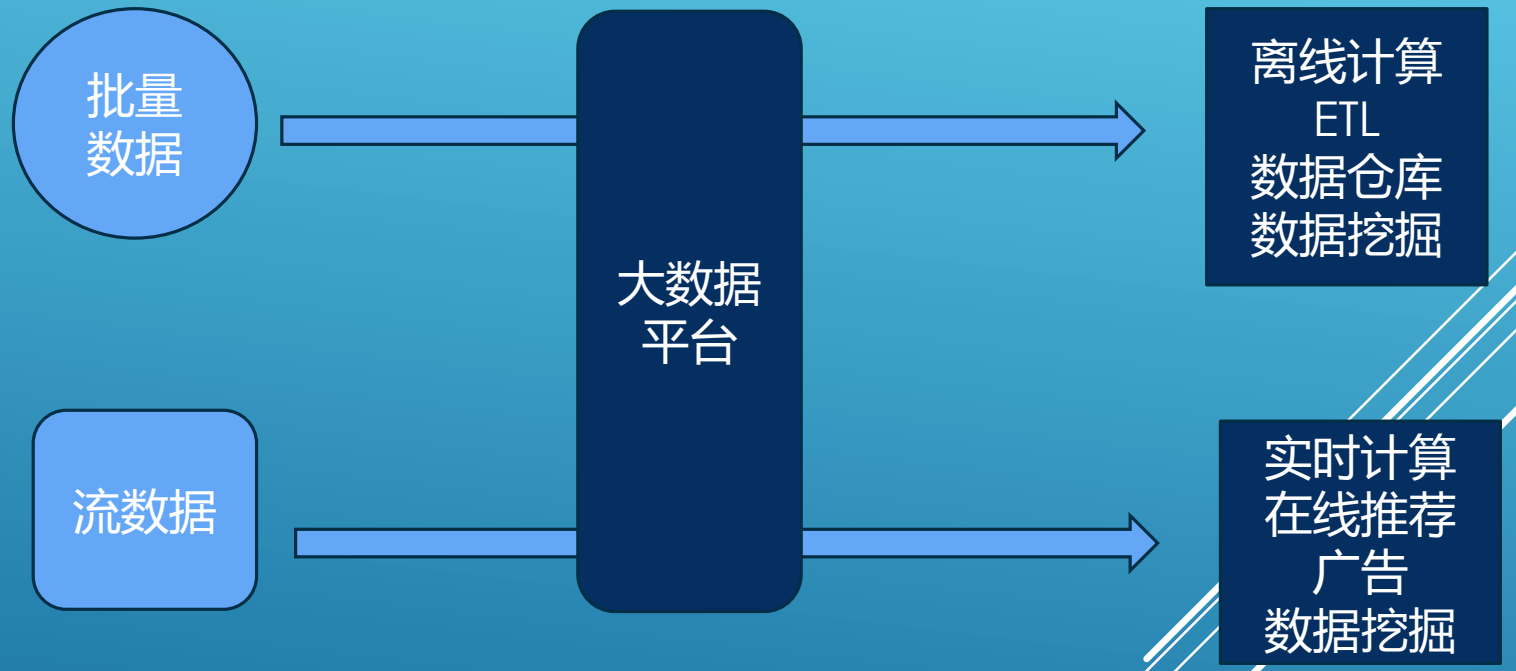
目　　标

1. 理解批处理任务与流式处理任务
2. 了解Spark流式处理的基本概念和流程
3. 掌握Spark流式编程模式

一、主流的大数据处理任务



一、批量数据与流式数据的各自特点

批量数据

大规模静态数据，
特点是时间跨度大、持久化、数据规模大

流数据

一组数据序列
特点是实时性高、有顺序（该顺序有独立性）、
无法探测边界、易丢失（需要保存）

共性：都具有**时间边界**

一、批量数据与流式数据示例

批量数据

起点网近一年全站小说TXT
游戏后台服务器上个月的日志数据

流数据

飞机传感器的监测数据

证券公司股票波动数据

你眼中的批量数据与流式数据

一、以下各自是什么任务？

1、部门老大要看一份报告，查看公司的产品去年每个月在全国各地的销售额

2、高德地图根据用户的轨迹推荐 用户可能感兴趣的POI

3、举例说明身边的典型数据处理任务

一、流式数据处理框架

- Storm
- SparkStreaming
- Flink
- Samza

SparkStream ——Spark中的流数据处理框架

准实时– 秒级

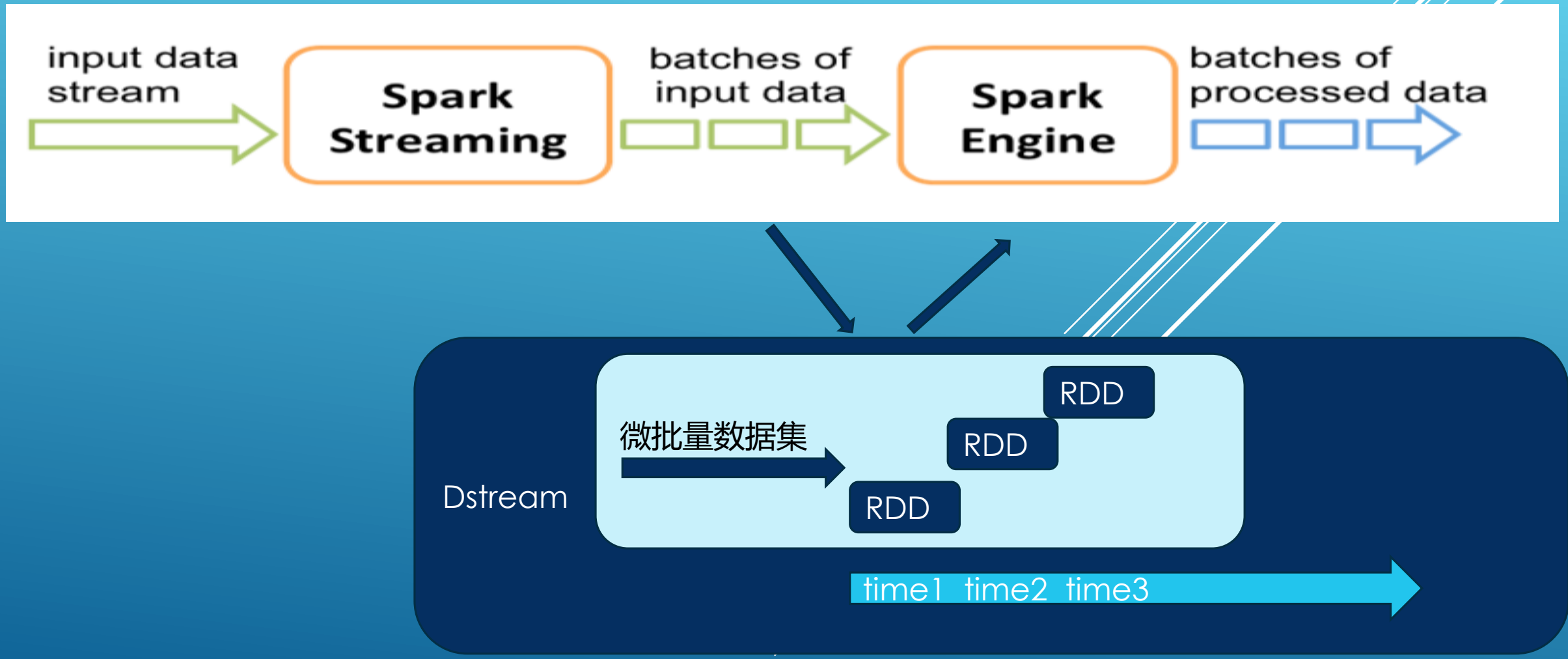
微批处理

扩展性强、吞吐量高

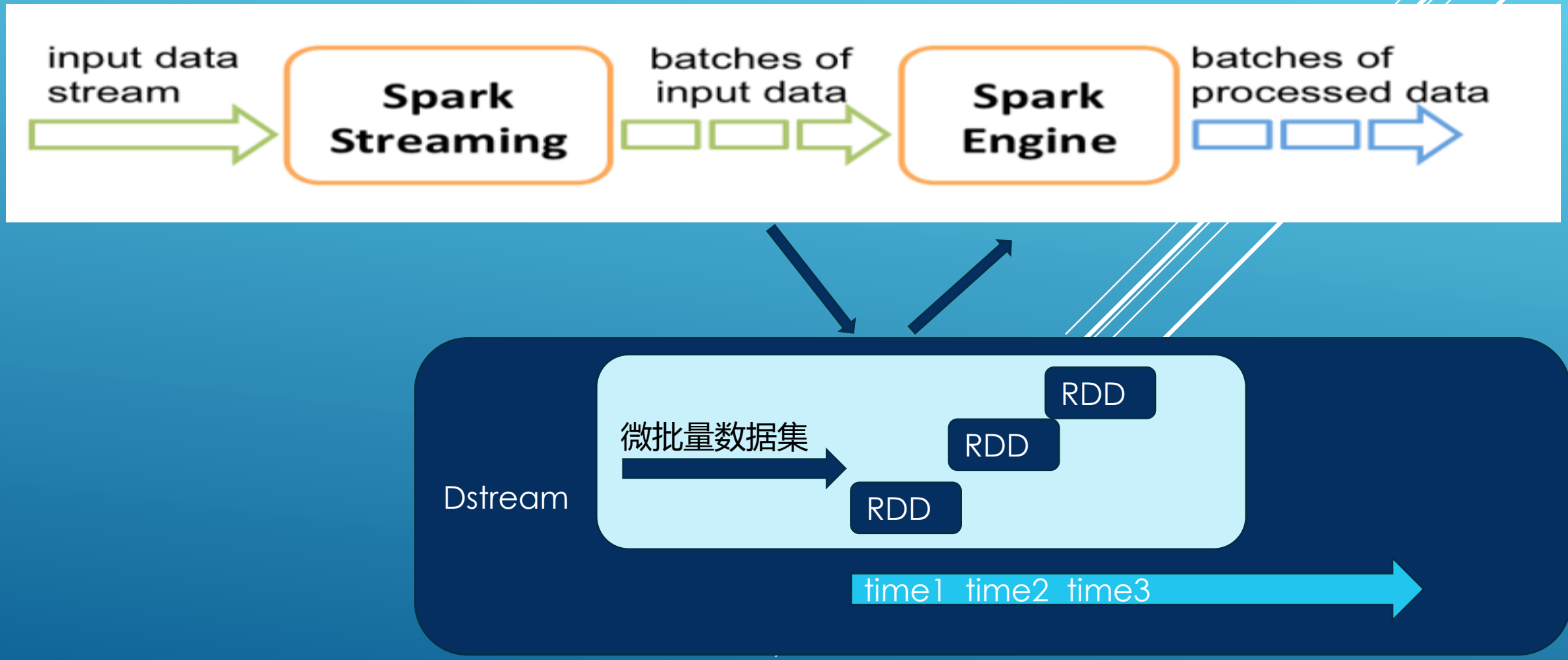
容错性好，恢复快

省事省时，统一的编程逻辑

一、 SparkStream 处理流程基础知识

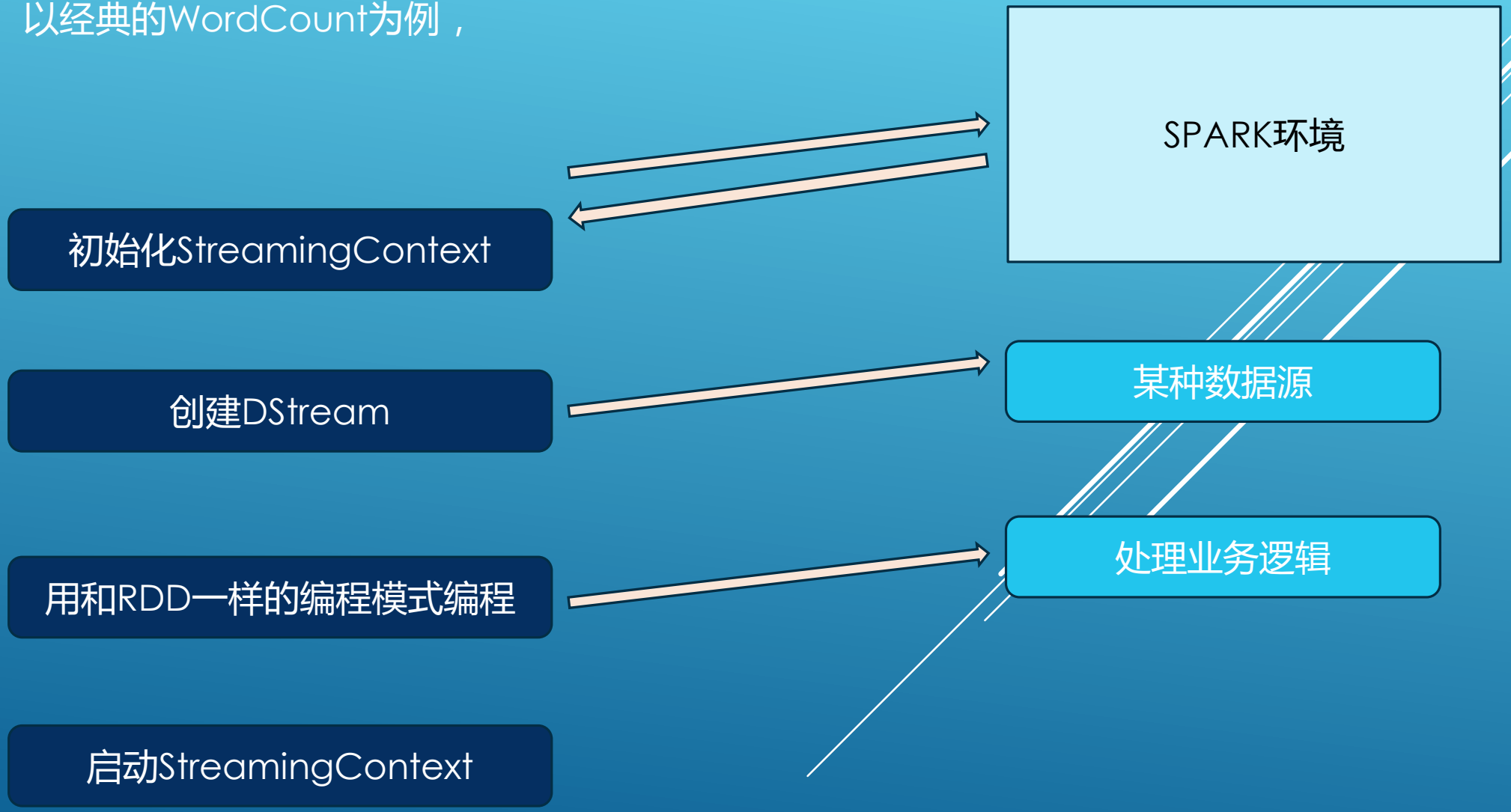


一、 SparkStream 处理流程-底层原理（中级略）



一、 SparkStream 编程示例

以经典的WordCount为例，



一、 SparkStream 编程示例

以经典的WordCount为例，

初始一个StreamContext

```
val conf = new SparkConf().setMaster("local[*]").setAppName("WordCountByStream")
val ssc = new StreamingContext(conf, Seconds(5))
val lines = ssc.socketTextStream( hostname = "localhost",  port = 9527)
```

构建一个基于Socket的Stream

一、 SparkStream 编程示例

SparkStream 支持多种数据源，这里我们使用**套接字**数据源

```
val listener: ServerSocket = new ServerSocket( port = 9527)
while (true) {
  val socket: Socket = listener.accept()
  while (socket.isConnected) {
    val out = new DataOutputStream(socket.getOutputStream())
    while (true) {
      println("begin send " + socket.getInetAddress.getHostAddress)
      //     val in = new ObjectInputStream(new DataInputStream(socket.getInputStream()))
      out.writeUTF( str = "长江 长江 这是 黄河\n")
      out.writeUTF( str = "老鹰 呼叫 猎狗\n")
      out.flush()
      Thread.sleep( millis = 500)
    }

  }
```

为演示目的：模拟数据生产服务，
注意：生产环境不会这么写，应该
与多线程结合

一、 SparkStream 编程示例

```
val words = lines.flatMap(line => line.split( regex = "\\s+"))
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)
wordCounts.print()
  wordCounts.saveAsTextFiles("data/wc")
ssc.start()
ssc.awaitTermination()
```

启动SparkSteam应用

与RDD编程模式的区别?

一、 SparkStream 编程示例

课堂作业：

猜测一下这个创建的是什么Dstream， 有什么作用，并验证

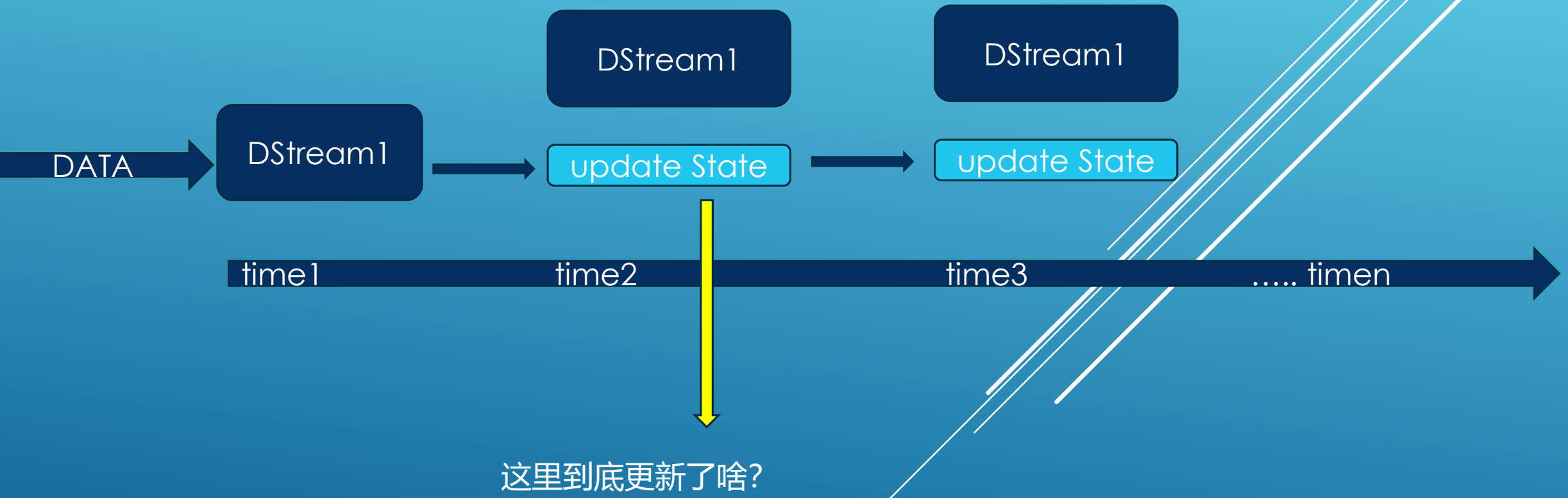
```
val lines = ssc.textFileStream( directory = "data/docs")
```

一、 SparkStream 编程示例

仔细观察前面的 WordCont 示例的结果输出，与以往有什么不同?

一、 SparkStream 编程示例

SparkStreaming支持有状态的DStream



一、 SparkStream 编程示例

SparkStreaming支持有状态的DStream

- 什么样的状态
- 如何更新

一、 SparkStream 编程示例

针对WordCount，需要更新的是map阶段每个单词出现的次数

一、 SparkStream 编程示例

SparkStreaming支持有状态的DStream

针对WordCount，需要更新的是map阶段每个单词出现的次数

```
def updateFunction(newValues: Seq[Int], runningCount: Option[Int]): Option[Int] = {

  // 定义更新的逻辑
}
```

```
val runningCounts = pairs.updateStateByKey(updateFunction)
```

执行状态更新

```

val wordCounts = runningCounts.reduceByKey(_ + _)
```