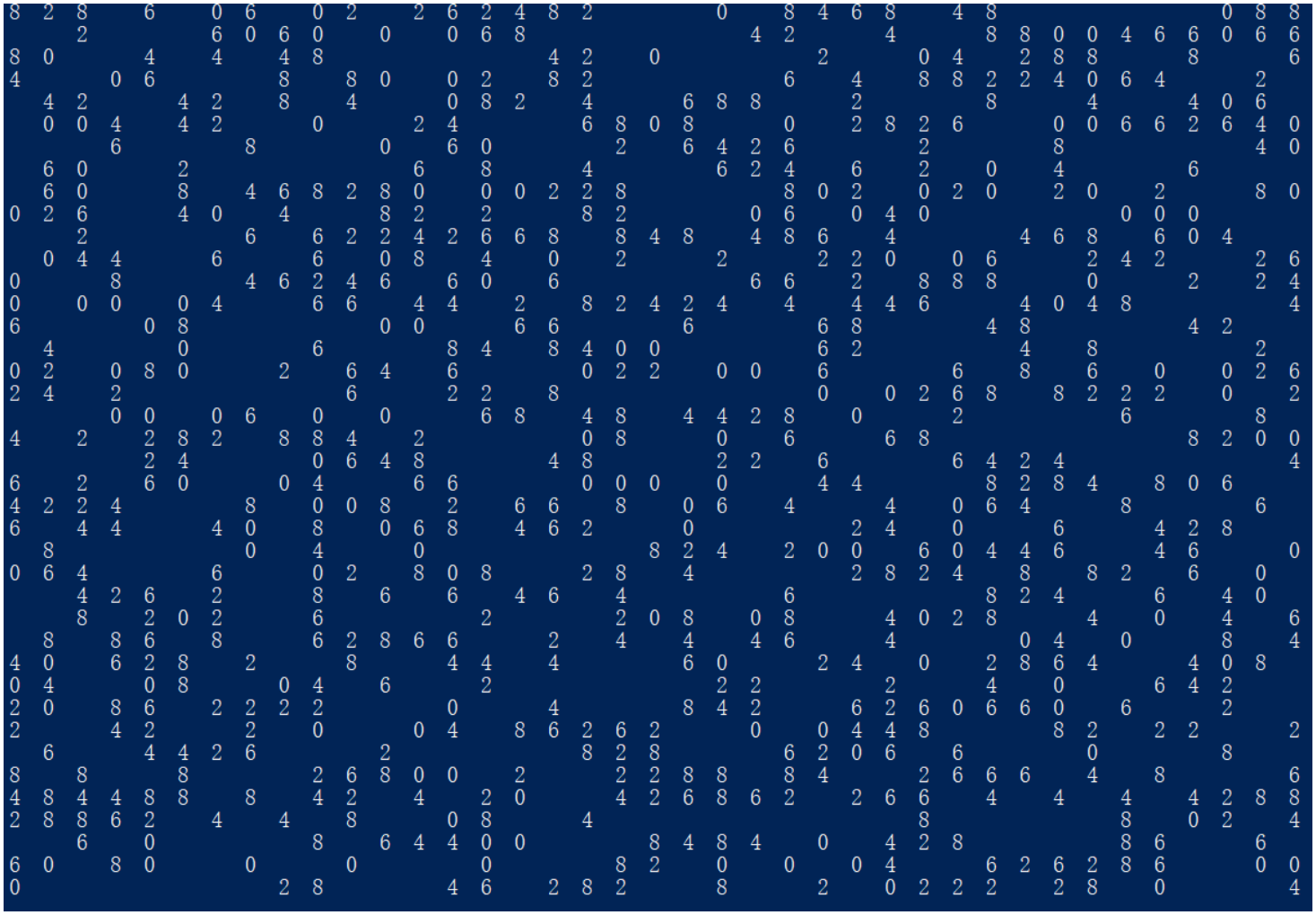
Scala 基础编程 和 Spark入门

课程目标：

1. 熟悉scala的基本语法、数据类型、程序控制结构
2. 熟悉scala的常用数据容器
3. 了解scala面向对象
4. 了解Spark程序的基本运行原理、程序结构、编程模式
5. 掌握RDD概念及相应的基本操作
6. 能够在老师的指导下，完成编程案例

案例1：模拟你们熟悉的数据雨



要求：

1. 以控制台的方式进行模拟
2. 可指定雨帘的高度、宽度、动态降落速度

案例2：BZT通讯录程序

主菜单

********* BZT通讯录 V1 *********

1：添加联系人
2：查看通讯录列表
3：查看联系人详情
4：删除联系人
5：修改联系人详情
6：退出

添加功能

```
*****添加联系人*****
请输入姓名：
  张三
请输入电话号码：
  1585868238
```

```
*****添加成功*****

a: 继续添加
b: 返回主菜单
```

删除功能

```
*****删除联系人*****
    请输入姓名：
    张三
```

```
*****删除联系人成功*****
  任意键返回主菜单
```

```
*****联系人不存在******

  联系人 张三 不存在
  a: 重新输入
  b: 返回主菜单
```

修改功能

*****修改联系人*****
请输入姓名：
张三

*****联系人不存在******

联系人 张三 不存在
a: 重新输入
b: 返回主菜单

*****修改联系人*****

张三　18283481234

请输入新的姓名:
请输入新的电话：

查找功能

*****查看联系人*****
请输入姓名：

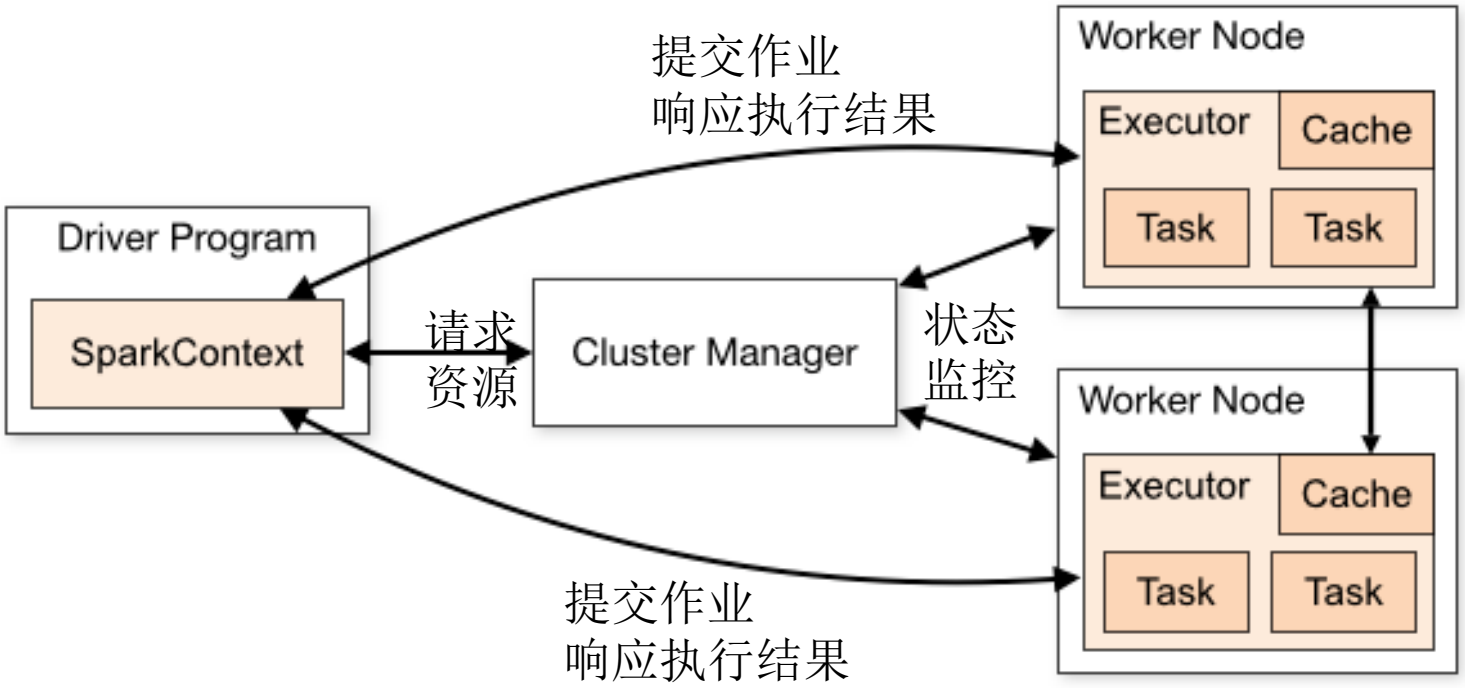
*****联系人详情*****
姓名：张三
电话：283481243

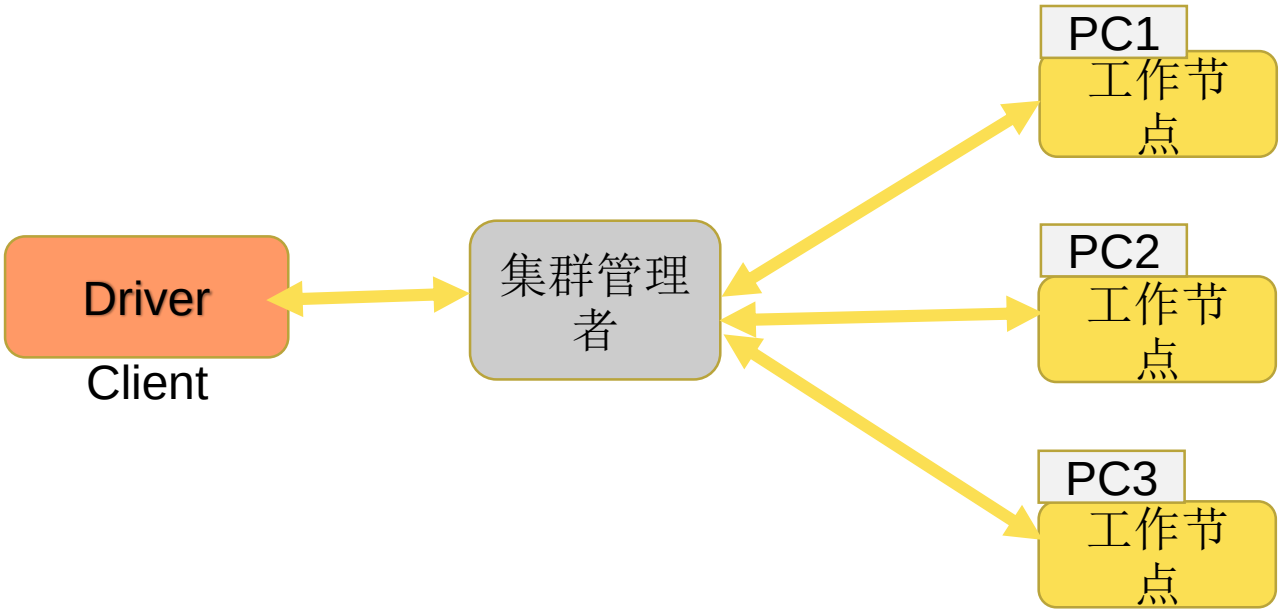
*****联系人不存在******

联系人 张三 不存在
a: 重新输入
b: 返回主菜单

*****联系人列表*****
张三　1283841234
李江　2838421213
刘二狗　8234248 2

了解Spark 架构及运行原理





SparkCore：将分布式数据抽象为弹性分布式数据集（RDD），实现了应用任务调度、RPC、序列化和压缩，并为运行在其上的上层组件提供API。

SparkSQL：Spark Sql 是Spark来操作结构化数据的程序包，可以让我使用SQL语句的方式来查询数据，Spark支持 多种数据源，包含Hive表，parquest以及JSON等内容。

SparkStreaming： 是Spark提供的实时数据进行流式计算的组件。

MLlib：提供常用机器学习算法的实现库。

GraphX：提供一个分布式图计算框架，能高效进行图计算。

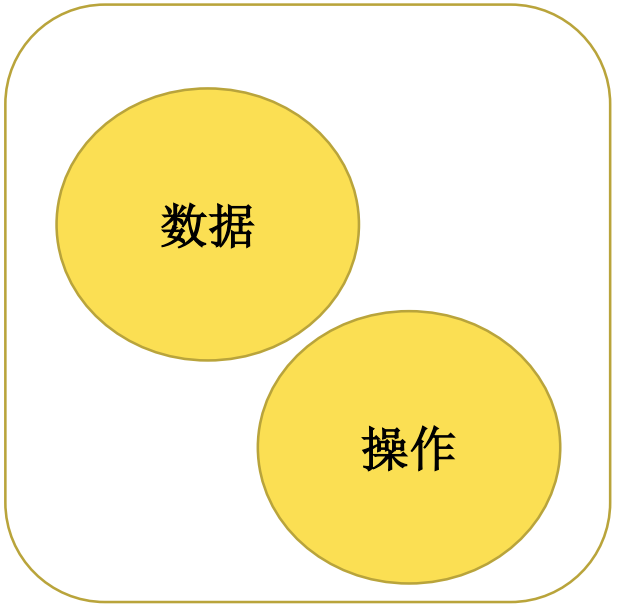
~~**BlinkDB**：用于在海量数据上进行交互式SQL的近似查询引擎。~~

~~**Tachyon**：以内存为中心高容错的的分布式文件系统。~~

假如把计算看作数据 + 操作

Spark 怎么定义数据

Spark支持哪些操作



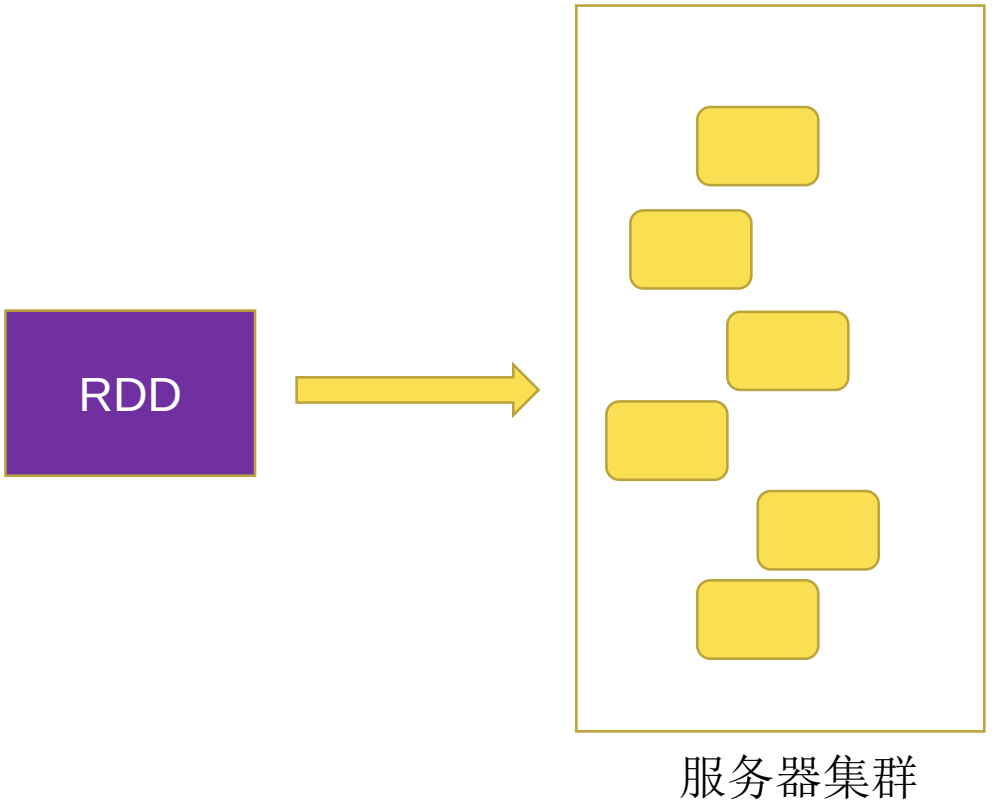
Spark的数据集抽象：
RDD（Resilient Distributed Datasets）———弹性分布式数据集

任何运行在Spark集群上的并行化计算都是基于RDD

Spark的数据集抽象：RDD（Resilient Distributed Datasets）
弹性分布式数据集

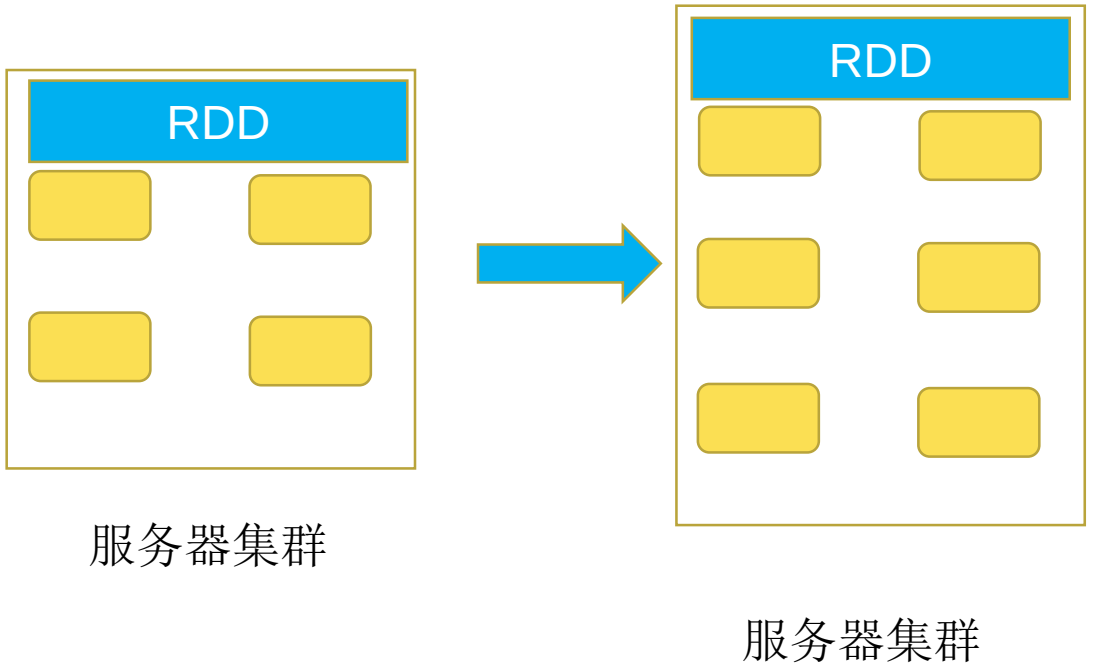
RDD基本特性：

1. **分布式**
2. 弹性
3. 不可变



Spark的数据集抽象：RDD（Resilient Distributed Datasets）
弹性分布式数据集

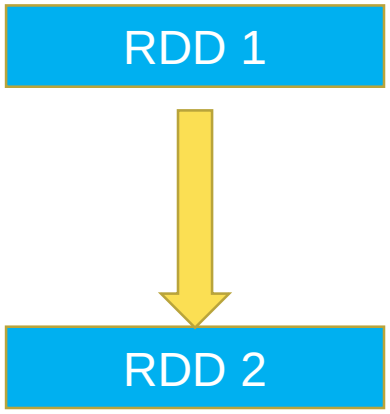
RDD基本特性：
1. 分布式
2. **弹性**
3. 不可变



Spark的数据集抽象：RDD（Resilient Distributed Datasets）
弹性分布式数据集

RDD基本特性：

1. 分布式
2. 弹性
3. **不可变**



如何构建RDD

RDD可以直接来自内存数据 (常见的集合容器)

也可以来自外部数据(数据库、HDFS、本地文件系统、数据流……)

示例1:构建一个基于本地文件系统的RDD

```
val spark = SparkSession.builder()
      .appName("countDistinctDemo")
      .master(“master")
      .getOrCreate();
```

```
    val data = spark.sparkContext.textFile("data\\test.data");
```

示例2:构建一个基于本地内存集合容器的RDD

```
val rdd1 = spark.sparkContext.parallelize(Seq("张飞", "关羽", "刘备"))
rdd1.foreach(println)
```

构建一个显示指定5个分区的RDD

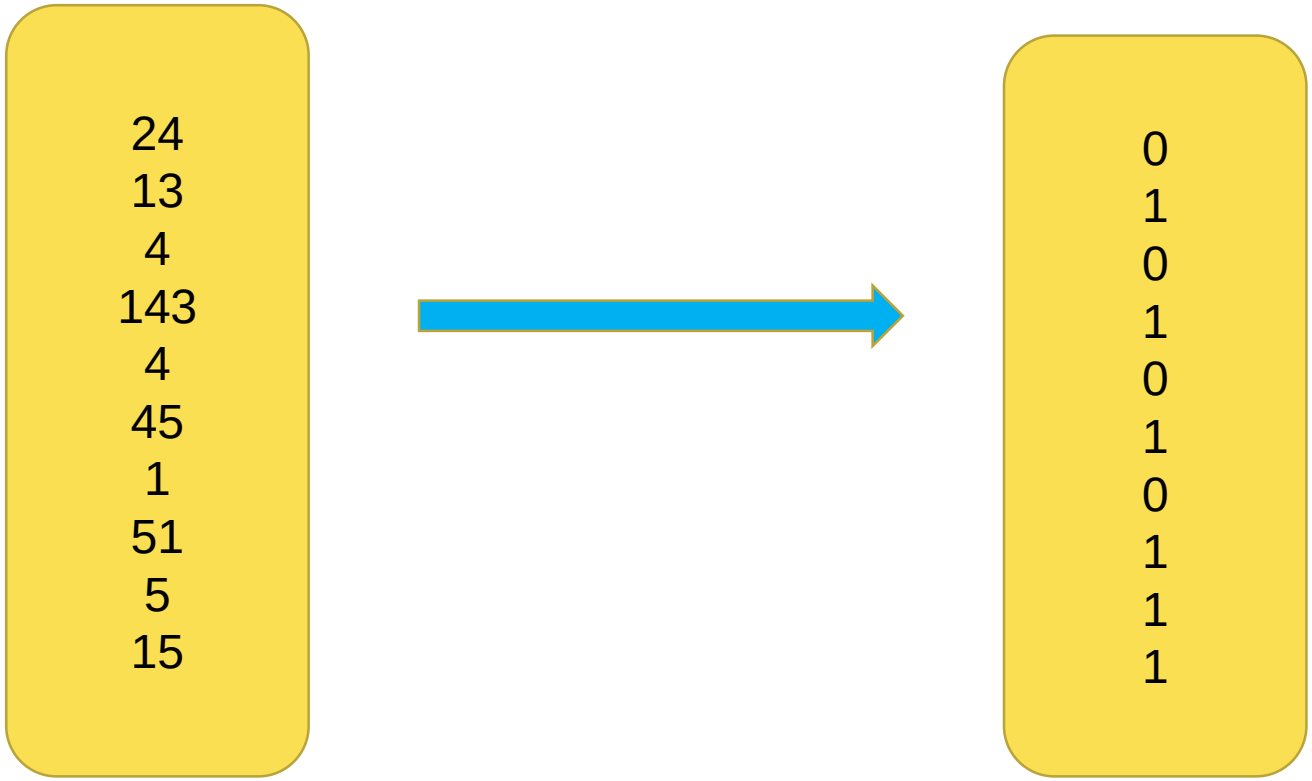
```
val rdd1 = spark.sparkContext.parallelize(1 to 100, 5)
rdd1.foreach(println)
```

什么是RDD分区

RDD的基本操作

转换
行为
创建及状态控制

RDD的基本操作---转换
特点：延迟执行



RDD的基本操作---转换

Map操作

```
val rdd2 = spark.sparkContext.parallelize(1 to 100, 5)
  rdd2.foreach(println)
  val rdd3 = rdd2.map(x => if(x%2==0) 0 else 1)
```

RDD的基本操作---转换

flatMap操作与map操作

分别执行以下两组代码，比较输出内容

```
val rdd4 = spark.sparkContext.textFile("data\\test.data")
  val element = rdd4.map(line => line.split("\\s+")).take(1)
  element.foreach(println)
```

```
val rdd5 = spark.sparkContext.textFile("data\\test.data")
  val element5 = rdd4.flatMap(line => line.split("\\s+")).take(2)
  element5.foreach(println)
```

RDD的基本操作---转换

RDD 并集、交集、差集

```
val rdd5 = spark.sparkContext.parallelize(Seq(1,3,5,7,9))
val rdd6 = spark.sparkContext.parallelize(Seq(1,3,2,4,6,8))
//并集  （注意是否去重了）
rdd5.union(rdd6).foreach(println)
//交集
rdd5.intersection(rdd6).foreach(println)
//差集
rdd5.subtract(rdd6).foreach(println)
```

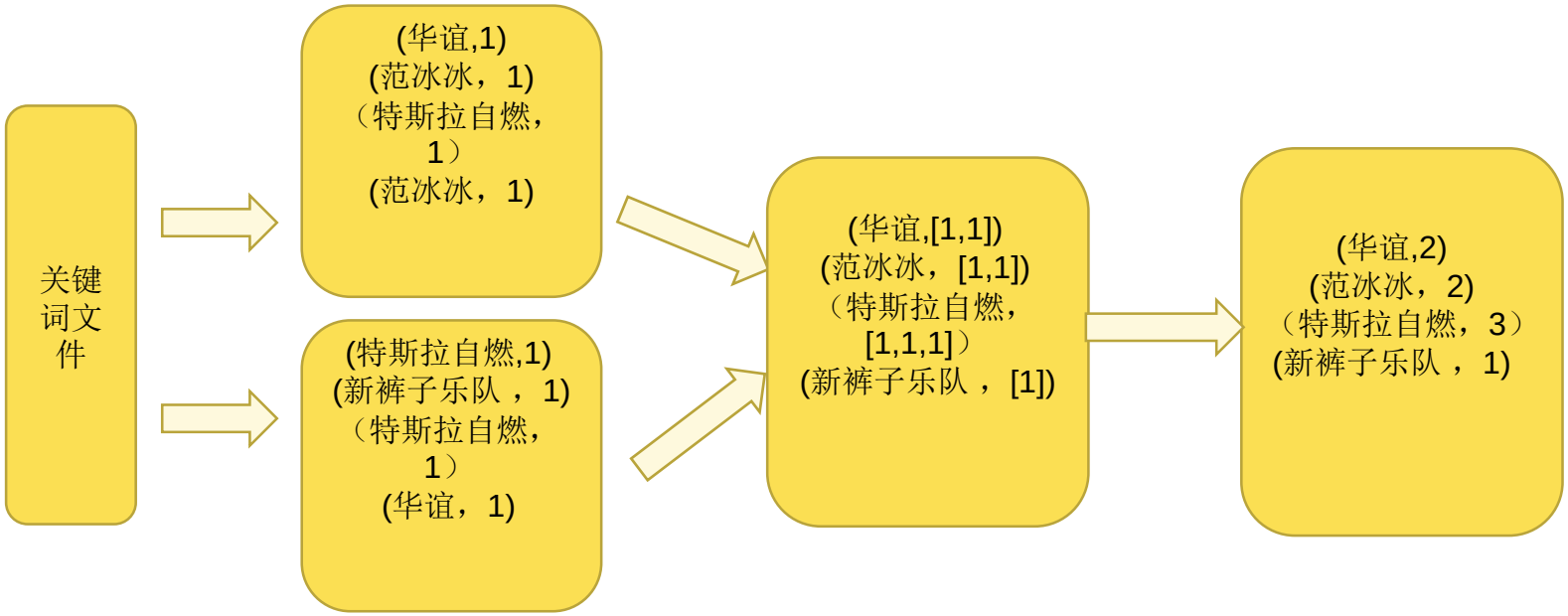
RDD的基本操作---转换

RDD reduceByKey

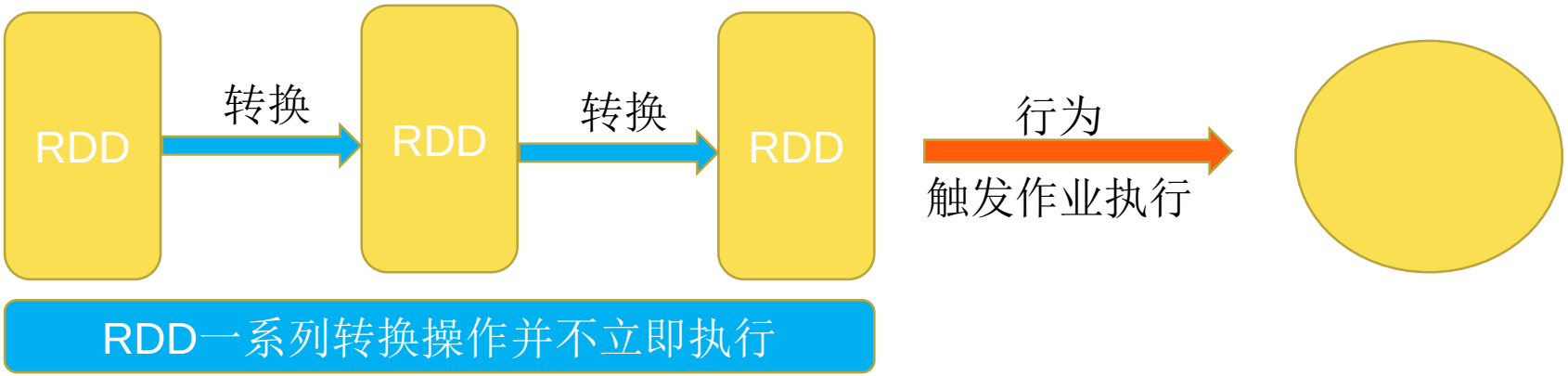
```
val result1 = data.flatMap(line => line.split("\\s+")).map(word => (word, 1))
 .reduceByKey( (x:Int, y:Int) => x + y )

// groupByKey, 比较二者的输出差异
val result2 = data.flatMap(line => line.split("\\s+")).map(word => (word, 1))
  .groupByKey()
result2.foreach(println)
```

回顾上节课的问题，热搜关键词top10排名，解决这个问题需要哪些操作？



RDD的基本操作---行为



为什么？

RDD的基本操作---行为

foreach: 迭代RDD每一个元素，执行指定操作
take：取出前n个元素， 不排序
collect: 从worker 拉取 数据并转换成数组
reduce：对RDD中的每两个元素执行二元操作

RDD的基本操作---行为

collect():

```
val data = spark.sparkContext.textFile("data\\test.data");
val localData = data.collect()
localData.foreach(println)
```

对所有元素求和：

```
val rdd8 = spark.sparkContext.parallelize(Seq(1,3,5,7,9))
    val result3 = rdd8.reduce((x :Int, y :Int) => x + y)
    println(result3)
```

RDD的基本操作---行为

对RDD中最大的数

```
def getBigger(x :Int, y:Int):Int={
    if(x >=y)return x
    else return y
  }
```

```
val result4 = rdd8.reduce((x,y) => getBigger(x, y))
   println(result4)
```

RDD的基本操作—转换

join, leftJoin , rightJoin

ID	姓名
100	张飞
101	关羽
102	刘备
103	赵云

ID	攻击力
100	30000
101	35000
105	4000
106	15000