# RDD

Spark的RDD体现了装饰者设计模式。和java的IO流类似。

## 1.1什么是RDD

RDD（Resilient Distributed Dataset）叫做弹性分布式数据集，是Spark中最基本的数据抽象（计算抽象、逻辑抽象）。在代码中表现为一个抽象类，代表一个不可变、可分区、所包含元素可并行计算的集合。

## 1.2 RDD的属性

1）一组分区（partition），即数据集的基本组成单位

2）一个计算每个分区的函数（当前RDD封装的计算操作）

3）RDD之间的依赖关系

4）一个partitioner，即RDD分片函数

5）一个列表，存储存取每个partition的优先位置（preferred location）。因为移动数据不如移动计算，RDD将计算移动到数据上面，然后执行。

## 1.3 RDD特点

RDD表示只读的分区数据集，如果要对RDD进行改动，只能通过RDD的转换操作，由一个RDD得到一个新的RDD，新的RDD包含了从其他RDD衍生所必需的信息。RDDs之间存在依赖关系，RDD的执行是按照血缘关系延时计算的。如果血缘关系过长，可以通过持久化RDD来切断血缘关系。

### 1.3.1 分区

RDD称为弹性分布式数据集，但是它并不存储数据，可理解为一个数据集的代理。通过RDD，我们可以很方便的对这个数据集进行操作。

虽然RDD没有存储数据，但是它记录了应该读取哪些数据，然后对这些数据进行操作。

现在我们对RDD进行分区，把需要读取的数据进行划分，每个分区记录一部分需要读取的数据。那么，就可以对**每个分区的数据开启一个计算任务**，实现**并行计算**。

### 1.3.2 只读

RDD是只读的，要想改变RDD中的数据，只能在现有RDD基础上创建新的RDD。

由一个RDD转换到另一个RDD，可以通过丰富的操作算子实现，不像MapReduce那样只能写map和reduce。

Spark中所有的方法都称为算子，算子分为两类：转换算子和行动算子

转换算子：将RDD进行转换，构建RDD的血缘关系。

行动算子：用来出发RDD的计算，得到RDD的相关计算结果（向应用程序返回结果：count、collect；向存储系统保存数据：saveAsTextFile）

### 1.3.3 依赖

RDDs之间的血缘关系。包括窄依赖和宽依赖。

窄依赖：RDDs之间的分区是一一对应的

宽依赖：下游RDD的每个分区与上游RDD的所有分区都有关系

### 1.3.4 缓存

应用程序中多次使用同一个RDD时，可以将该RDD缓存起来。该RDD只在第一次计算的时候会根据血缘关系得到分区的数据，在后续其他地方用到该RDD时，会直接使用它的缓存数据。

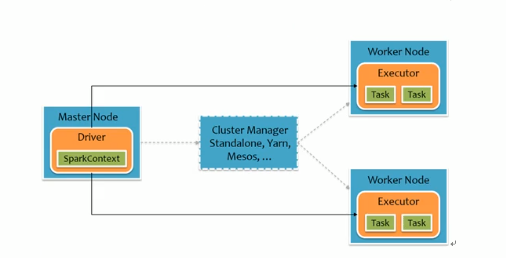
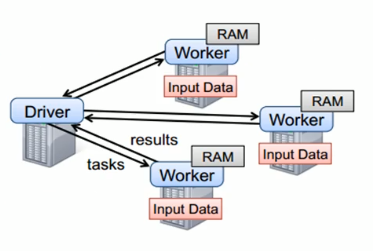
# RDD编程模型

## 2.1 编程模型

要使用Spark，开发者需要编写一个Driver程序，它被提交到集群以调度Worker。Driver中定义了一个或多个RDD，并调用RDD上的action，Worker则执行RDD分区计算任务。

Driver：定义并调用RDD的action

Worker：执行RDD的分区计算任务



## 2.2 RDD创建

从集合中创建

从外部存储中创建

从其他RDD创建

### 2.2.1 从集合中创建

parallelize和makeRDD 函数

1. parallelize

val rdd = sc.parallelize(Array(1,2,3,4,5,6))

1. makeRDD

val rdd = sc.makeRDD(Array(1,2,3,4,5,6))

makeRDD底层也是调用parallelize方法

### 2.2.2 从外部存储中创建

testFile函数

可以从本地文件系统，也可以从HDFS等分布式文件系统中读

### 2.2.3 从其他RDD创建