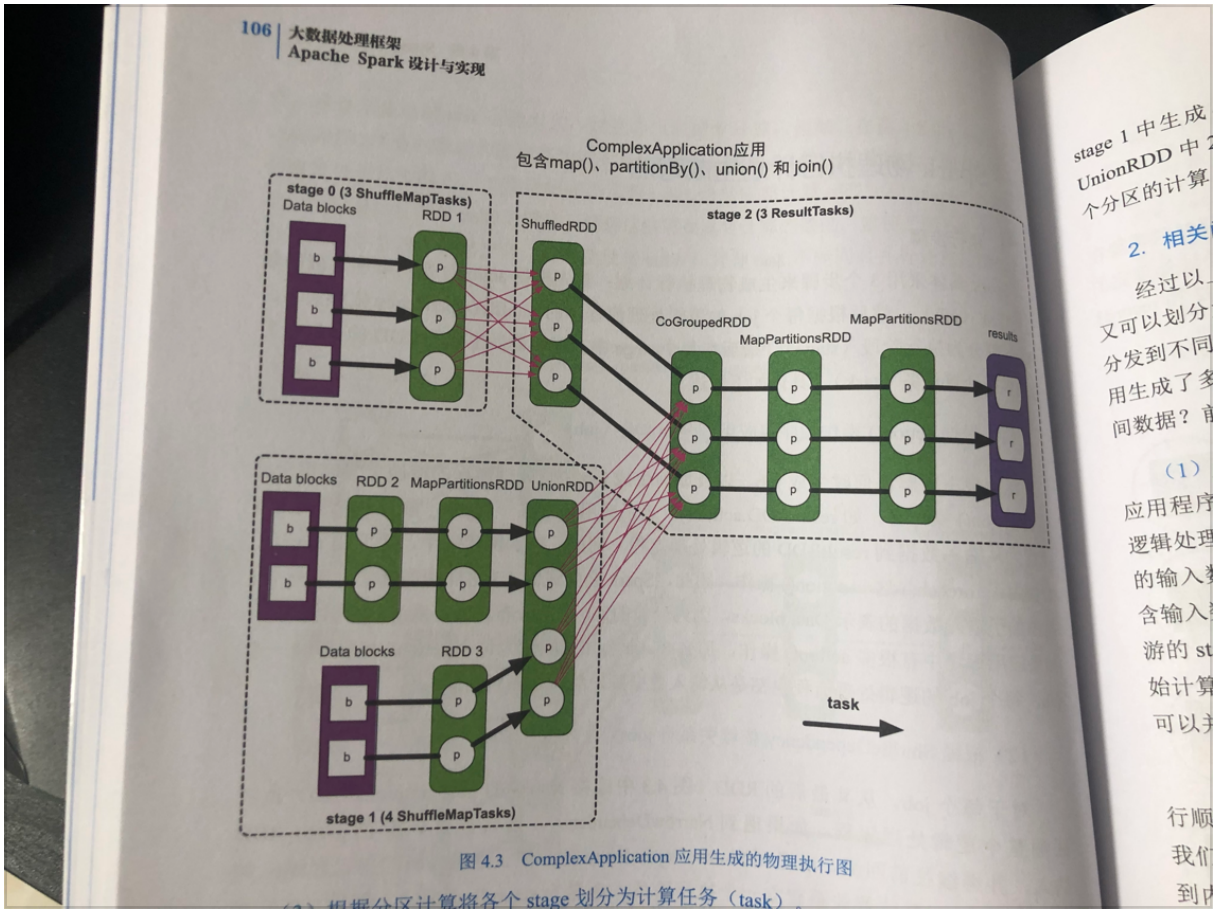
Spark - 任务的执行过程

物理执行计划



任务执行的三种想法

- 将每个箭头当成一个Job，前后关联的RDD作为一个stage：存在性能问题
- 所有阶段称为一个stage，中间RDD产生之后用于下一步计算，即算完成删除上一步产生的RDD：产生重复计算
- Spark的做法：对ShuffleDependency（宽依赖）进行划分，宽依赖之前形成一个stage，之后形成新的stage

物理执行计划生成步骤

1，根据action操作生成Job，每个action对应一个Job
2，根据宽依赖划分stage（执行阶段）
3，根据分区划分task（计算任务）

计算顺序

RDD.action()之后生成Job，逻辑处理流程为DAG图，不同stage之间可并行，从输入数据的stage开始，本例中stage0，1先开始，执行完之后stage2开始执行

task内部数据存储与计算细节

Task内部的RDD调用不同函数，会产生不同的依赖关系，不同依赖关系存储计算细节不同

- 1对1，流水线式操作，只保存上一步执行完之后的结果，之前的删除，节约内存，如果一个RDD分为多个record，只需要在内存保存一个
- 多对多，需要保存前一个RDD所有record，而下一步函数根据复杂程度确定需要消耗资源的程度

不同task之间数据传递

- 上游的stage把数据传递给下游stage时，会根据下游stage中task的数量进行分区（Hash，Range），并把数据写入分区中（Shuffle Write）
- 下游stage从上游stage分好的区域中读取数据（Shuffle Read）

stage里task的划分细节

- stage划分是根据何时发生宽依赖
- task是根据分区进行划分的，RDD间的不同依赖关系会导致划分有所不同
  - 一对一，每个分区生成一个task
  - 同一个stage中，RDD1对RDD2 ManyToMany或ManyToOne：每个task读取多个parent数据，处理一个child数据
  - 宽依赖发生时的ManyToOne和ManyToMany：也是根据child数量生成task