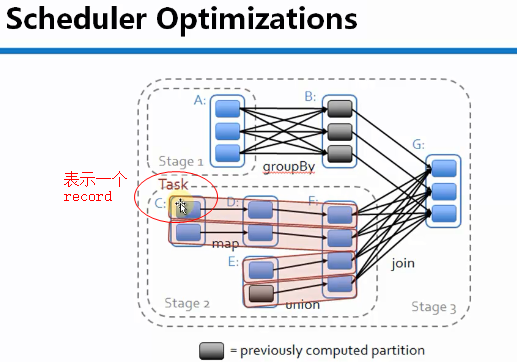
**一：再次思考pipeline**

即使采用pipeline的方式，函数f对依赖的RDD中的数据集合的操作也会有两种方式：

1. f(record),f作用于集合的每一条记录，每次只作用于一条记录；
2. f(records),f一次性作用于集合的全部数据；

Spark采用是是第一种方式，原因：

1. 无需等待，可以最大化的使用集群的计算资源。因为C一个record算完算D -> F，不需要等父RDD中所有的record都计算完之后再计算子RDD。
2. 减少OOM的发生；因为每次算一个record，线性的执行。
3. 最大化的有利于并发；
4. 可以精准的控制每一个Partition本身(Dependency)及其内部的计算(compute)。
5. 基于lineage的算子流动式函数式编程，节省了中间结果的产生，



一次只算一个record,c上面的一个record计算完，流到D，D计算完流到F这样就不会产生中间结果。并且可以最快的恢复。

疑问:

会不会增加网络通信？当然不会！因为在pipeline！

**二：思考Spark Job具体的物理执行**

Spark Application里面可以产生1个或者多个Job，例如：spark-shell默认启动的时候内部就没有job，只是作为资源的分配程序，可以在spark-shell里面写代码产生若干个Job，普通程序中一般有不同的Action，每一个Action一般也会触发一个Job。有些action也会触发其他action的实现。

Spark是MapReduce思想的一种更加精致和高效的实现。MapReduce有很多不同的实现，例如Hadoop的MapReduce的基本流程如下：首先是以JVM为对象的并发执行的Mapper,Mapper中map的执行会产生输出数据，输出数据会经过Partition指定的规则放到Local FileSystem中，然后再经由Shuffle，Sort, Aggregate变成Reducer中的reduce的输入，执行reduce产生最终的执行结果。Hadoop MapReduce执行的流程虽然简单，但是过于死板，尤其是在构建复杂算法(迭代)的时候非常不利于算法的实现，且执行效率极为低下。

Spark算法构造和物理执行时最基本的核心：**最大化pipeline！**

**基于Pipeline的思想，数据被使用的时候才开始计算，从数据流动的视角来说，是数据流动到计算的位置！！！实质上从逻辑的角度来看，是算子在数据上流动！**

从算法的构建角度而言：肯定是算子作用于数据，所以是算子在数据上流动；**方便算法的构建。**

从物理执行的角度而言：是数据流动到计算的位置！方便系统最为高效的运行！

**从人思考的角度讲，我们在计算数据的时候，算子的计算会分为一步，两步，三步。。。**

**算子在数据上流动，这是人们自然的思考方式，因此适合构建算法！！！但是内部是pipeline，数据流到算子的地方，因此可以加速系统的高效运行！！！**

对于pipeline而言，**数据计算的位置**就是**每个Stage中最后的RDD！！！！每个Stage中除了最后一个RDD算子是真实的以外，前面的算子都是假的！！！**

由于计算的Lazy特性，导致计算从后往前回溯，形成Computing Chain,导致的结果就是需要首先计算出具体一个Stage内部最左侧的RDD中本次计算依赖的Partition,在算子发送给executor之前，所有的算子已经合并成一个算子了，假设有五千步，那么内部就是第一步，第二步，第三步。。。。。其实他还是按照这个步骤去执行。只不过它就是一个算子在线程中它就是一个函数而已。

三：窄依赖的物理执行内幕

一个stage内部的RDD都是窄依赖，窄依赖计算本身是从逻辑上看，是从stage内部最左侧的RDD开始立即计算的，根据计算链条Computing Chain,数据(record)从一个计算步骤流动到下一个计算步骤，以此类推，直到计算到Stage内部的最后一个RDD来产生计算结果。

Computing Chain是从后往前回溯构建而成的，而计算实际的物理计算则是让数据从前往后在算子上流动，直到流动到不能再流动位置才开始计算下一个record。这就导致一个美好的结果：后面的RDD对前面的RDD的依赖虽然是Partition级别的数据集合的依赖，但是并不需要父RDD把Partition中所有的Records计算完毕才整体往后流动数据进行计算，这就极大的提高了计算速率。

四：宽依赖物理执行内幕：

**必须等到依赖的父Stage中的最后一个RDD把全部数据彻底计算完毕才能够经过shuffle来计算当前的Stage。例如：groupByKey的时候，必须等计算完才能分组。**

作业：

**我理解的Spark Job执行**