> 当你认知有局限的时候，要做成一件事情要耗费大量的能量

RDD编程入门

RDD定义

> RDD（ResilientDistributedDataset）叫做弹性分布式数据集，是Spark中最基本的数据抽象。代码中是一个抽象类，它代表一个弹性的、不可变、可分区、里面的元素可并行计算的集合。

RDD概念很重要，主要理解它的弹性，不可变，可分区什么意思就行

RDD特点

弹性

```
存储的弹性：内存与磁盘的自动切换；
容错的弹性：数据丢失可以自动恢复；
计算的弹性：计算出错重试机制；
分片的弹性：可根据需要重新分片。
```

分区

RDD逻辑上是分区的，每个分区的数据是抽象存在的，计算的时候会通过一个compute函数得到每个分区的数据。如果RDD是通过已有的文件系统构建，则compute函数是读取指定文件系统中的数据，如果RDD是通过其他RDD转换而来，则compute函数是执行转换逻辑将其他RDD的数据进行转换。

只读

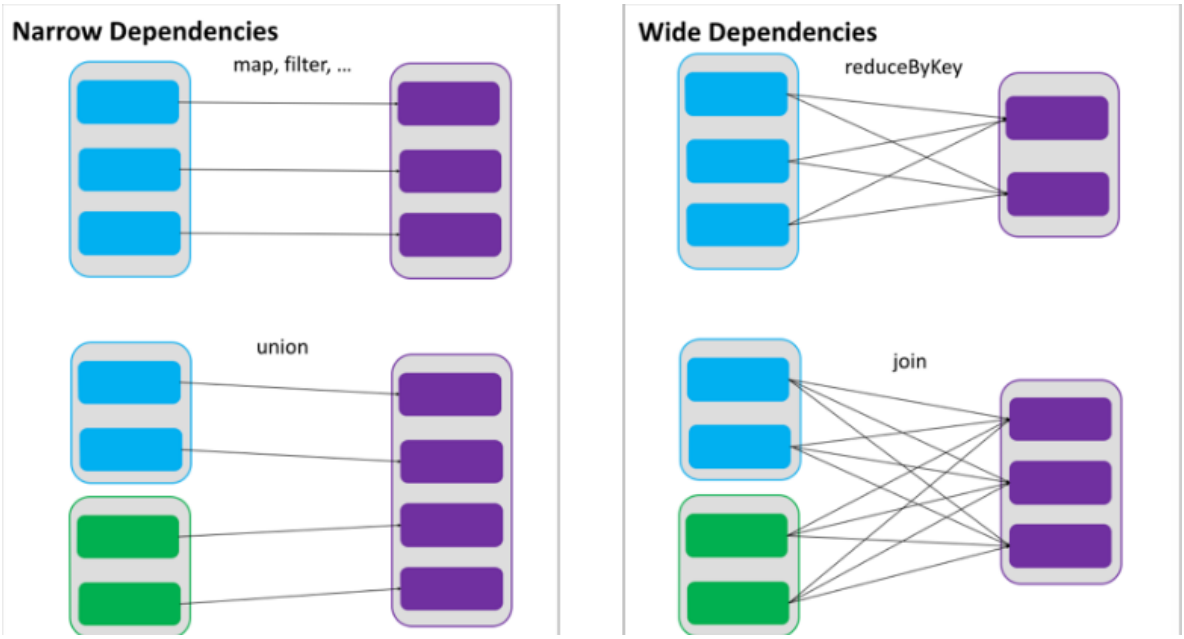
RDD是只读的，要想改变RDD中的数据，只能在现有的RDD基础上创建新的RDD。由一个RDD转换到另一个RDD，可以通过丰富的操作算子实现，不再像MapReduce那样只能写map和reduce了。

依赖

RDDs通过操作算子进行转换，转换得到的新RDD包含了从其他RDDs衍生所必需的信息，RDDs之间维护着这种血缘关系，也称之为依赖。如下图所示，依赖包括两种，

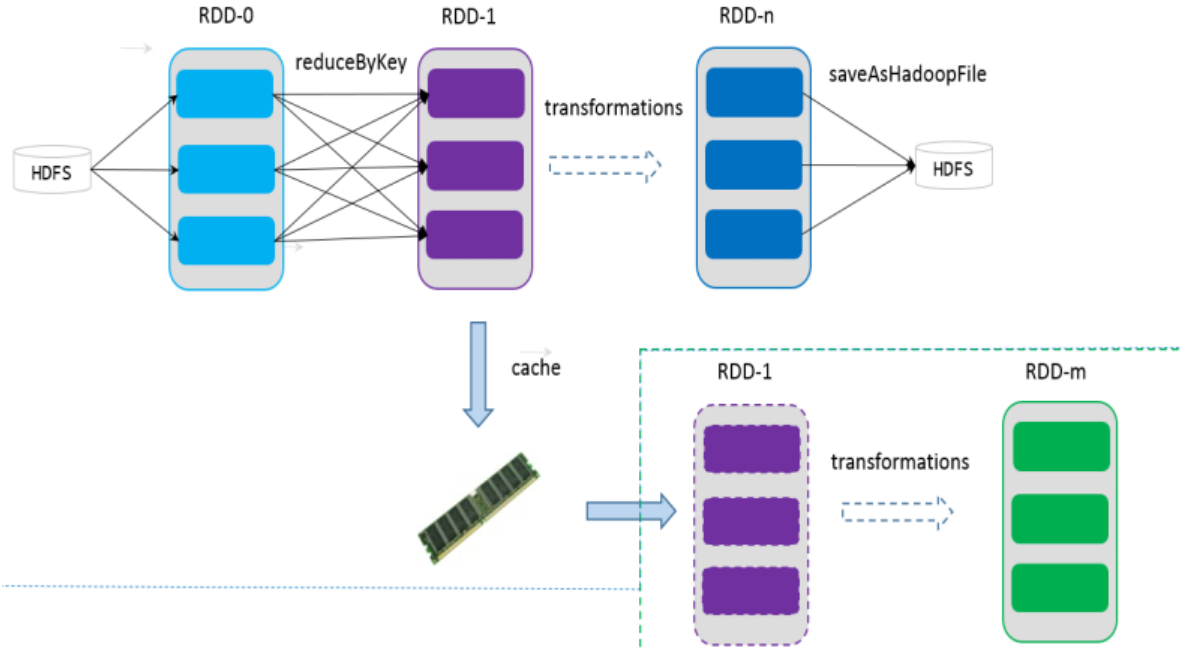
窄依赖：RDDs之间分区是一一对应的，

宽依赖：下游RDD的每个分区与上游RDD(也称之为父RDD)的每个分区都有关，是多对多的关系。



缓存

如果在应用程序中多次使用同一个RDD，可以将该RDD缓存起来，该RDD只有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的时候，会直接从缓存处取而不用再根据血缘关系计算，这样就加速后期的重用。如下图所示，RDD-1经过一系列的转换后得到RDD-n并保存到hdfs，RDD-1在这一过程中会有个中间结果，如果将其缓存到内存，那么在随后的RDD-1转换到RDD-m这一过程中，就不会计算其之前的RDD-0了。



CheckPoint

虽然RDD的血缘关系天然地可以实现容错，当RDD的某个分区数据失败或丢失，可以通过血缘关系重建。但是对于长时间迭代型应用来说，随着迭代的进行，RDDs之间的血缘关系会越来越长，一旦在后续迭代过程中出错，则需要通过非常长的血缘关系去重建，势必影响性能。为此，RDD支持checkpoint将数据保存到持久化的存储中，这样就可以切断之前的血缘关系，因为checkpoint后的RDD不需要知道它的父RDDs了，它可以从checkpoint处拿到数据。

RDD编程模型

> 在Spark中，RDD被表示为对象，通过对象上的方法调用来对RDD进行转换。经过一系列的transformations定义RDD之后，就可以调用actions触发RDD的计算，action可以是向应用程序返回结果(count, collect等)，或者是向存储系统保存数据(saveAsTextFile等)。在Spark中，**只有遇到action，才会执行RDD的计算(即延迟计算)**，这样在运行时可以通过管道的方式传输多个转换。
>
> 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运行Worker，如下图所示。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。

只有遇到Action算子才会触发计算！！！

RDD创建方式

(一)从集合中创建

从集合中创建RDD，Spark主要提供了两种函数：parallelize和makeRDD

（1）使用parallelize()从集合创建

```
// 这里的sc就是SparkContext，代表Spark的上下文环境
scala> val rdd = sc.parallelize(Array(1,2,3,4,5,6,7,8))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at
<console>:24
```

（2）使用makeRDD()从集合创建

```
scala> val rdd1 = sc.makeRDD(Array(1,2,3,4,5,6,7,8))
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at makeRDD at
<console>:24
```

(二)从外部存储系统中的数据创建

包括本地的文件系统，还有所有Hadoop支持的数据集，比如HDFS、Cassandra、HBase等

```
scala> val rdd2=sc.textFile("hdfs://master1:8020/data.txt")
rdd2:org.apache.spark.rdd.RDD[String]=hdfs://master1:8020/data.txtMapPartitionsR
DD[4]attextFileat<console>:24
```

(三)从其它RDD转换得来

RDD经过某些算子转换还是RDD，下一文档讲解