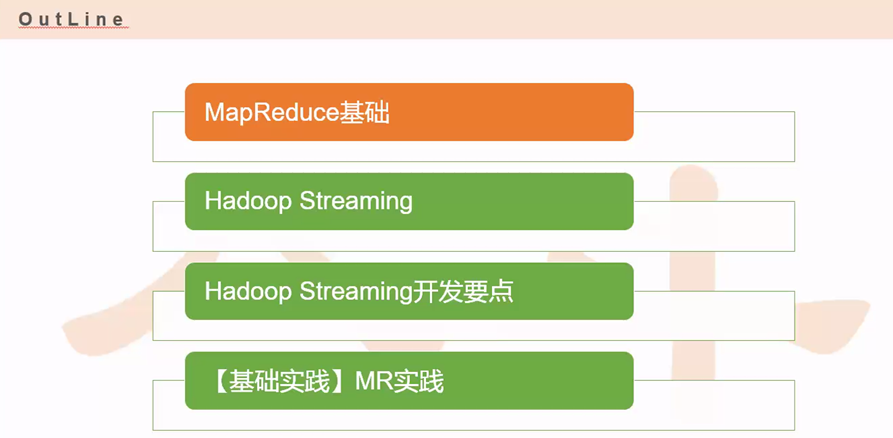
MapReduce进阶1

制作人：八斗大数据黄海峰



1. 今天我们继续讲MapReduce的一个进阶，相当于基于上次MapReduce基础课之上再去做进一步的深入。
2. 那么今天我们的整个的课程大纲是分四个部分（如下图1所示）

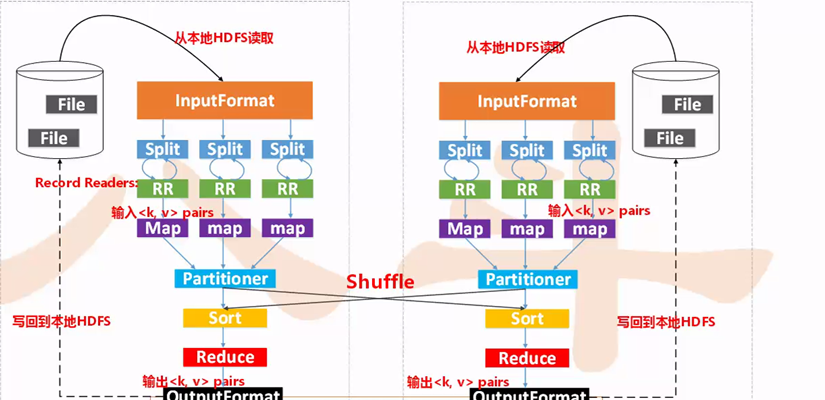
图1：



那么第一部分是基于上次MapReduce的基础课之上换一种角度再进一步的进行巩固和知识点的丰富，那在第二部分的目的就是因为我们最后要进入一个MapReduce的一个代码实践部分，相当于是第二部分和第三部分都是给最后的代码实践做准备工作的，因为我们目前写这个MapReduce是以python进行开发的而不是用java去开发的，那么用python去开发MapReduce的时候需要用到一个叫Hadoop Streaming的这个一个工具。

1. 今天会用另外一种表达方式，换一个角度，把上节课讲的一些图里面的涉及到的各个的一个细节和环节再做一些细致的拆解，这样的话有助于加深印象（如下图2所示）

图2：



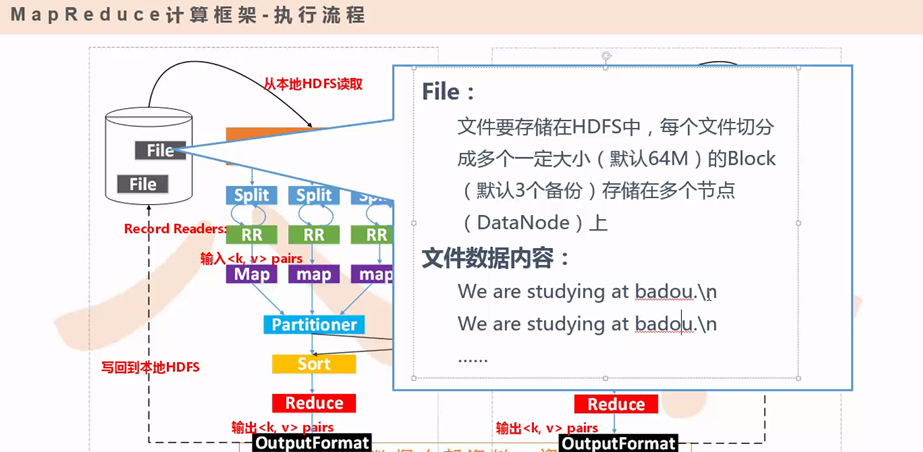
在这个图2里面看上去很复杂，但是拆解完了以后从头到尾学完之后就贼清晰了。

在这个图里面有6个Map和两个Reduce加起来就有8个独立的进程空间，当这个MapReduce跑起来的时候，是需要占用8个进程的空间，那这些进程中有可能在同一个节点上或者同一个机器上运行的也有可能都不在同一个节点上不同机器上运行的，说到进程这个事情，为何MapReduce采用的是多进程并发方式而不是多线程的并发方式呢？首先这个多进程并发的模型便于每一个任务去占用资源去进行控制，因为这个进程空间是独享的。

那么进程有什么样子的优势？可以方便的对于每一个任务进行控制和调配这是进程的优点，那么缺点就是进程相对比线程来说它会消耗更多的启动时间，也就是说进程的启动要比线程启动要慢很多，所以MapReduce只适合做一些批量操作，适合高吞吐的情况下不能寄托它太多的实效性。

如图二里面左边和右边可以认为是两个机器两个节点，相当于这两个不同的机器同时并发执行。那么MapReduce的数据源是来自HDFS上面的，数据源都是以File的方式表示，那这里面就涉及到了HDFS的一些概念，当你提交一个大文件，这个文件会被切分成多个小的文件，那么HDFS是一个文件系统，文件系统里面它就有一个数据单位，这个数据单位就是Block，默认是64M内存，并且在没有配置的情况这个Block会给你做三个备份，这个三个备份存在了不同的机器上，就避免了某一台机器挂掉了之后数据丢失的问题。

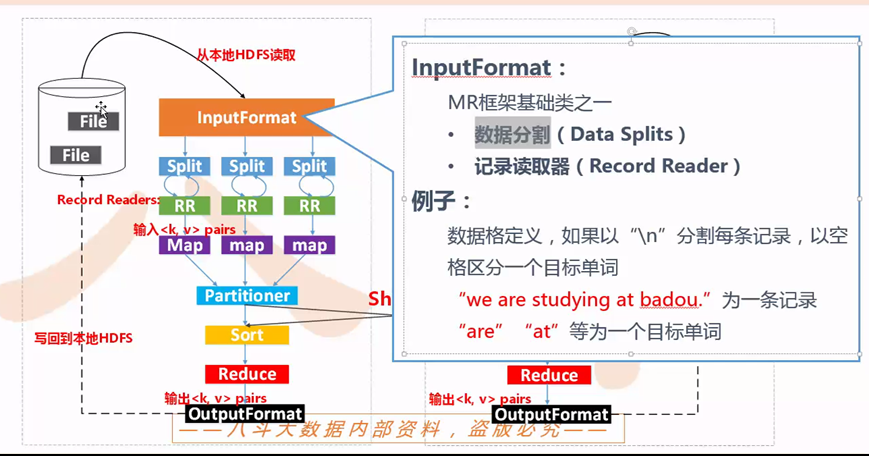
如果把一个文件打开，里面有两条句子（如下图3所示）



这句子是明文的，这明文既然能读的话的就代表根本没有带压缩，我们在通过python去开发MapReduce的时候，我们的输入数据通常都是这种明文TextFile的方式读入进来的，但是这种明文的方式缺点也有：比如这种明文数据不压缩怼到空间，会导致磁盘开销有点大，当然HDFS上不仅仅可以存放TextFile这种二进制格式还是可以存放其他格式的文件，那么这种二进制文件我们查看的时候是看不到里面写的是什么意思，那么我们要通过cat的方式去查看该文件的内容，那么我们要查看HDFS上的一个明文文件内容就通过：hadoop fs -cat /xxx，但是如果查看二进制文件就要通过：hadoop fs -test /xxx文件。使用hadoop fs -test /xxx文件去查看一个文件可以查看二进制文件也可以查看一个明文文件。

明文文件数据源，当我们数据源已经准备就绪了，需要通过一个inputFormat一个接口来进行对接，这个接口是框架帮我们支持好的，这个接口有两个功能，第一个功能就是：数据切割（Data Splits），第二个功能就是记录读取器(Record Reader)（如下图4所示）

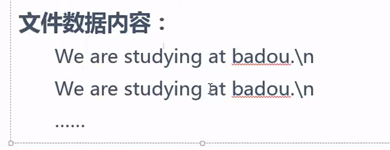
图4：



4.Map的个数依赖于这个Split这个个数，所以这一块相当于把这个原始的数据进行分割，分割成多个Split。

数据分割：就是在合适的位置对Block进行切割，保证你的句子的完整性（如下图5的两个句子）

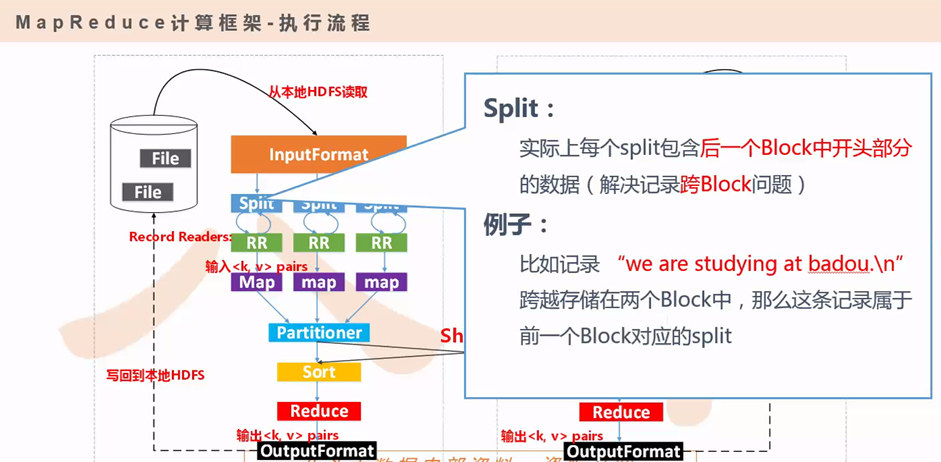
图5：



切割的时候在studying和at的中间进行切割，相当于把句子给切散了，那么前面那一段用一个Map去跑，然后后面一半的部分用另外一个Map去跑，不应该把这完整的一个句子切散。当有一个记录横跨了两个Block的时候，被截断的这个它会被划到前面的Split里面去。通常在这个Hadoop Streaming的方式去读明文的数据的时候，因为有句子中.\n分隔,它会保证这个句子不会被切散。

实际上每个split包含后一个Block中开头部分的数据(解决记录跨越block问题)如下图图6所示

图6：



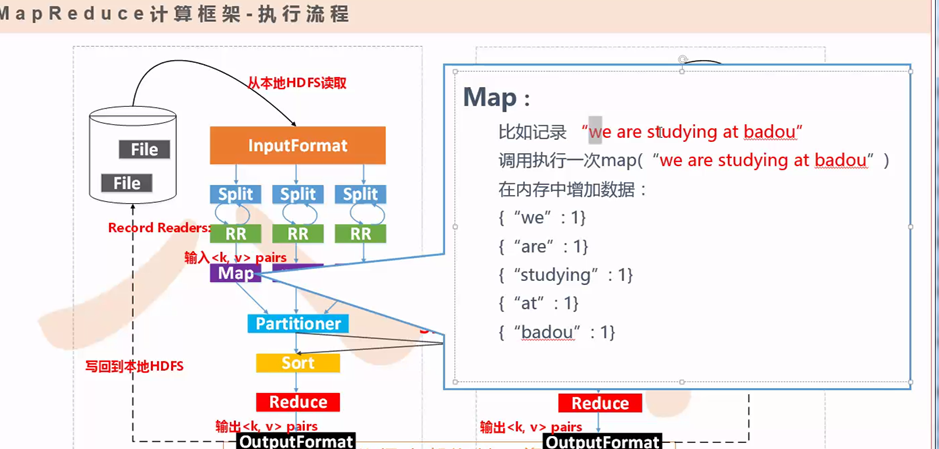
RecordReader记录读取器的意思是：一个split一个切分好了，就应该用map进程把切分好的split用标准输入导入进来，标准输入在你使用map逻辑的时候，这里面会调用map函数方法，这个函数方法会传一个值，这个值叫value，这个value代表（如下图7所示）的例子里面的红色的句子

图7：



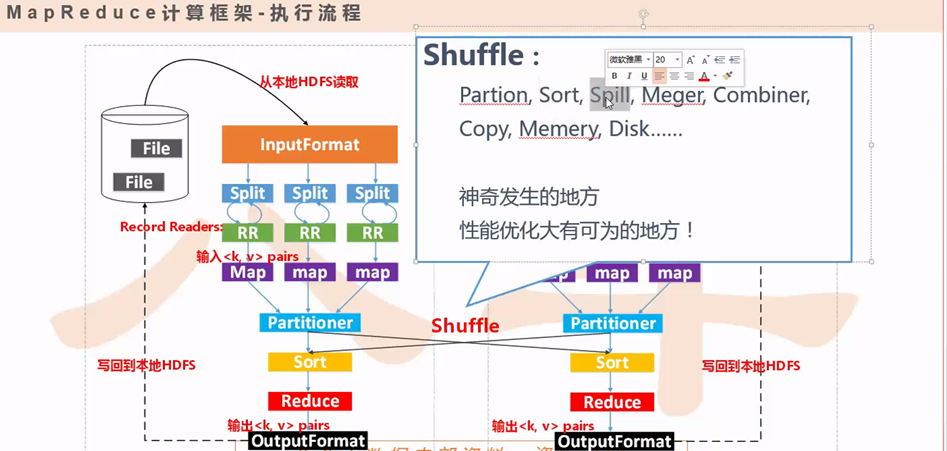
一个文档有很多的句子，一条句子一行，一行调一次map，相当于这个指针从这个文档的从头到尾，所以当执行每一条句子（记录）的时候就会调一次map，通过人为在这个map里面赋予一些逻辑，对这个句子进行分词处理，把每一个句子的单词取出来做一个计算，相同的加1（如下图8所示）

图8：



然后把这些数据读进来以后，有一个重要的环节，那就是shuffle环节（如下图9所示）

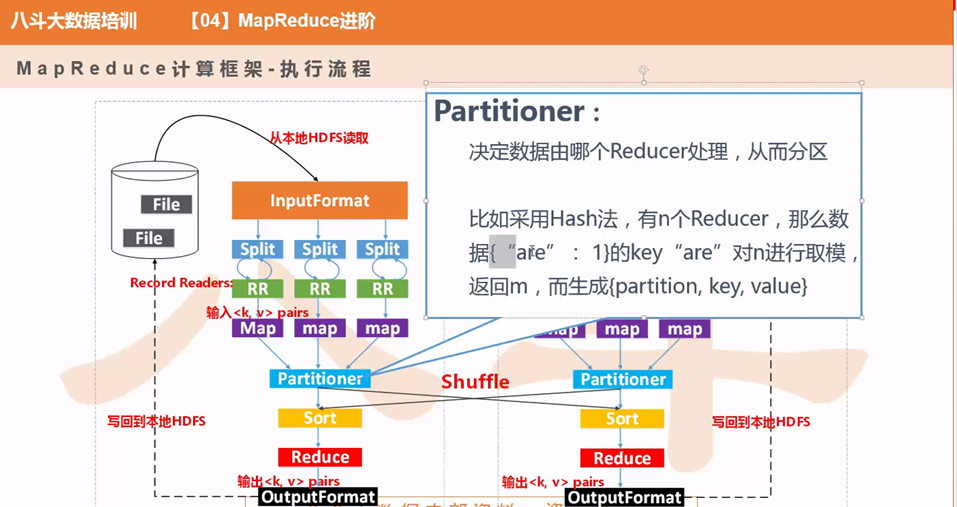
图9：



5.Partitioner是属于Shuttle中的一部分，shuttle是整个MapReduce一个很重要的环节。

Map中要区分key和value，key是通过hash的方法来决定未来你这条记录要分配到哪个Reduce里面去（如下图10所示）

图10：



在这个Partitioner处理是在Map缓冲区Buffer里面做的，内存缓冲区默认是100m，如果100M内超过了80M，就会把这80M的数据给锁住，然后将这些数据进行排序然后能够清空内存并且写入本地磁盘（如下图11和图12图13所示）

图11：

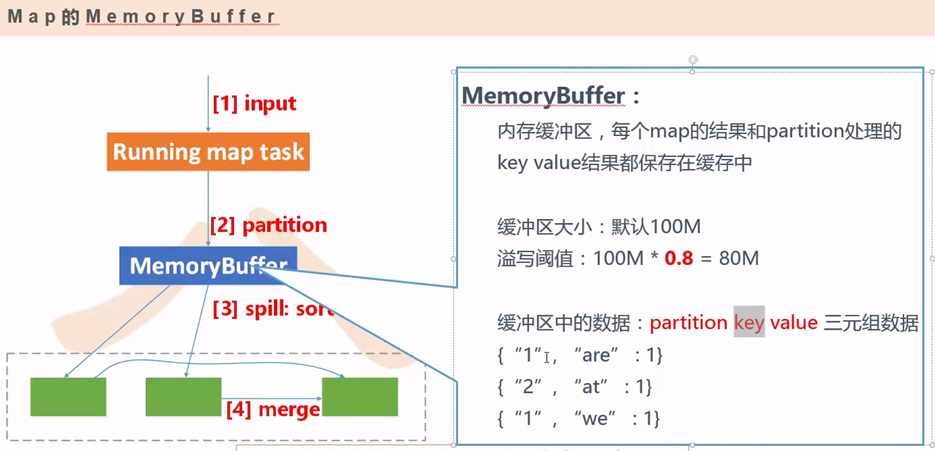


图12：

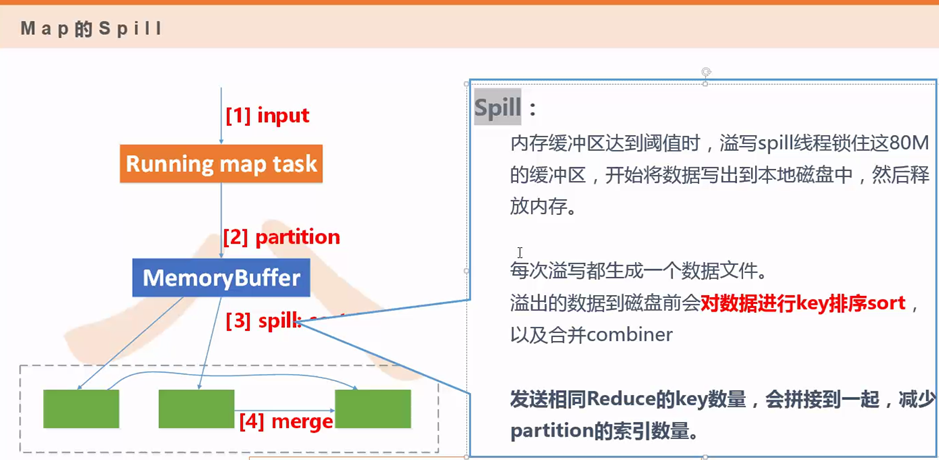


图13：

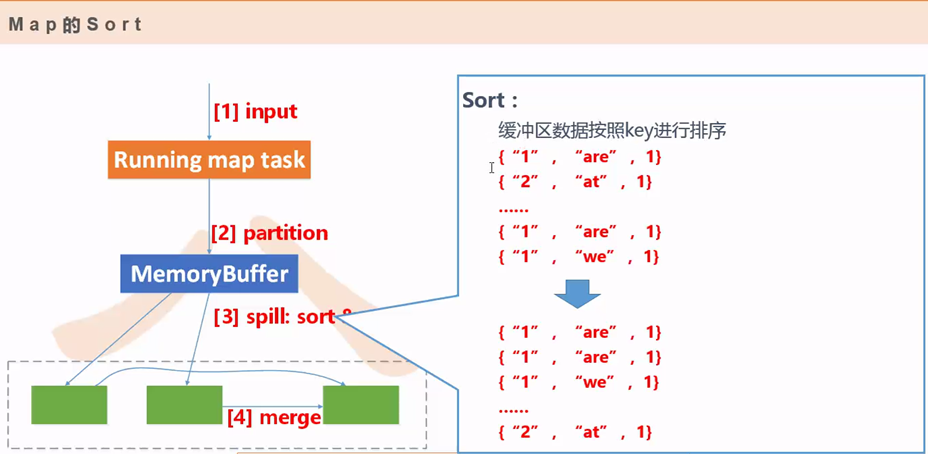
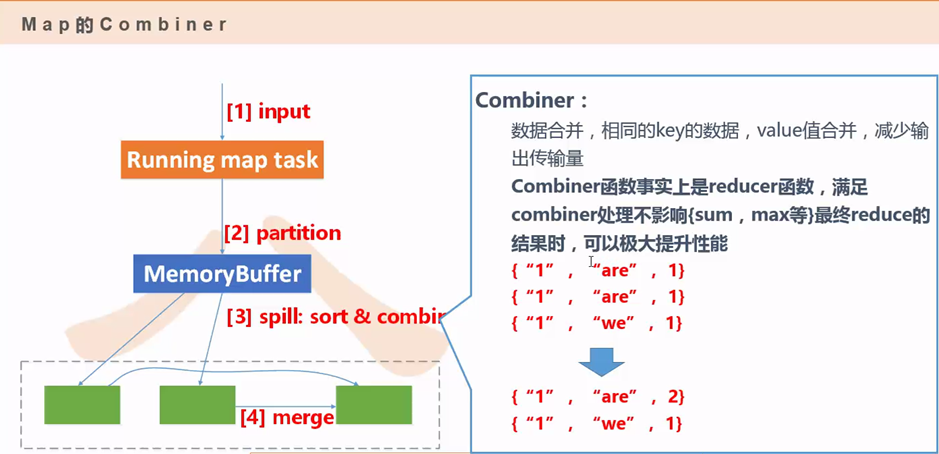


图13是把key进行排序，把相同的key放在一起并且对值进行加加（++）操作

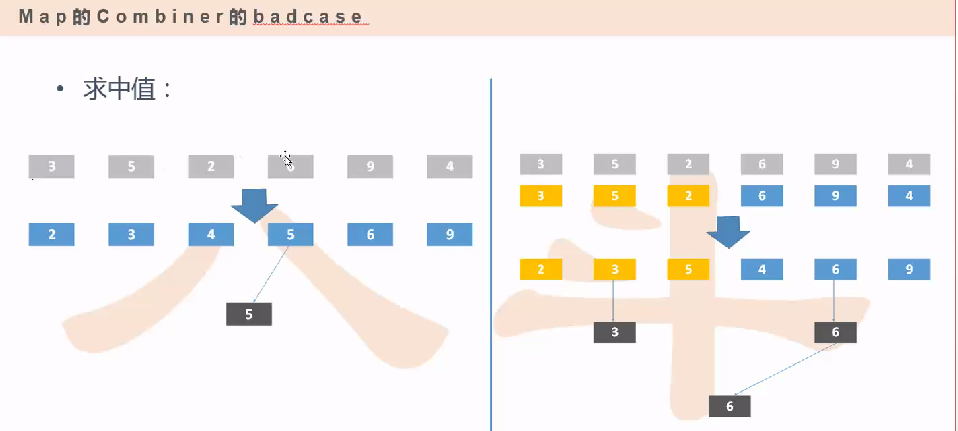
1. Combiner:相当于把部分的redeuce提前在map阶段做，具体怎么做的呢？（如下图14所示）

图14：



红色上面的还没合并的数据可以理解为还是在map缓冲区里面还没进行数据合并操作，如果说把相同的key提前累加起来是不是不会影响我最后的统计结果，通过把相同的key进行合并，合并完了以后再从磁盘上产生的数据要远远小于我不合并前的数据，这样就算是一个很大的优化，提高了数据的传输效率。但是Combiner这个东西不能乱用，举个例子：

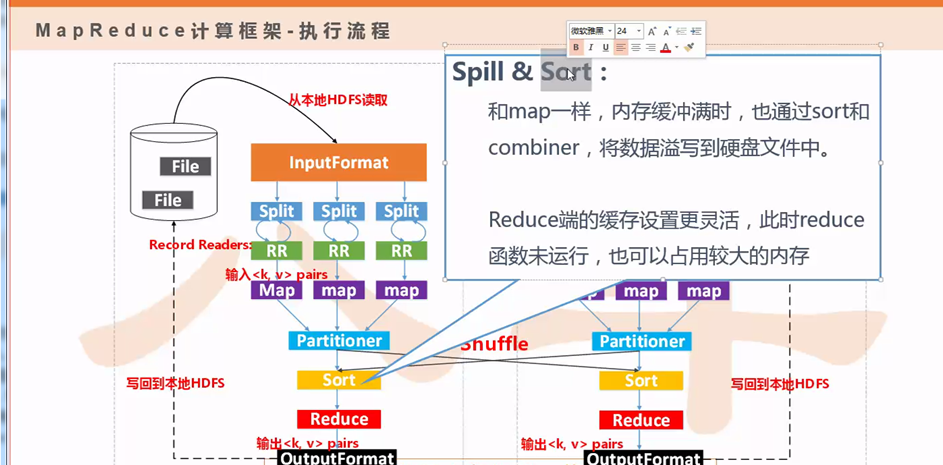
在解决一些特殊的问题的时候就可以用，因为可以提高效率，反之如果用的话就会导致你的程序你的代码会有一个错误的结果，比如一个求中值例子（如下图15所示）

图15：  


上面图15的求中值的例子可以得到一个求中间的值，求中间值的时候要先把数据进行排序，然后在2，3，4，5，6，9中取出最大的中值那就是5，然后如果第二个小图右手边的图，三个值为一组进行取值，那么两对三个值取出来的中间值都是3和6，然后按照最大值就是6为中间值，那么导致右手边图跟左手边图的中间值都不一样，所以Combiner是好用，但是也不能乱用。

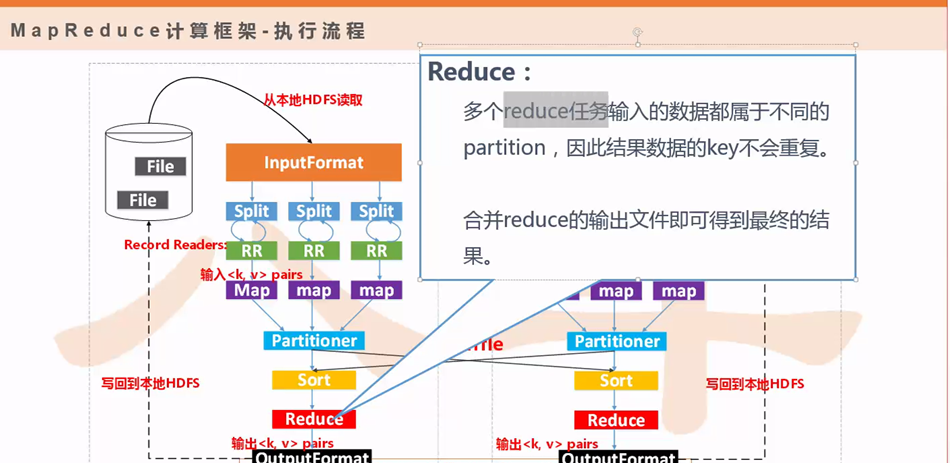
1. Reduce里面也有一个内存缓冲区，它这个内存缓冲区跟Map的内存缓冲区一样（如下图16和图17所示）

图16：



真正到了Reduce里面，也就是说因为你的Partitioner是已经提前做了，那肯定会保证相同的Partitioner这个id记录，肯定会统一发送到一个后面相同id的Reduce上面去进行进一步处理（如下图17所示）

图17：

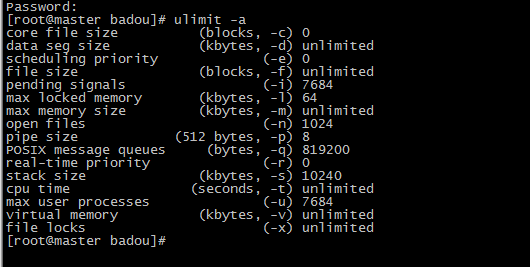


8.跑MapReduce的时候需要注意的几个点（如下图18所示）

图18：



1｝首先跟ulimit系统的参数有关（如下图19所示）

图19：  


它里面都是一些系统已经设定的参数号。

2｝跑MapReduce系统，肯定要是多核的，一台机器要启动多个map和Reduce，因为一个进程一个核，为了高并发的情况下就要多核

3｝内存就越大越好，一般公司服务器都是32G和64G

1. 下面的图20就要重点关注一下

图20：



我们这个机器它的一些硬件的一些资源，它直接制约着你的slot

slot的意思是可以认为是一个槽位，例子：你去餐馆吃饭，如果你中午去的时候人就比较多，那这餐馆里面只有10个凳子里面已经有10个人正在吃饭了，那没办法你只能在外面排队，当其中一个人吃完了这个位置空了，你就可以进去，那么这个凳子就可以认为一个槽位。

1. 我们在开发大数据的时候，开发MapReduce的时候，会经常遇到这样的一个需求，那就是配置的问题（如下图21所示）

图21：



那配置有大有小，小的尽量本地上传，本地上传就是-file的方式去上传。

大的就需要提前把这个数据放到你的HDFS上（后面会有实践，经过实践会有更深的体会）

1. 总结（如下图22所示）

图22：



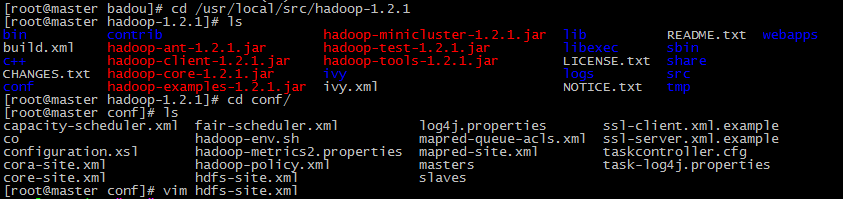
压缩文件不可切分：就是说你去面试，问你，是所有的文件都要经过split吗？答案是：不是，什么情况下不是呢？那就是这个数据源是一个压缩文件，压缩文件是没办法做split的，所以这个压缩文件是大还是小，都必须由一个map来处理，一个压缩文件对应一个map。

压缩文件是你故意要做的，因为一个压缩文件对应一个map，如果要是严格控制map的个数，比如说搞100个map，那是不是就是搞100个压缩文件就可以了。所以压缩文件就是可以控制map的一个很好的方法

但是你如果是使用明文testFile的方式的话你想要加大一些map的个数，你就得去调一些参数，临时把block的单位调整一下，但是再怎么调整，我就希望100个Map并发，那你再怎么去调整其实都调整不了一个准确的数字，所以通过一个压缩文件的方式是可以控制map的个数

Block默认大小是64m,但是这个默认的大小是可以改的（那么这个怎么改呢？请看下面的图23图24和相关的命令行输入），注意要修改的话对每台机器都要进行修改

图23：



命令行输入：cd /usr/local/src/hadoop-1.2.1

命令行输入：ls

命令行输入：cd conf/

命令行输入：vim hdfs-site.xml

编辑打开hdfs-site.xml 文件以后输入：

<property>

<name>dfs.block.size</name>

<value>512000</value>

</property>

图24：

