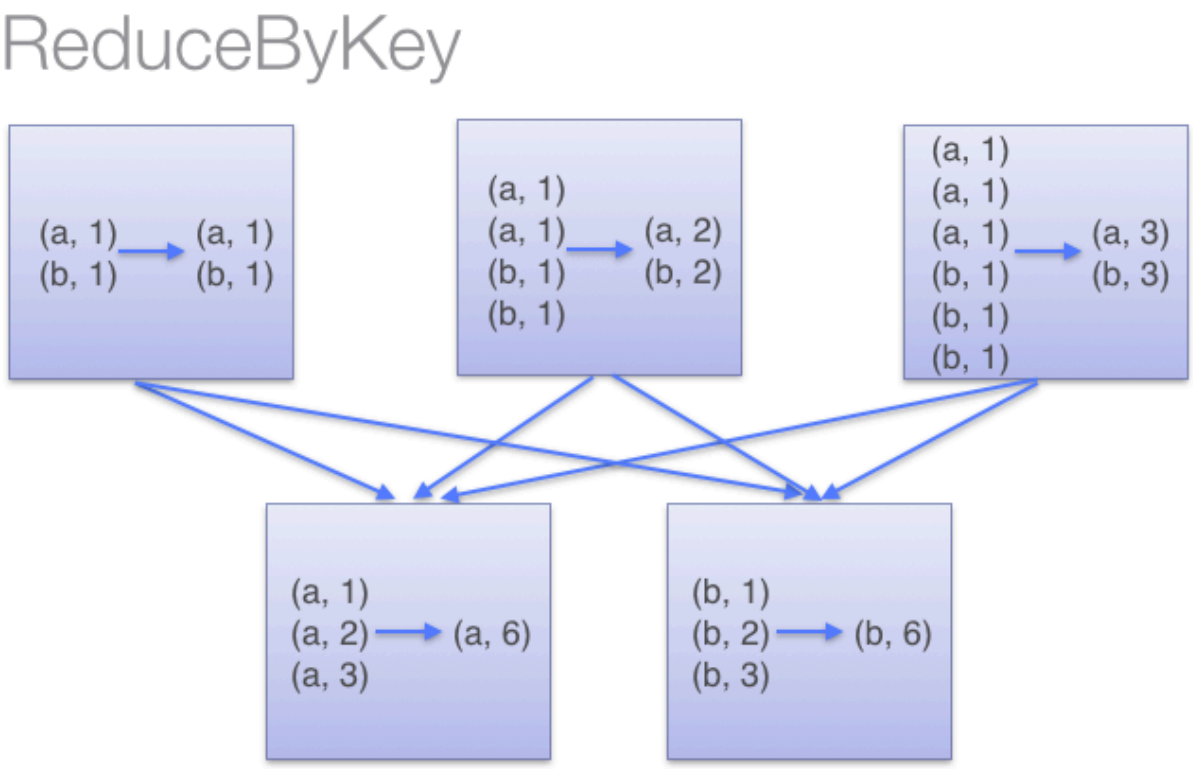
在Spark中尽量少使用GroupByKey函数

为什么建议尽量在Spark中少用GroupByKey，让我们看一下使用两种不同的方式去计算单词的个数，第一种方式使用 reduceByKey ；另外一种方式使用groupByKey，代码如下：

```
# User: 过往记忆
# Date: 2015-05-18
# Time: 下午22:26
# bolg:
# 本文地址：/archives/1357
# 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
# 过往记忆博客微信公共帐号：iteblog_hadoop

val words = Array("one", "two", "two", "three", "three", "three")
val wordPairsRDD = sc.parallelize(words).map(word => (word, 1))

val wordCountsWithReduce = wordPairsRDD
  .reduceByKey(_ + _)
  .collect()

val wordCountsWithGroup = wordPairsRDD
  .groupByKey()
  .map(t => (t._1, t._2.sum))
  .collect()
```

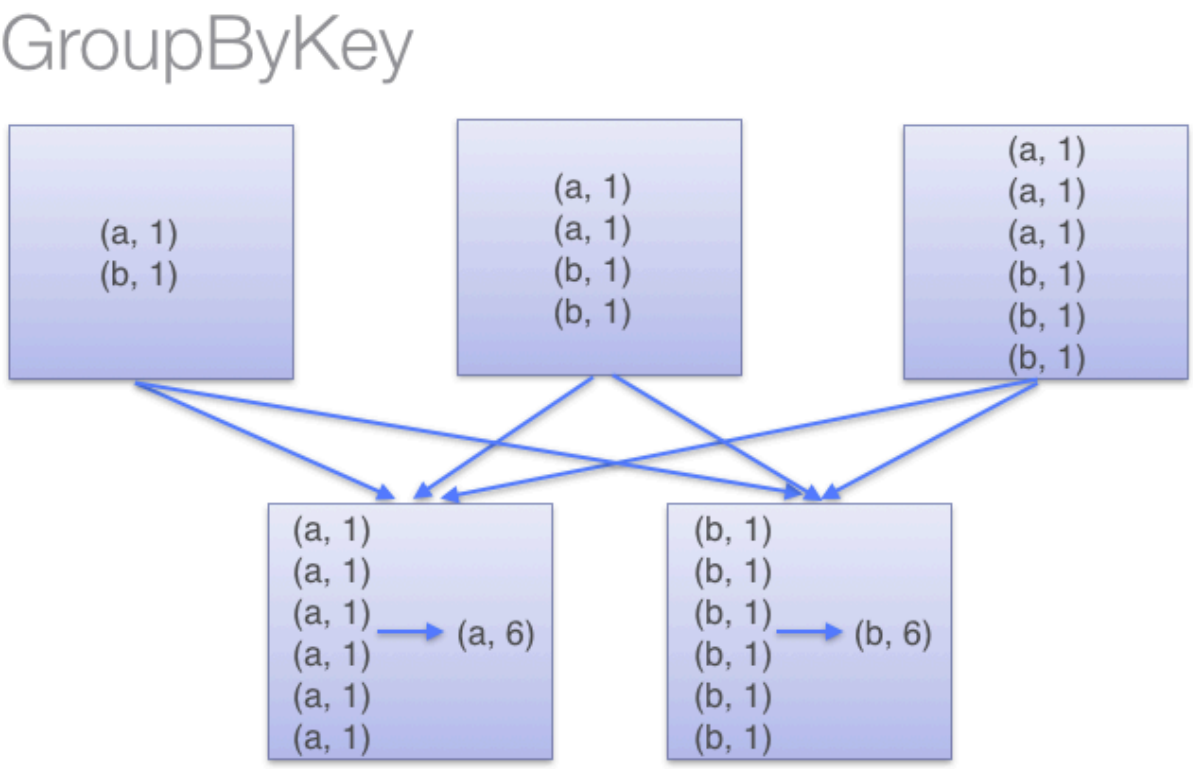
虽然两个函数都能得出正确的结果， 但reduceByKey函数更适合使用在大数据集上。这是因为Spark知道它可以在每个分区移动数据之前将输出数据与一个共用的 key 结合。

借助下图可以理解在reduceByKey里发生了什么。 注意在数据对被搬移前同一机器上同样的 key 是怎样被组合的(reduceByKey中的 lamdba 函数)。然后 lamdba 函数在每个区上被再次调用来将所有值 reduce成一个最终结果。整个过程如下：



另一方面，当调用 groupByKey时，所有的键值对(key-value pair) 都会被移动。在网络上传输这些数据非常没有必要。避免使用 GroupByKey。

为了确定将数据对移到哪个主机，Spark会对数据对的 key 调用一个分区算法。 当移动的数据量大于单台执行机器内存总量时 Spark 会把数据保存到磁盘上。 不过在保存时每次会处理一个 key 的数据，所以当单个 key 的键值对超过内存容量会存在内存溢出的异常。 这将会在之后发行的 Spark 版本中更加优雅地处理，这样的工作还可以继续完善。 尽管如此，仍应避免将数据保存到磁盘上，这会严重影响性能。



你可以想象一个非常大的数据集，在使用 reduceByKey 和 groupByKey 时他们的差别会被放大更多倍。以下函数应该优先于 groupByKey ：

（1）、combineByKey组合数据，但是组合之后的数据类型与输入时值的类型不一样。

（2）、foldByKey 合并每一个 key 的所有值，在级联函数和“零值”中使用。