在spark中,用于存储和执行任务的内存空间虽然是按照比例划分的 但是这2块内存空间是可以共享的,这里假设以下几种情况:
1.假设执行任务的内存占用较少 而需要缓存的数据较多 则此时缓存可以去占用执行任务的内存空间
2.而如果缓存空间使用较少且此时执行任务需要较多的内存的话 也可以占用存储的内存来用于执行任务,所以说是共享的
3.假设在执行任务的时候 内存中存储占用的空间达到了(JVM heap space - 300M)*60%的70%,而此时执行任务时需要较多的内存,则原先缓存在内存中的数据会被丢弃,内存空间会被回收以供任务执行,但最多回收到(JVM heap space - 300M)*60%的50% 即spark.memory.storageFraction这个参数保证了在该种情况下有(JVM heap space - 300M)*60%的50%的内存空间是不被回收的,即最多只能回收(JVM heap space - 300M)*60%的20%供任务执行,当然,这里是按需回收,假设存储占用了执行空间的100M,而执行任务的空间不够,但只需要50M就够了,那么只会回收50M的空间
4.假设在执行任务时,执行任务占用的内存空间达到(JVM heap space - 300M)*60%的70% 而此时正好需要缓存数据,发现剩余可供缓存的内存空间不够时,此时不会去回收执行任务多占用的那20%的内存,因为要保证任务执行而不是保证缓存,所以此时只有(JVM heap space - 300M)*60%*0.3的空间用于缓存

spark的内存管理主要有2个参数:
1.spark.memory.fraction,默认为0.6,用来划分执行任务和存储用得空间占整个JVM的比例,默认JVM空间中有300M预留空间,所以用于执行和存储的空间为(JVM heap space - 300MB)*0.6
2.spark.memory.storageFraction,默认为0.5,用来划分用于执行任务和存储的空间中,存储空间的比例,默认情况下(JVM heap space)*0.6*0.5为存储空间

300M内存为系统预留内存

(JVM heap space - 300MB)*40%
该区域用于存储用户数据结构和spark内部的元数据等 并可以在处理非常大的数据量时防止OOM(内存溢出)

该区域为默认为(JVM heap space - 300MB)*60%大小,是用于spark 执行任务和存储数据用的

spark的这种内存管理有以下好处:
1.不使用缓存的应用程序可以将整个空间用于执行，从而避免了不必要的磁盘溢出。
2.需要使用缓存的应用程序可以保留最小的存储空间（R），以免其数据块被全部逐出。
3.这种方法可为各种工作负载提供合理的即用型性能，而无需用户专门了解如何在内部划分内存。

spark中管理内存的类为UnifiedMemoryManager,其中:
1.getMaxMemory方法用于获取执行内存和存储内存的总大小,即(JVM heap space - 300M)*0.6
2.maybeGrowExecutionPool方法用于当Execution的内存空间不够的时候 会向storage去借 或者将storage占用Execution的空间回收回来