Spark的shufffle分析

1. AppendOnlyMap

**def** apply(key: K): V根据所给的key进⾏取值

**def** update(key: K, value: V): Unit插⼊或者更新元素,

defchangeValue(key: K, updateFunc: (Boolean, V) => V): V根据指定的key以及聚合函数,进⾏key对应的value的聚合

defdestructiveSortedIterator(keyComparator: Comparator[K]): Iterator[(K, V)]进行排序的方法

1. PartitionedPairBuffer

**def** insert(partition: Int, key: K, value: V): Unit 插入元素

**def** destructiveSortedIterator(keyComparator: Comparator[K]): Iterator[(K, V)]进行排序的方法

1. BufferedIterator 带预览版本的迭代器

**def** head: A

1. PriorityQueue 优先级队列，元素有序
2. shuffle的基本过程

插入：向map或者是buffer里面插入元素

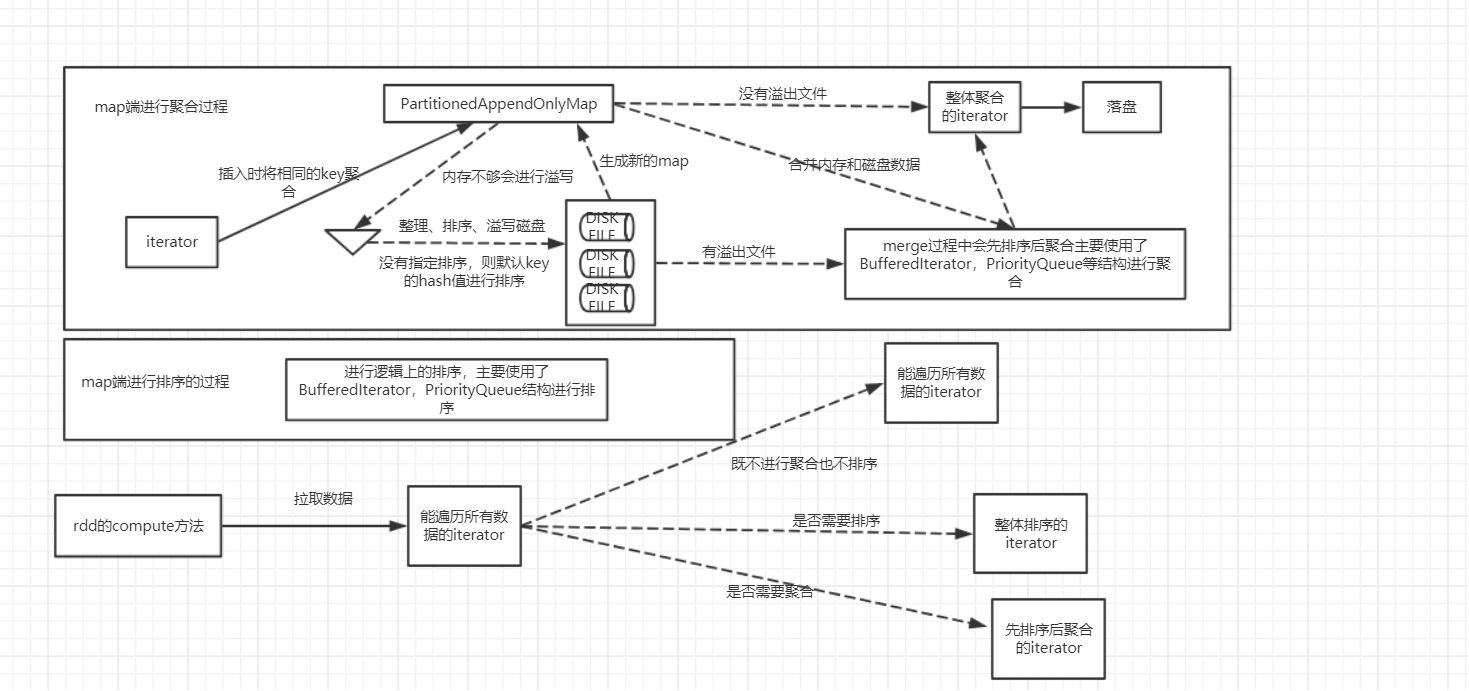
溢写：内存不够将数据溢写到磁盘

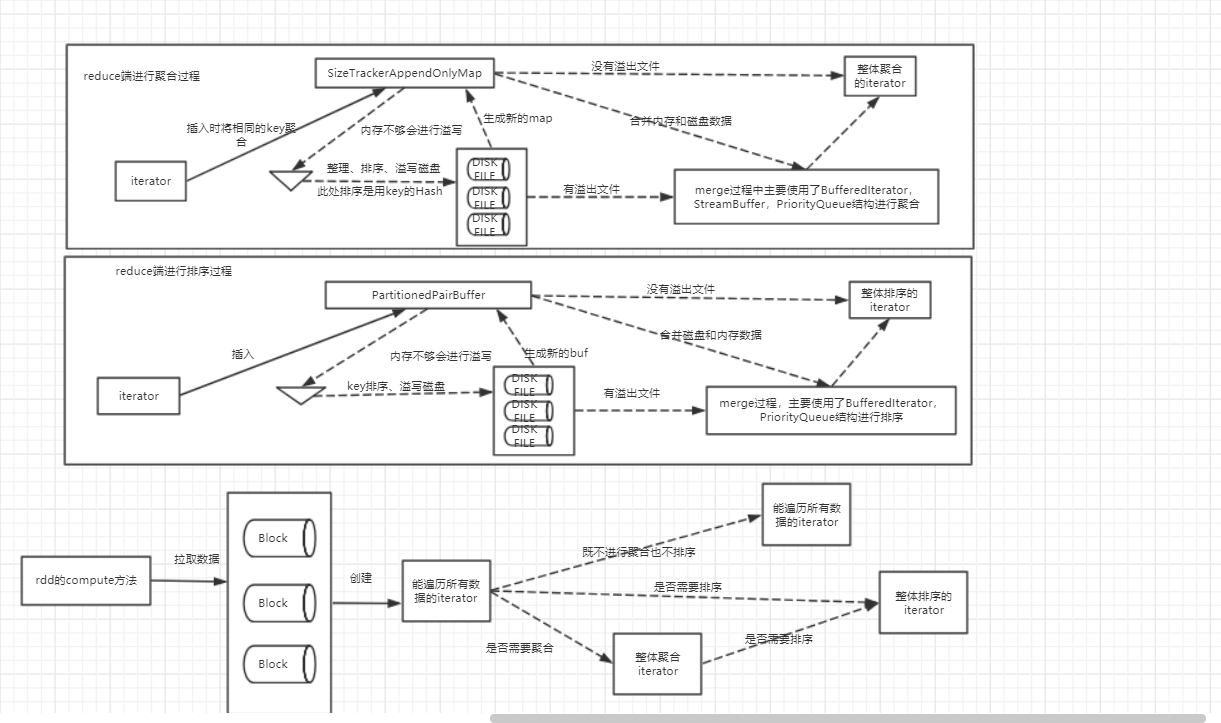
排序：溢写到磁盘之前将数据进行排序

合并：将溢写到磁盘的数据和内存中的数据进行逻辑上的合并

这是spark的shuffle的一个基本过程，一般的每个shuffle map阶段最终会生成一个结果文件和索引文件，Shuffle的reduce阶段最开始会有一个拉取数据的过程。有的shuffle算子可能会重复好几次这个过程

以下是spark的shuffle 的map和reduce的图解





1. reduceBykey和join的shuffle过程分析。

两者最大的不同在于，map里面存的数据结构不一样，最终join比reduceBykey多了一次（插入--溢写—排序--合并）的过程

1. 从数据流向看shuffle，手动模拟shuffle过程
2. 通过案例代码看shuffle的细节实现

*思考一个问题：有一个10G的数据，现在只有1G内存，现在要求按照指定的顺序进行全局排序，该怎么做？  
\* 如果又要求将这些数据中的元素按照相同的key进行聚合呢？*

1. 总结

Spark的shuffle过程从总体上说就是（插入--溢写—排序--合并）的过程，如果一次不行那就多来几次。

1、插入--溢写--排序--合并，是shuffle的基本单元，一般的每一个map任务会执行一次，最终生成按照分区排序的一个数据文件和记录每个分区字节大小的索引文件。

在每一个reduce任务也会执行一次，生成最终的迭代器，也就是rdd的compute方法返回的分区数据。常见的向combineByKeyWithClassTag等就是这样的一个过程。

但是像join等算子，因为它有多个父rdd，所以它的reduce过程会多次执行（插入--溢写--排序--合并）。

2、一般的map端和reduce端向AppendOnlyMap或者是buffer里面插入数据的时候会保存它的分区信息（（parition，k）,v），parition代表分区数。

而如果涉及到多个依赖（多个父rdd）在进行聚合的时候，在最终一次聚合过程插入的数据会保存它的依赖信息（k,(v,parentId））parentId代表哪个一个父rdd的数据，这个过程参考join的最后一次聚合过程。

3、map端溢写前会进行排序，先按照分区来排序，然后看是否指定按照key进行排序，如果没有指定会按照key的hash之进行排序，当我们在生成rdd的时候就已经知道是按照会不会按照key进行排序。常见的join，groupBy都是按照key的hash值进行排序

4、最终在生成数据文件时map端会按照分区顺序写数据，索引文件里面会记录每个分区的字节数大小。

5、在整体聚合之前，一定会进行整体的排序，在整体排序的时候会使用BufferedIterator和PriorityQueue等结构

6、迭代器是懒执行的，只有在调用它next方法时，才会去拉取一条数据，spark里面大量使用了迭代器，也时通过迭代器来实现逻辑上的排序和聚合