**Shuffle读**

我们把Shuffle读操作也称为fetch，对应图中的粉色箭头

考虑下面几个问题：

什么时候会fetch 数据?

Join操作所在的stage如何定位要fetch的数据

fetch操作和数据计算是否可以同时进行？

fetch出的数据存在哪里？

问题1：会在所有ShuffleMapTask（也就是Stage1，Stage2）执行完毕后才能执行fetch操作

问题2：会由DAGScheduler在创建Stage时在Driver端注册所有的ShuffleId，这些ShuffleId会由每个ShuffleMapTask所携带，当ShuffleMapTask执行完成后，它们会向Driver报告每个文件块的位置，保存在MapOutputTrackerMaster之中。

Reducer fetch 时就会向Driver中的MapOutputTrackerMaster询问要读取的文件块位置。如果该文件块在本地那更好，直接就读取了，如果在远程需要通过网络IO获取

问题3：边fetch就可以边运算，原理在下面说明

问题4：fetch出来的数据会缓存在一个称为SoftBuffer的结构中

如果spark.shuffle.spill=false，那么SoftBuffer会始终使用内存（AppendOnlyMap）来缓冲数据

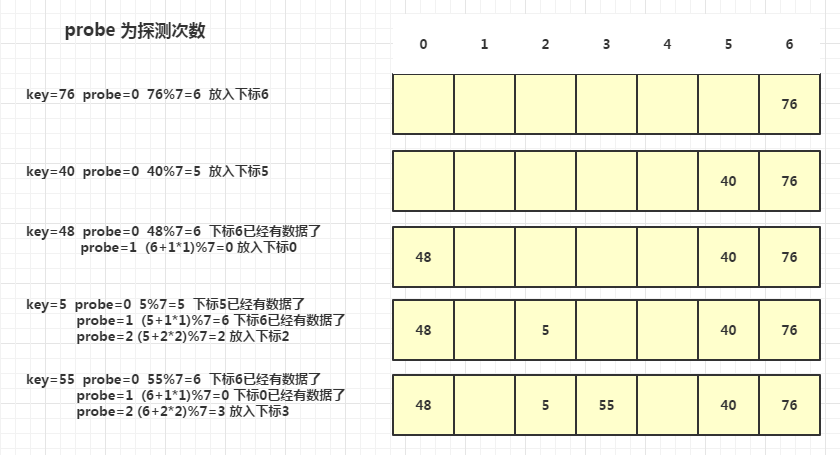
如果spark.shuffle.spill=true，那么SoftBuffer会在内存不足时将数据写入磁盘（ExternalAppendOnlyMap）

最后看看shuffle 读时是如何边fetch边运算，其实很简单

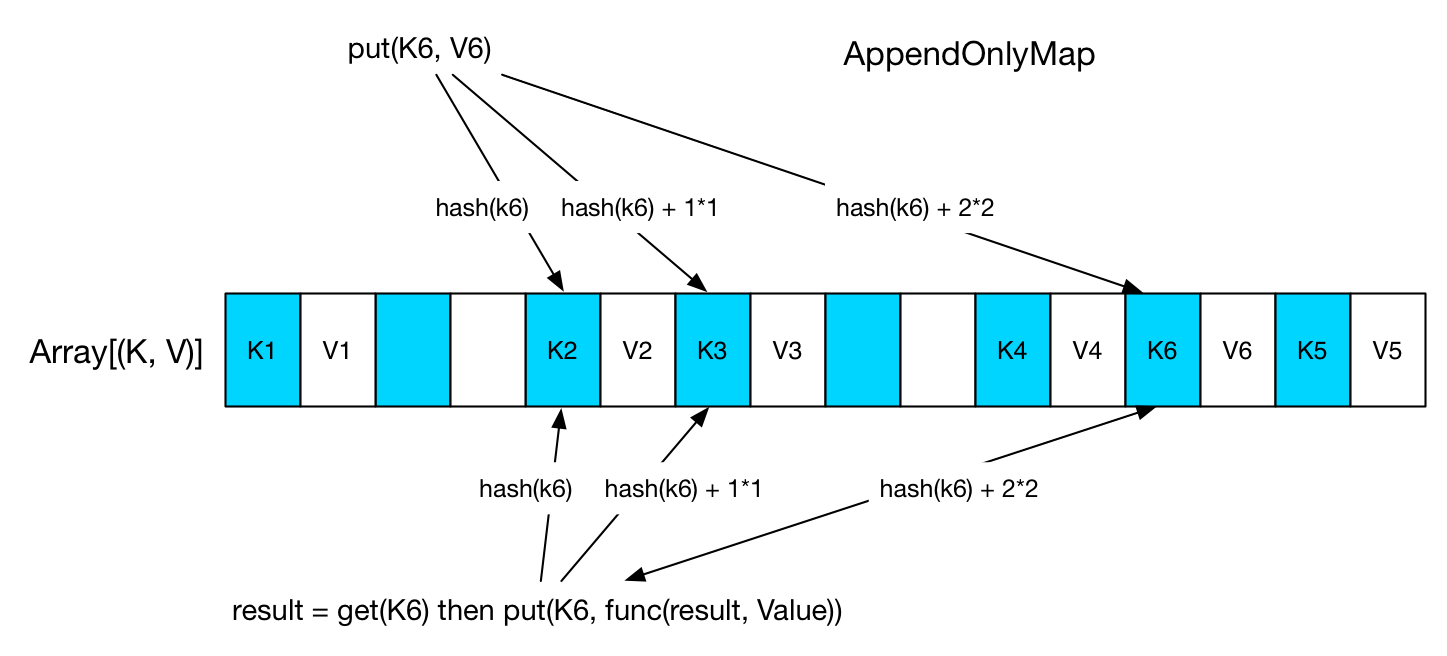
Fetch出的数据根据key执行AppendOnlyMap.get(key) 如果没有，就放入AppendOnlyMap.put(key, value)

如果该key已存在，则执行 AppendOnlyMap.put(key, 聚合函数(AppendOnlyMap.get(key))

这需要我们深入AppendOnlyMap 的内部实现：它是一个特殊的map实现，它采用了一种Quadratic Probing 的算法：



AppendOnlyMap 的实现与之类似不过它会将key和value邻接存储：



这个map的阈值是70%，当超过阈值时map的容量翻倍，当然所有key需要rehash

如果数据量比较大且涉及排序操作时，应当使用ExternalAppendOnlyMap（默认配置）