Spark会把数据都载入到内存么

2016-04-21 Hadoop技术博文

本文是面向Spark初学者，有Spark有比较深入的理解同学可以忽略。

前言

很多初学者其实对Spark的编程模式还是RDD这个概念理解不到位，就会产生一些误解。比如，很多时候我们常常以为一个文件是会被完整读入到内存，然后做各种变换，这很可能是受两个概念的误导：1、RDD的定义，RDD是一个分布式的不可变数据集合；2、Spark是一个内存处理引擎
如果你没有主动对RDD进行Cache/Persist等相关操作，它不过是一个概念上存在的虚拟数据集，你实际上是看不到这个RDD的数据的全集的(他不会真的都放到内存里)。

RDD的本质是什么

一个RDD 本质上是一个函数，而RDD的变换不过是函数的嵌套。RDD我认为有两类：
1、输入RDD,典型如KafkaRDD，JdbcRDD以及HadoopRDD等
2、转换RDD，如MapPartitionsRDD
我们以下面的代码为例做分析：

```
1 sc.textFile("abc.log").map().saveAsTextFile("")
```

textFile 中间会构建出一个HadoopRDD,然后返回了MapPartitionsRDD
map函数运行后会构建出一个MapPartitionsRDD
saveAsTextFile触发了实际流程代码的执行
所以RDD不过是对一个函数的封装，当一个函数对数据处理完成后，我们就得到一个RDD的数据集(是一个虚拟的，后续会解释)。
HadoopRDD是数据来源，每个parition负责获取数据，获得过程是通过iterator.next 获得一条一条记录的。假设某个时刻拿到了一条数据A,这个A会立刻被map里的函数处理得到B（完成了转换）,然后开始写入到HDFS上。其他数据重复如此。所以整个过程：

1、理论上某个MapPartitionsRDD里实际在内存里的数据等于其Partition的数目，是个非常小的数值。
2、HadoopRDD则会略多些，因为属于数据源，读取文件，假设读取文件的buffer是1M，那么最多也就是partitionNum*1M 数据在内存里
3、saveAsTextFile也是一样的，往HDFS写文件，需要buffer，最多数据量为 buffer* partitionNum
所以整个过程其实是流式的过程，一条数据被各个RDD所包裹的函数处理。
刚才我反复提到了嵌套函数，怎么知道它是嵌套的呢？
如果你写了这样一个代码：

```
1 sc.textFile("abc.log").map().map().....map().saveAsTextFile("")
```

有成千上万个map,很可能就堆栈溢出了。为啥?实际上是函数嵌套太深了。
按上面的逻辑，内存使用其实是非常小的，10G内存跑100T数据也不是难事。但是为什么Spark常常因为内存问题挂掉呢？ 我们接着往下看。

Shuffle的本质是什么

这就是为什么要分Stage了。每个Stage其实就是我上面说的那样，一套数据被N个嵌套的函数处理(也就是你的transform动作)。遇到了Shuffle,就被切开来，所谓的Shuffle，本质上是把数据按规则临时都落到磁盘上，相当于完成了一个saveAsTextFile的动作，不过是存本地磁盘。然后被切开的下一个Stage则以本地磁盘的这些数据作为数据源，重新走上面描述的流程。
我们再做一次描述：
所谓Shuffle不过是把处理流程切分，给切分的上一段(我们称为Stage M)加个存储到磁盘的Action动作，把切分的下一段(Stage M+1)数据源变成Stage M存储的磁盘文件。每个Stage都可以走我上面的描述，让每条数据都可以被N个嵌套的函数处理，最后通过用户指定的动作进行存储。

为什么Shuffle容易导致Spark挂掉

前面我们提到，Shuffle不过是偷偷的帮你加上了个类似saveAsLocalDiskFile的动作。然而，写磁盘是一个高昂的动作。所以我们尽可能的把数据先放到内存，再批量写到文件里，还有读磁盘文件也是给费内存的动作。把数据放内存，就遇到个问题，比如10000条数据，到底会占用多少内存？这个其实很难预估的。所以一不小心，就容易导致内存溢出了。这其实也是一个很无奈的事情。

我们做Cache/Persist意味着什么

其实就是给某个Stage加上了一个saveAsMemoryBlockFile的动作，然后下次再要数据的时候，就不用算了。这些存在内存的数据就表示了某个RDD处理后的结果。这个才是说为啥Spark是内存计算引擎的地方。在MR里，你是要放到HDFS里的，但Spark允许你把中间结果放内存里。

原文：http://www.jianshu.com/p/b70fe63a77a8