**目标：**

1. **数据读入RDD时间**
2. **RDD落盘时间，序列化/反序列化时间**
3. **GC时间**

**获取数据读入RDD时间**

获取第1和第2项目前有两种方案，一种是使用Java Instrumentation对字节码做修改，在指定方法前插入监听函数，进行计时；另一种是直接修改源码，插入相应代码，将所需时间戳输出到日志中。

两种方案各有优缺，前一种方案有更好的移植性，只需要打包出agent jar包，就能在各个环境下使用，只需要添加javaagent参数，可以在Spark submit时指定，也可以直接在spark配置文件中指定；缺点是自由度较低，只能在指定方法前添加代码，而不能获取运行时对象的详细信息。第二种方案直接修改源码，可以获取更多的信息，但是每次修改需要重新编译整个模块。

下面是一个javaagent的示例，获取一个wordcount程序读入RDD所花费的时间。

**val** file = sc.textFile("file:///home/shenyu/Documents/README.md", 6)  
 file.flatMap(\_.split(" ")).map(word => (word, 1)).reduceByKey(\_+\_).collect()

从hdfs或从本地读取文件都调用到了hadoopFile方法来初始化一个HadoopRDD，在action操作collect触发job后，将从后至前遍历到该HadoopRDD，进行compute操作，从文件中进行读取。因此我在compute前后加上监听代码，获取每个partition读入文件所花时间。

ctmethod.insertBefore("System.out.println(\"HadoopRDD Start read at \"+System.currentTimeMillis());");  
ctmethod.insertAfter("System.out.println(\"HadoopRDD end read at\"+System.currentTimeMillis());");

提交wordcount程序，能够看到输出数量，因为在读入的时候设置了partition数为６，所以共有６个输出。

HadoopRDD Start read at 1528208070965

HadoopRDD Start read at 1528208070965

HadoopRDD Start read at 1528208070965

HadoopRDD end read at1528208070980

HadoopRDD end read at1528208070980

HadoopRDD end read at1528208070980

HadoopRDD Start read at 1528208071742

HadoopRDD end read at1528208071743

HadoopRDD Start read at 1528208071746

HadoopRDD end read at1528208071756

HadoopRDD Start read at 1528208071791

HadoopRDD end read at1528208071793

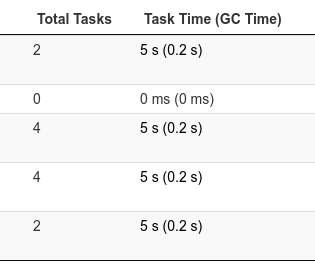
对start与end时间戳进行加减操作，就能得到总共花费的时间与平均时间。总花费的时间为58ms，平均每个partition读入需要10ms。

该方法缺点在于不能获取每个partition的id，也就不能细化地获取每个partition读取文件用时。如果直接修改源码，能获取更详细的信息。

**RDD落盘时间，序列化/反序列化时间的获取应该需要在BlockManager中插入监听代码，具体实践还没有完成。**

**获取GC时长有两个方案**

第一个是直接在history server中查看，能查到每个task gc时长。



如果只需要获取gc时间占运行时长的比例，该方法已经够用，如果需要获取更详细的gc信息，就需要用第二种方法，使用Spark额外参数比如XX:+PrintGCDetails -XX:+PrintGCTimeStamps打出每次gc的详情，再通过gcviewer等工具分析gc，可以得到minor gc，major gc各自的用时。