默认情况下，如果某个RDD没有进行持久化，但却设置了checkpoint，那么就悲剧了，本来这个job都执行完成了，但因为该rdd没有进行持久化，那么checkpoint job在进行checkpoint的时候，需要从RDD之前的所有RDD重新进行计算一次，然后将计算后的数据保存到checkpoint的目录中，所以通常建议对要进行checkpoint的RDD，先调用persist(StorageLevel.DISK_ONLY),将需要进行checkpoint的rdd的数据直接持久化到磁盘中， 这样的话当checkpoint job运行的时候，只需要直接从磁盘中读取出RDD的数据，然后保存到checkpoint的目录中去即可，这样可以更高效的checkpoint

checkpoint和持久化的区别：持久化只是将其数据保存在blockManager中，其lineage不变，且持久化的数据更容易丢失
checkpoint是斩断其lineage的，且因为checkpoint的数据通常保存于具有容错性的文件系统中，比如HDFS，所以checkpoint的容错性也较高

对RDD调用checkpoint()方法之后，它就接受RDDCcheckPointData对象的管理

RDDCheckPointData对象，会负责将调用了checkpoint()方法的RDD的状态设置为MarkedForCheckpoint

被标记为MarkedForCheckpoint的RDD所在的Job运行完之后，会调用job中最后一个RDD的doCheckpoint方法，该方法会沿着finalRDD的lineage向上查找标记为MarkedForCheckpoint的RDD，并将其状态改为CheckpointingInProgress

启动一个新的job，将lineage中标记为CheckPointingInProgress的RDD进行checkpoint操作，也就是将其数据写入到sparkContext.setCheckPointDir()所设置的文件系统中

进行checkpoint之后，会改变其lineage，清除掉其所依赖的RDD，并强行将其父RDD设置为checkpointRDD,RDD状态改为checkpointed

RDD进行checkpoint之后，后续在某个task中又调用该RDD的iterator方法的时候，就实现了高容错性，即使该RDD的持久化的数据丢失，也可以通过从RDD的父RDD-checkpointRDD中读取HDFS的数据