**[Spark——SparkContext简单分析](http://www.cnblogs.com/softlin/p/5792126.html)**

本篇文章就要根据源码分析**SparkContext**所做的一些事情，用过Spark的开发者都知道SparkContext是编写Spark程序用到的第一个类，足以说明SparkContext的重要性；这里先摘抄SparkContext源码注释来 简单介绍介绍SparkContext，注释的第一句话就是说**SparkContext为Spark的主要入口点**，简明扼要，如把Spark集群当作服务端那Spark Driver就是客户端，SparkContext则是客户端的核心；如注释所说 SparkContext用于**连接Spark集群、创建RDD、累加器（accumlator）、广播变量（broadcast variables）**，所以说SparkContext为Spark程序的根本都不为过，这里使用的Spark版本为2.0.1；



　　图片来自Spark官网，可以看到SparkContext处于DriverProgram核心位置，所有与Cluster、Worker Node交互的操作都需要SparkContext来完成；

## **SparkContext相关组件**

**1、SparkConf**  
　　SparkConf为Spark配置类，配置已键值对形式存储，封装了一个ConcurrentHashMap类实例settings用于存储Spark的配置信息；配置项包括：master、appName、Jars、ExecutorEnv等等；  
　　**2、SparkEnv**  
　　SparkEnv可以说是Context中非常重要的类，它维护着Spark的执行环境，包含有：serializer、RpcEnv、block Manager、map output tracker、etc等；所有的线程都可以通过SparkCotext访问到同一个SparkEnv对象；SparkContext通过SparkEnv.createDriverEnv创建SparkEnv实例；在SparkEnv中包含了如下主要对象：

　　SecurityManager：用于对权限、账号进行管理、Hadoop YARN模式下的证书管理等；  
　　RpcEnv：为Rpc环境的封装，之前使用的是Akka现在默认已经使用了Netty作为Spark的Rpc通信框架，Spark中有RpcEnvFactory trait特质默认实现为NettyRpcEnvFactory，在Factory中默认使用了Jdk的Serializer作为序列化工具；  
　　SerializerManager：用于管理Spark组件的压缩与序列化；  
　　BroadcastManager：用与管理广播对象，默认使用了TorrentBroadcastFactory广播工厂；  
　　MapOutputTracker：跟踪Map阶段结果的输出状态，用于在reduce阶段获取地址与输出结果，如果当前为Driver则创建MapOutputTrackerMaster对象否则创建的是MapOutputTrackerWorker两者都继承了MapOutputTracker类；  
　　ShuffleManager：用于管理远程和本地Block数据shuffle操作，默认使用了SortShuffleManager实例；  
　　MemoryManager：用于管理Spark的内存使用策略，有两种模式StaticMemoryManager、UnifiedMemoryManager，第一种为1.6版本之前的后面那张为1.6版本时引入的，当前模式使用第二种模式；两种模式区别为粗略解释为第一种是静态管理模式，而第二种为动态分配模式，execution与storage之间可以相互“借”内存；  
　　BlockTransferService：块传输服务，默认使用了Netty的实现，用于获取网络节点的Block或者上传当前结点的Block到网络节点；  
　　BlockManagerMaster：用于对Block的协调与管理；  
　　BlockManager：为Spark存储系统重要组成部分，用于管理Block；  
　　MetricsSystem：Spark测量系统；

**3、LiveListenerBus**  
　　异步传递Spark事件监听与SparkListeners监听器的注册；  
　　**4、JobProgressListener**  
　　JobProgressListener监听器用于监听Spark中任务的进度信息，SparkUI上的任务数据既是该监听器提供的，监听的事件包括有，Job：active、completed、failed；Stage：pending、active、completed、skipped、failed等；JobProgressListener最终将注册到LiveListenerBus中；

**5、SparkUI**  
　　SparkUI为Spark监控Web平台提供了Spark环境、任务的整个生命周期的监控；

**6、TaskScheduler**  
　　TaskScheduler为Spark的任务调度器，Spark通过他提交任务并且请求集群调度任务；TaskScheduler通过Master匹配部署模式用于创建TashSchedulerImpl与根据不同的集群管理模式（local、local[n]、standalone、local-cluster、mesos、YARN）创建不同的SchedulerBackend实例；

**7、DAGScheduler**  
　　DAGScheduler为高级的、基于stage的调度器，为提交给它的job计算stage，将stage作为tasksets提交给底层调度器TaskScheduler执行；DAGScheduler还会决定着stage的最优运行位置；  
　　**8、ExecutorAllocationManager**  
　　根据负载动态的分配与删除Executor，可通过ExecutorAllcationManager设置动态分配最小Executor、最大Executor、初始Executor数量等配置，调用start方法时会将ExecutorAllocationListener加入到LiveListenerBus中监听Executor的添加、移除等；  
　　**9、ContextClearner**  
　　ContextClearner为RDD、shuffle、broadcast状态的异步清理器，清理超出应用范围的RDD、ShuffleDependency、Broadcast对象；清理操作由ContextClearner启动的守护线程执行；  
　　**10、SparkStatusTracker**  
　　低级别的状态报告API，对job、stage的状态进行监控；包含有一个jobProgressListener监听器，用于获取监控到的job、stage事件信息、Executor信息；  
　　**11、HadoopConfiguration**  
　　Spark默认使用HDFS来作为分布式文件系统，HadoopConfigguration用于获取Hadoop配置信息，通过SparkHadoopUtil.get.newConfiguration创建Configuration对象，SparkHadoopUtil 会根据SPARK\_YARN\_MODE配置来判断是用SparkHadoopUtil或是YarnSparkHadoopUtil，创建该对象时会将spark.hadoop.开头配置都复制到HadoopConfugration中；

## **简单总结**

　　以上的对象为SparkContext使用到的主要对象，可以看到SparkContext包含了Spark程序用到的几乎所有核心对象可见SparkContext的重要性；创建SparkContext时会添加一个钩子到ShutdownHookManager中用于在Spark程序关闭时对上述对象进行清理，在创建RDD等操作也会判断SparkContext是否已stop；  
　　通常情况下一个Driver只会有一个SparkContext实例，但可通过spark.driver.allowMultipleContexts配置来允许driver中存在多个SparkContext实例；

**参考资料：**  
<http://spark.apache.org/docs/latest/>