**学习总结和收获**

这一周的重点是Spark RDD的基本操作和Spark SQL,DataFrame,了解了一下Spark GraphX，为下周的学习做准备。

Spark RDD 的基本操作有两种，一种是Transformation和Action,Transformation操作主要是将一个RDD转换成为一个新的RDD,从新的RDD中找出我们需要的结果。Action就是执行的操作。

为什么RDD高效呢？总结一下有三点：

1.RDD数据只读，不可修改。如果需要修改数据，必须从父RDD转换（transformation）到子RDD。所以，在容错策略中，RDD没有数据冗余，而是通过RDD父子依赖（血缘）关系进行重算实现容错。

2.多个RDD操作之间，数据不用落地到磁盘上，避免不必要的I/O操作。

3.RDD中存放的数据可以是java对象，所以避免的不必要的对象序列化和反序列化。

总而言之，RDD高效的主要因素是尽量避免不必要的操作和牺牲数据的操作精度，用来提高计算效率。

Spark SQL的学习主要是利用sql语句进行查询，得到我们需要的结果，最重要的就是要熟悉SQL语句的。Spark SQL的核心是把已有的RDD，带上Schema信息，然后注册成类似sql里的”Table”，对其进行sql查询。这里面主要分两部分，一是生成SchemaRD，二是执行查询。对于Spark SQL来说，数据方面，RDD可以来自任何已有的RDD，也可以来自支持的第三方格式，如json file、parquet file。SQLContext下会把带case class的RDD隐式转化为SchemaRDD。Spark SQL 中所有相关功能的入口点是 [SQLContext](http://spark.apache.org/docs/latest/api/scala/index.html" \l "org.apache.spark.sql.SQLContext) 类或者它的子类， 创建一个 SQLContext 的所有需要仅仅是一个 SparkContext。得到一个SQLContext的实例后，你可以调用里面的sql方法进行SQL语句查询。

在Spark1.3.0之后，增加了DataFrame的API。DataFrame让Spark具备了处理大规模结构化数据的能力，在比原有的RDD转化方式易用的前提下，计算性能更还快了两倍。DataFrame中封装了许多方法方便了我们查询和过滤数据

Spark SQL 支持两种方法将存在的 RDD 转换为 DataFrame 。

第一种方法使用反射来推断包含特定对象类型的 RDD 的模式。在你写 spark 程序的同时，当你已经知道了模式，这种基于反射的方法可以使代码更简洁并且程序工作得更好.

第二种方法是通过一个编程接口来实现，这个接口允许你构造一个模式，然后在存在的 RDD 上使用它。虽然这种方法更冗长，但是它允许你在运行期之前不知道列以及列的类型的情况下构造 DataFrame。

对于SparkGraphX,了解了怎么样构建出一张有向多重图和一些基本的方法。还有待进一步学习。