Spark序列化和压缩

在分布式计算中，序列化和压缩是提升性能的两个重要的性能。Spark序列化将链式分布的数据转化为连续分布的数据，这些就能够进行分布式的进程间数据通信或者在内存进行数据压缩等操作，通过压缩能够减少内存占用以及IO和网络数据传输开销，提供Spark整体的应用性能。

# 序列化

在Spark中内置了两个数据序列化类：JavaSerializer和KryoSerializer，这两个继承于抽象类Serializer，而在Spark SQL中SparkSqlSerializer继承于KryoSerializer，之间的关系如下所示：



默认情况下Spark默认使用的JavaSerializer序列方法，使用的是Java的ObjectOutputStream序列化框架。JavaSerializer继承于java.io.Serializable，性能不佳，但是生成的序列结果比较大，因此Spark提供性能更佳、压缩效率更高的KryoSerializer方法。

Spark初始序列化是在SparkEnv类进行创建，在该类中根据spark.serializer配置初始化序列化实例，然后把该实例作为参数初始化SerializerManager实例，而SerializerManager作为参数初始化BlockManager，代码如下所示：

*private SerializerManager serializerManager = new SerializerManager(  
 new JavaSerializer(new SparkConf()),  
 new SparkConf().set("spark.shuffle.spill.compress", "false"));*

这里可配的序列化的对象是Shuffle数据以及RDD缓存等场合，对于Spark任务的序列化是通过spark.closure.serializer来配置，目前只支持JavaSerializer。

# 压缩

Spark内置提供了三种压缩方法，分别是：LZ4、LZF和Snappy，这三个方法均继承CompressionCodec，并实现了其压缩和解压的两个方法，Spark压缩类图如下所示：



这三个压缩算法采用了第三方库实现的，Snappy提供了更高的压缩速度，LZF提供了更高的压缩比，LZ4提供了压缩速度和压缩比俱佳的性能：

* Snappy，使用了org.xerial.snappy库，Snappy算法的前身是Zippy，被Google用于MapReduce和BigTable等许多项目
* LZF，使用了com.ning.compress.lzf库，其中Ning-compress是一个队数据进行LZF格式压缩和解压缩的库
* LZ4，使用了net.jpountz.lz4库，高效的无损压缩算法