1. Shuffle含义

Shuffle的中文意思“洗牌”，Shuffle可以将具有某种共同特征的一类数据需要最终汇聚(aggregate)到一个计算节点上进行计算.

1. Shuffle过程面临的问题
2. 数据量非常大，TB或者PB的数据分散到几百甚至数千，数万台机器上。
3. 为了将这个数据汇聚到正确的节点，需要将这些数据放入正确的partition，因为数据大小已经大于节点的内存，因此这个过程可能会发生多次硬盘续写。
4. 数据在网络传输的时候，为了节省带宽，需要压缩和解压缩数据，如何在这个过程中比较好的选择？
5. 数据在网络传输，序列化和反序列化也变得相对复杂。
6. Shuffle的结果一定要持久化，因为如果出错，计算所以依赖的所有的RDD全部要重算。

**深入分析每个Shuffle Map Task结束时，数据是如何持久化(即Shuffle Write)以使得下游的Task可以获取到其需要处理的数据的(即Shuffle Read)？Shuffle会将数据持久化到本地文件系统**

1. Hash Based Shuffle

在实际应用场景中有些事不需要排序，因此多余的排序只能使性能变差。而hadoop中的Map Reduce是需要排序的。

每个Shuffle Map Task根据Key的哈希值，计算出每个Key需要写入的Partition然后将数据单独写入一个文件，这个Partition实际上就对应了下游的一个Shuffle Map Task或者Result Task。因此下游的Task在计算时会通过网络(如果该Task与上游的Shuffle Map Task运行在同一个节点上，那么此时就是一个本地的硬盘读写)读取这个文件并进行计算。

4.1 Basic Shuffle Writer实现解析

在Exexutor上执行Shuffle Map Task时，最终会调用org.apache.spark.scheduler.ShuffleMapTask的runTask。核心代码如下：

执行逻辑如下：

1. 从SparkEnv中获取shuffleManager，现在Spark除了支持Hash和Sort Based Shuffle外，还支持external 的Shuffle Service用户自定义的Shuffle。
2. manager.getWriter就得到了Writer，此时的Writer是默认的HashShuffleWriter.
3. Writer.write运行rdd开始计算，运算结果通过Writer进行持久化。

首先，通过ShuffleDependency是否定义了aggregator来确定是否需要做Map端的聚合。

其次,将原始结果或者聚合后的结果通过org.apache.spark.shuffle. FileShuffleBlockResolver.forMapTask的方法写入。

最后，写入完成之后，将元数据信息写入org.apache.spark.scheduler.MapStatus,然后下游的Task可以通过这个MapStatus取得需要处理的数据。

/\*\* Write a bunch of records to this task's output \*/

override def write(records: Iterator[Product2[K, V]]): Unit = {

val iter = if (dep.aggregator.isDefined) { // aggregator是否被定义

if (dep.mapSideCombine) { //判断是否需要聚合，需要则聚合records

dep.aggregator.get.combineValuesByKey(records, context)

} else {

records

}

} else {

require(!dep.mapSideCombine, "Map-side combine without Aggregator specified!")

records

}

for (elem <- iter) {

//获得该elem需要写入的partition

val bucketId = dep.partitioner.getPartition(elem.\_1)

//根据bucketId来写elem

shuffle.writers(bucketId).write(elem.\_1, elem.\_2)

}

}

注意：根据并行度