如何合理设置Spark并行度

1、在Spark中，什么是并行度？

在Spark作业中，各个stage的task的数量，也就代表了Spark作业在各个阶段stage的并行度。

在Spark任务运行过程中，资源调度器根据程序的需要分配资源，当分配完所能分配的最大资源以后，对应资源去调节程序的并行度，如果并行度没有与资源相匹配，将导致分配的资源浪费掉。并行运行可以让每个task要处理的数量变少（原理：合理设置并行度，可以充分利用集群资源，减少每个task处理数据量，而增加性能加快运行速度）

举例：

假如，现在已经在spark-submit 脚本里面，给spark作业分配了足够多的资源，比如50个executor，每个executor 有10G内存，每个executor有3个cpu core。task没有设置，或者设置的很少，比如就设置了，100个task。50个executor，每个executor 有3个core ，也就是说，Application 任何一个stage运行的时候，都有总数150个cpu core可以并行运行。但是，现在只有100个task，平均分配一下，每个executor 分配到2个task，那么同时在运行的task，只有100个task，每个executor 只会并行运行 2个task。每个executor 剩下的一个cpu core 就浪费掉了！这就是虽然资源分配充足了，但是并行度没有与资源相匹配，导致分配下去的资源都浪费掉了。合理的并行度的设置应该要设置的足够大，大到可以完全合理的利用你的集群资源； 比如上面的例子，总共集群有150个cpu core ，可以并行运行150个task。那就应该把Application 的并行度至少设置成150个，才能完全有效的利用集群的资源，让150个task 并行执行，而且task增加到150个以后，即可以同时并行运行，还可以让每个task要处理的数量变少； 比如总共 150G 的数据要处理， 如果是100个task，每个task 要计算1.5G的数据。现在增加到150个task，每个task只要处理1G数据。

2、如何去提高并行度？

task数量至少设置成与spark Application 的总cpu core 数量相同（最理性情况，150个core，分配150个task，一起运行，差不多同一时间运行完毕）。官方推荐，task数量设置成spark Application 总cpu core数量的2~3倍 ，比如150个cpu core，基本设置 task数量为 300~ 500. 与理想情况不同的是有些task 会运行快一点，比如50s 就完了，有些task 可能会慢一点，要一分半才运行完，所以如果你的task数量，刚好设置的跟cpu core 数量相同，可能会导致资源的浪费，因为 比如150个task，10个先运行完了，剩余140个还在运行，但是这个时候，就有10个cpu core空闲出来了，导致浪费。如果设置2~3倍，那么一个task运行完以后，另外一个task马上补上来，尽量让cpu core不要空闲。同时尽量提升spark运行效率和速度，提升性能。

3、如何设置一个Spark Application的并行度？

* 在spark-submit中采用spark.defalut.parallelism设置， 默认是没有设置值的，如果设置了值比如说10，是在shuffle的过程才会起作用（val rdd2 = rdd1.reduceByKey(\_+\_) //rdd2的分区数就是10，rdd1的分区数不受这个参数的影响）
* 在程序中采用new SparkConf().set(“spark.defalut.parallelism”,”10“)的方式设置
* 如果读取的数据在HDFS上，增加block数，默认情况下split与block是一对一的，而split又与RDD中的partition对应，所以增加了block数，也就提高了并行度。
  + RDD.repartition，给RDD重新设置partition的数量
  + reduceByKey的算子指定partition的数量

val rdd2 = rdd1.reduceByKey(\_+\_,10)

val rdd3 = rdd2.map.filter.reduceByKey(\_+\_)

* + join操作partition的数量。如val rdd3 = rdd1.join（rdd2）//rdd3里面partiiton的数量是由父RDD中最多的partition数量来决定，因此使用join算子的时候，增加父RDD中partition的数量。
  + spark sql中shuffle过程中partitions的数量采用spark.sql.shuffle.partitions指定

参考：

https://www.cnblogs.com/double-kill/p/9012383.html