Spark共享变量

通常情况下，当一个函数传递给远程集群节点上运行的Spark操作时（如Map，Reduce），该函数中所有的变量都会在各节点上创建副本，在各节点中的变量相互隔离并由所在节点的函数进行调用，并且这些变量的更新都不会传递回Driver程序。在任务间进行通用、可读写的共享变量是低效的，然后Spark还是提供了两种类型的共享变量：广播变量和累加器。

# 广播变量

广播变量允许开发人员在每个节点缓存只读的变量，而不是在任务之间传递这些变量。例如，使用广播变量能够高效地在每个节点创建大数据集的副本。同时，Spark还使用高效广播算法分发这些变量，从而减少通信的开销。

Spark应用程序作业的执行由一系列调度阶段构成，这些调度阶段通过Shuffle进行分离。Spark能够在每个调度阶段自动广播任务所需通用的数据，这些数据在广播时需要进行序列化缓存，并且在任务运行前进行反序列化。这就意味着当多个调度阶段的任务需要相同的数据，显式的创建广播变量才有用。

通过通过调用SparkContext.broadcast(v)创建一个广播变量v，该广播变量封装在v变量中，可使用获取该变量value的方法进行访问，代码如下所示：

*scala> val broadcastVar = sc.broadcast(Array(1,2,3))*

*broadcastVar: org.apache.spark.broadcast.Broadcast[Array[Int]] = Broadcast(0)*

*scala> broadcastVar.value*

*res0: Array[Int] = Array(1, 2, 3)*

当广播变量创建后，在集群中所有函数将以变量v代表该广播量，并且该变量v一次性分发到各节点上。另外，为了确保所有节点获得相同的变量，对象v广播后只读不能够被修改。

# 累加器

累加器是Spark中仅有通过关联操作进行累加的变量，因此能够有效的支持并行计算，他们能够用于计数（MapReduce）和求和。Spark原生支持数值类型的累加器，不过开发人员能够定义新的类型。如果在创建累加器时指定了名称，可以通过Spark的UI监控界面中进行查看，这种方式能够帮助理解作业所构成的调度阶段执行过程。

通过调用SparkContext.accumulator(v)方法初始化累加器变量v，在集群中的任务能够使用加法或者“+=”操作进行累加操作。然而它们不能在应用程序中读取这些值，只能由Driver程序通过读方法获取这些累加器的值。

下面的代码演示如何把一个数组的元素追加到累加器中：

*scala> val accum = sc.accumulator(0,"My Accumulator")*

*warning: there were 2 deprecation warning(s); re-run with -deprecation for details*

*accum: org.apache.spark.Accumulator[Int] = 0*

*scala> sc.parallelize(Array(1,2,3,4)).foreach(x => accum += x)*

*scala> accum.value*

*res2: Int = 10*

尽管上面的例子使用Spark原生所支持的累加器Int类型，但是开发人员能够通过继承AccumulatorParam类来创建自定义的累加类型。AccumulatorParam接口提供了两个方法：zero方法为自定义类型设置0和addInPlace方法将两个变量进行求和。