7. yarn调优

关于Yarn内存分配与管理，主要涉及到了ResourceManage、ApplicationMatser、NodeManager这几个概念，相关的优化也要紧紧围绕着这几方面来开展。这里还有一个Container的概念，现在可以先把它理解为运行map/reduce task的容器，后面有详细介绍。

7.1 RM的内存资源配置, 配置的是资源调度相关

RM1：yarn.scheduler.minimum-allocation-mb 分配给AM单个容器可申请的最小内存 RM2：yarn.scheduler.maximum-allocation-mb 分配给AM单个容器可申请的最大内存 注：

最小值可以计算一个节点最大Container数量 一旦设置，不可动态改变

7.2 NM的内存资源配置，配置的是硬件资源相关

NM1：yarn.nodemanager.resource.memory-mb 节点最大可用内存 NM2：yarn.nodemanager.vmem-pmem-ratio 虚拟内存率，默认2.1 注：

RM1、RM2的值均不能大于NM1的值 NM1可以计算节点最大最大Container数量，max(Container)=NM1/RM1 一旦设置，不可动态改变

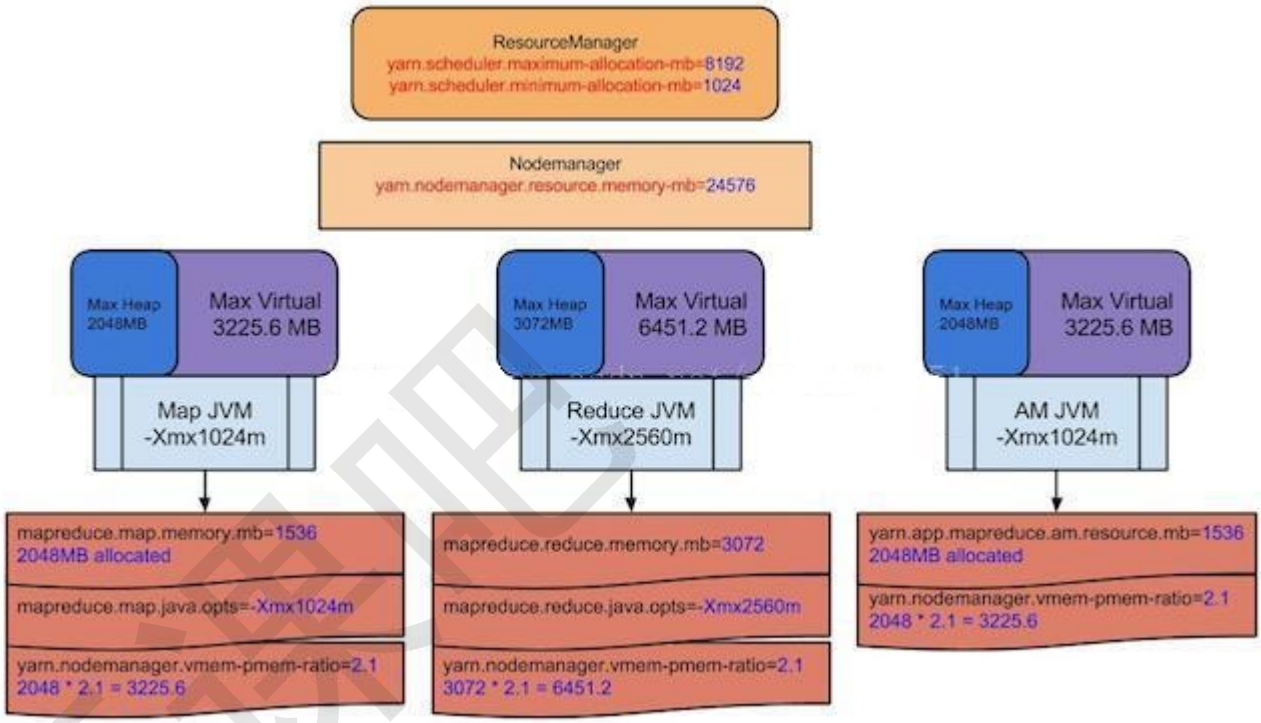
7.3 AM内存配置相关参数，配置的是任务相关

AM1：mapreduce.map.memory.mb 分配给map Container的内存大小 AM2：mapreduce.reduce.memory.mb 分配给reduce Container的内存大小

这两个值应该在RM1和RM2这两个值之间 AM2的值最好为AM1的两倍 这两个值可以在启动时改变

AM3：mapreduce.map.java.opts 运行map任务的jvm参数，如-Xmx，-Xms等选项 AM4：mapreduce.reduce.java.opts 运行reduce任务的jvm参数，如-Xmx，-Xms等选项 注：

这两个值应该在AM1和AM2之间



如上图所示，先看最下面褐色部分， AM参数mapreduce.map.memory.mb=1536MB，表示AM要为map Container申请1536MB资源，但RM实际分配的内存却是2048MB，因为yarn.scheduler.mininum-allocation-mb=1024MB，这定义了RM最小要分配1024MB，1536MB超过了这个值，所以实际分配给AM的值为2048MB(这涉及到了规整化因子，关于规整化因子，在本文最后有介绍)。 AM参数mapreduce.map.java.opts=-Xmx 1024m，表示运行map任务的jvm内存为1024MB,因为map任务要运行在Container里面，所以这个参数的值略微小于mapreduce.map.memory.mb=1536MB这个值。 NM参数yarn.nodemanager.vmem-pmem-radio=2.1,这表示NodeManager可以分配给map/reduce Container 2.1倍的虚拟内存，安照上面的配置，实际分配给map Container容器的虚拟内存大小为2048*2.1=3225.6MB，若实际用到的内存超过这个值，NM就会kill掉这个map Container,任务执行过程就会出现异常。 AM参数mapreduce.reduce.memory.mb=3072MB，表示分配给reduce Container的容器大小为3072MB,而map Container的大小分配的是1536MB，从这也看出，reduce Container容器的大小最好是map Container大小的两倍。 NM参数yarn.nodemanager.resource.mem.mb=24576MB,这个值表示节点分配给NodeManager的可用内存，也就是节点用来执行yarn任务的内存大小。这个值要根据实际服务器内存大小来配置，比如我们hadoop集群机器内存是128GB，我们可以分配其中的80%给yarn，也就是102GB。 上图中RM的两个参数分别1024MB和8192MB，分别表示分配给AM map/reduce Container的最大值和最小值。

7.4 关于Container

（1）Container是YARN中资源的抽象，它封装了某个节点上一定量的资源（CPU和内存两类资源）。它跟Linux Container没有任何关系，仅仅是YARN提出的一个概念（从实现上看，可看做一个可序列化/反序列化的Java类）。（2）Container由ApplicationMaster向ResourceManager申请的，由ResouceManager中的资源调度器异步分配给ApplicationMaster；（3）Container的运行是由ApplicationMaster向资源所在的NodeManager发起的，Container运行时需提供内部执行的任务命令（可以使任何命令，比如java、Python、C++进程启动命令均可）以及该命令执行所需的环境变量和外部资源（比如词典文件、可执行文件、jar包等）。 另外，一个应用程序所需的Container分为两大类，如下：（1）运行ApplicationMaster的Container：这是由ResourceManager（向内部的资源调度器）申请和启动的，用户提交应用程序时，可指定唯一的ApplicationMaster所需的资源；（2）运行各类任务的Container：这是由ApplicationMaster向ResourceManager申请的，并由ApplicationMaster与NodeManager通信以启动之。 以上两类Container可能在任意节点上，它们的位置通常而言是随机的，即

ApplicationMaster可能与它管理的任务运行在一个节点上。 Container是YARN中最重要的概念之一，懂得该概念对于理解YARN的资源模型至关重要，望大家好好理解。 注意：如下图，map/reduce task是运行在Container之中的，所以上面提到的mapreduce.map(reduce).memory.mb大小都大于mapreduce.map(reduce).java.opts值的大小。