## UDH YARN配置调优

目录

[UDH YARN配置调优 1](#_Toc423335713)

[一、内存配置 1](#_Toc423335714)

[二、 cpu配置 3](#_Toc423335715)

YARN作为一个资源调度器，应该考虑到集群里面每一台机子的计算资源，然后根据application申请的资源进行分配Container。Container是YARN里面资源分配的基本单位，具有一定的内存以及CPU资源。

在UDH YARN集群中，平衡内存、CPU、磁盘的资源的很重要的，根据经验，每两个container使用一块磁盘以及一个CPU核的时候可以使集群的资源得到一个比较好的利用。

### 一、内存配置

YARN以及MAPREDUCE所有可用的内存资源应该要除去系统运行需要的以及其他的hadoop的一些程序，总共保留的内存=系统内存+HBASE内存。

参考hortonwork公司的文档[Determine HDP Memory Configuration Settings](http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1.1/bk_installing_manually_book/content/rpm-chap1-11.html)来配置你的集群。

|  |  |  |
| --- | --- | --- |
| 每台机器内存 | 系统需要的内存 | Hbase需要的内存 |
| 4GB | 1GB | 1GB |
| 8GB | 2GB | 1GB |
| 16GB | 2GB | 2GB |
| 24GB | 4GB | 4GB |
| 48GB | 6GB | 8GB |
| 64GB | 8GB | 8GB |
| … |

计算每台机器最多可以拥有多少个container，可以使用下面的公式:

containers = min (2\*CORES, 1.8\*DISKS, (Total available RAM) / MIN\_CONTAINER\_SIZE)

* CORES为机器CPU核数
* DISKS为机器上挂载的磁盘个数
* Total available RAM为机器总内存
* MIN\_CONTAINER\_SIZE是指container最小的容量大小，

这需要根据具体情况去设置，可以参考下面的表格：

|  |  |
| --- | --- |
| 每台机器可用的RAM | Container最小值 |
| 小于4GB | 256MB |
| 4GB到8GB | 512MB |
| 8GB到24GB | 1024MB |
| 大于24GB | 2048MB |

每个container的平均使用内存大小计算方式为：

RAM-per-container = max(MIN\_CONTAINER\_SIZE, (Total Available RAM) / containers))

通过上面的计算，YARN以及MAPREDUCE可以这样配置：

|  |  |  |  |
| --- | --- | --- | --- |
| 配置文件 | 配置设置 | 默认值 | 计算值 |
| yarn-site.xml | yarn.nodemanager.resource.memory-mb | 8192 MB | = containers \* RAM-per-container |
| yarn-site.xml | yarn.scheduler.minimum-allocation-mb | 1024MB | =RAM-per-container |
| yarn-site.xml | yarn.scheduler.maximum-allocation-mb | 8192 MB | = containers \* RAM-per-container |
| yarn-site.xml | yarn.app.mapreduce.am.resource.mb | 1536 MB | = 2 \* RAM-per-container |
| yarn-site.xml | yarn.app.mapreduce.am.command-opts | -Xmx1024m | = 0.8 \* 2 \* RAM-per-container |
| mapred-site.xml | mapreduce.map.memory.mb | 1024 MB | =RAM-per-containr |
| mapred-site.xml | mapreduce.reduce.memory.mb | 1024 MB | = 2 \* RAM-per-container |
| mapred-site.xml | mapreduce.map.java.opts |  | = 0.8 \* RAM-per-container |
| mapred-site.xml | mapreduce.reduce.java.opts |  | = 0.8 \* 2 \* RAM-per-container |

举个例子：对于128G内存、32核CPU的机器，挂载了7个磁盘，根据上面的说明，系统保留内存为24G，不适应HBase情况下，系统剩余可用内存为104G，计算containers值如下：

containers = min (2\*32, 1.8\* 7 , (128-24)/2) = min (64, 12.6 , 51) = 13

计算RAM-per-container值如下：

RAM-per-container = max (2, (124-24)/13) = max (2, 8) = 8

这样的话，每个container内存为8G，似乎有点多，我更愿意根据集群使用情况任务将其调整为2G内存，则集群中下面的参数配置值如下：

|  |  |  |
| --- | --- | --- |
| 配置文件 | 配置设置 | 计算值 |
| yarn-site.xml | yarn.nodemanager.resource.memory-mb | = 52 \* 2 =104 G |
| yarn-site.xml | yarn.scheduler.minimum-allocation-mb | = 2G |
| yarn-site.xml | yarn.scheduler.maximum-allocation-mb | = 52 \* 2 = 104G |
| yarn-site.xml | yarn.app.mapreduce.am.resource.mb | = 2 \* 2=4G |
| yarn-site.xml | yarn.app.mapreduce.am.command-opts | = 0.8 \* 2 \* 2=3.2G |
| mapred-site.xml | mapreduce.map.memory.mb | = 2G |
| mapred-site.xml | mapreduce.reduce.memory.mb | = 2 \* 2=4G |
| mapred-site.xml | mapreduce.map.java.opts | = 0.8 \* 2=1.6G |
| mapred-site.xml | mapreduce.reduce.java.opts | = 0.8 \* 2 \* 2=3.2G |

另外，还有一下几个参数：

* yarn.nodemanager.vmem-pmem-ratio：任务每使用1MB物理内存，最多可使用虚拟内存量，默认是2.1。
* yarn.nodemanager.pmem-check-enabled：是否启动一个线程检查每个任务正使用的物理内存量，如果任务超出分配值，则直接将其杀掉，默认是true。
* yarn.nodemanager.vmem-pmem-ratio：是否启动一个线程检查每个任务正使用的虚拟内存量，如果任务超出分配值，则直接将其杀掉，默认是true。

第一个参数的意思是当一个map任务总共分配的物理内存为2G的时候，该任务的container最多内分配的堆内存为1.6G，可以分配的虚拟内存上限为2\*2.1=4.2G。另外，照这样算下去，每个节点上YARN可以启动的Map数为104/2=52个。

### 二、 cpu配置

YARN中目前的CPU被划分成虚拟CPU（CPU virtual Core），这里的虚拟CPU是YARN自己引入的概念，初衷是，考虑到不同节点的CPU性能可能不同，每个CPU具有的计算能力也是不一样的，比如某个物理CPU的计算能力可能是另外一个物理CPU的2倍，这时候，你可以通过为第一个物理CPU多配置几个虚拟CPU弥补这种差异。用户提交作业时，可以指定每个任务需要的虚拟CPU个数。

在YARN中，CPU相关配置参数如下：

* yarn.nodemanager.resource.cpu-vcores：表示该节点上YARN可使用的虚拟CPU个数，默认是8，注意，目前推荐将该值设值为与物理CPU核数数目相同。如果你的节点CPU核数不够8个，则需要调减小这个值，而YARN不会智能的探测节点的物理CPU总数。
* yarn.scheduler.minimum-allocation-vcores：单个任务可申请的最小虚拟CPU个数，默认是1，如果一个任务申请的CPU个数少于该数，则该对应的值改为这个数。
* yarn.scheduler.maximum-allocation-vcores：单个任务可申请的最多虚拟CPU个数，默认是32。

对于一个CPU核数较多的集群来说，上面的默认配置显然是不合适的，在我的测试集群中，4个节点每个机器CPU核数为31，留一个给操作系统，可以配置为：

 <property>

 <name>yarn.nodemanager.resource.cpu-vcores</name>

 <value>31</value>

 </property>

 <property>

 <name>yarn.scheduler.maximum-allocation-vcores</name>

 <value>124</value>

 </property>