美伊小公主的超级奶爸

美伊小公主的超级奶爸!

昵称：美伊小公主的超级奶爸
园龄：2年5个月
粉丝：0
关注：0
+加关注

< 2018年8月 >

日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论

YARN 内存参数终极详解

很多朋友在刚开始搭建和使用 YARN 集群的时候，很容易就被纷繁复杂的配置参数搞晕了：参数名称相近、新老命名掺杂、文档说明模糊 。特别是那几个关于内存的配置参数，即使看好几遍文档也不能完全弄懂含义不说，配置时一不小心就会张冠李戴，犯错误。

如果你同样遇到了上面的问题，没有关系，在这篇文章中，我就为大家梳理一下 YARN 的几个不易理解的内存配置参数，并结合源码阐述它们的作用和原理，让大家彻底清楚这些参数的含义。

一、YARN 的基本架构

介绍 YARN 框架的介绍文章网上随处都可以找到，我这里就不做详细阐述了。之前我的文章"YARN环境中应用程序JAR包冲突问题的分析及解决"中也对 YARN 的一些知识点做了总结，大家可以在TheFortyTwo 后台回复编号 0x0002 获得这篇文章的推送。下面附上一张 YARN 框架图，方便引入我们的后续内容：

我的标签

我的标签

hadoop(2)
mapreduce(2)
yarn(2)

随笔档案

2016年3月 (2)

阅读排行榜

1. YARN 内存参数终极详解(658)
2. YARN环境中应用程序JAR包冲突问题的分析及解决(389)

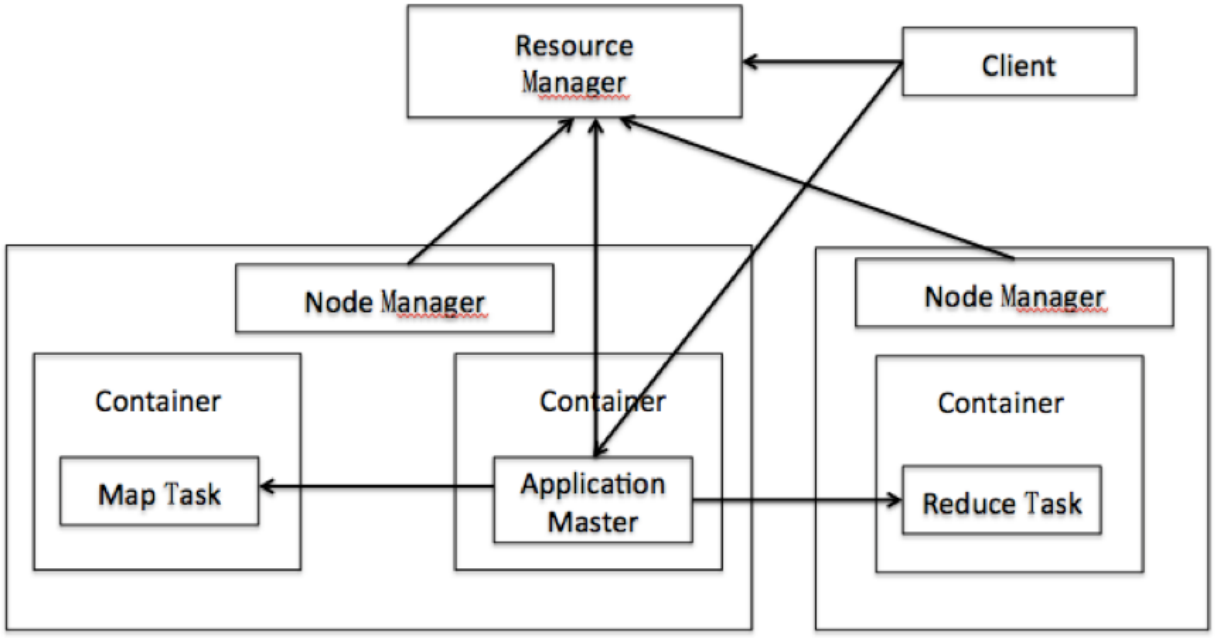

图 1：YARN 架构图

二、内存相关参数梳理

YARN 中关于内存配置的参数呢，乍一看有很多，其实主要也就是那么几个（如果你感觉实际接触到的比这更多更混乱，是因为大部分的配置参数都有新命名和旧命名，我后面会分别解释），我已经整理出来列在了下表中。大家先看一下，对于表中各列的意义，我会在本节后面详细说明；而对于每个参数的意义，我会放在下节进行详细解释。

配置对象	参数名称	旧参数名称	缺省值	所在配置文件	是否可在程序中覆盖设置
Map Task	mapreduce.map.java.opts	mapred.map.child.java.opts	-Xmx200m	mapred-default.xml	可以
	mapreduce.map.memor-mb	mapred.job.map.memory.mb	1024	mapred-default.xml	可以
Reduce Task	mapreduce.reduce.java.opts	mapred.reduce.child.java.opts	-Xmx200m	mapred-default.xml	可以
	mapred.job.reduce.memory.mb	mapreduce.reduce.memory.mb	1024	mapred-default.xml	可以
Map和Reduce Task	mapred.child.java.opts	无	无	mapred-default.xml	可以
NodeManager	yarn.nodemanager.resource.memory-mb	无	8192	yarn-default.xml	不行
	yarn.nodemanager.vmem-pmem-ratio	无	2.1	yarn-default.xml	不行
	yarn.nodemanager.vmem-check-enabled	无	true	yarn-default.xml	不行

图 2：内存参数整理图

下面我们解释一下表中的各列：

配置对象：指参数是针对何种组件起作用；

参数名称：这个不用解释，大家都明白；

旧参数名称：大家都知道，MapReduce 在大版本上，经历了 MR1 和 MR on YARN；而小版本则迭代了不计其数次。版本的演进过程中，开发人员发现很多参数的命名不够标准，就对参数名称做了修改；但是为了保证程序的前后兼容，仍然保留了旧参数名称的功能。这样等于是实现同一个功能的参数，就有了新旧两种不同的名称。比如 mapreduce.map.java.opts 和 mapred.map.child.java.opts 两个参数，其实是等价的。那如果新旧两个参数都设置了情况下，哪个参数会实际生效呢？Hadoop 的规则是，新参数设置了的话，会使用新参数，否则才会使用旧参数设置的值，而与你设置参数的顺序无关；

缺省值：如果没有设置参数的话，Hadoop 使用的默认值。需要注意的是，并非所有参数的默认值都是写在配置文件（如 mapred-default.xml）中的，比如 mapreduce.map.java.opts 这个参数，它的取值是在创建 Map Task 前，通过下面代码获得的：

```
if (isMapTask) {
userClasspath = jobConf.get("mapreduce.map.java.opts",
jobConf.get( "mapred.child.java.opts", "-Xmx200m"));
…
}
```

可以看到，这个参数的取值优先级是：

```
mapreduce.map.java.opts > mapred.child.java.opts > -Xmx200m
```

所在配置文件：指明了如果你想静态配置这个参数（而非在程序中调用 API 动态设置参数），应该在哪个配置文件中进行设置比较合适；

三、各参数终极解释

下面我们分别来讲解每个参数的功能和意义。

mapreduce.map.java.opts 和 mapreduce.map.memory.mb

我反复斟酌了一下，觉得这两个参数还是要放在一起讲才容易让大家理解，否则割裂开会让大家困惑更大。这两个参数的功能如下：

1. mapreduce.map.java.opts: 运行 Map 任务的 JVM 参数，例如 -Xmx 指定最大内存大小；
2. mapreduce.map.memory.mb: Container 这个进程的最大可用内存大小。

这两个参数是怎样一种联系呢？首先大家要了解 Container 是一个什么样的进程（想详细了解的话，就真的需要大家去看我的另一篇文章“YARN环境中应用程序JAR包冲突问题的分析及解决”，回复编号0x0002）。简单地说，Container 其实就是在执行一个脚本文件(launch_container.sh)，而脚本文件中，会执行一个 Java 的子进程，这个子进程就是真正的 Map Task。

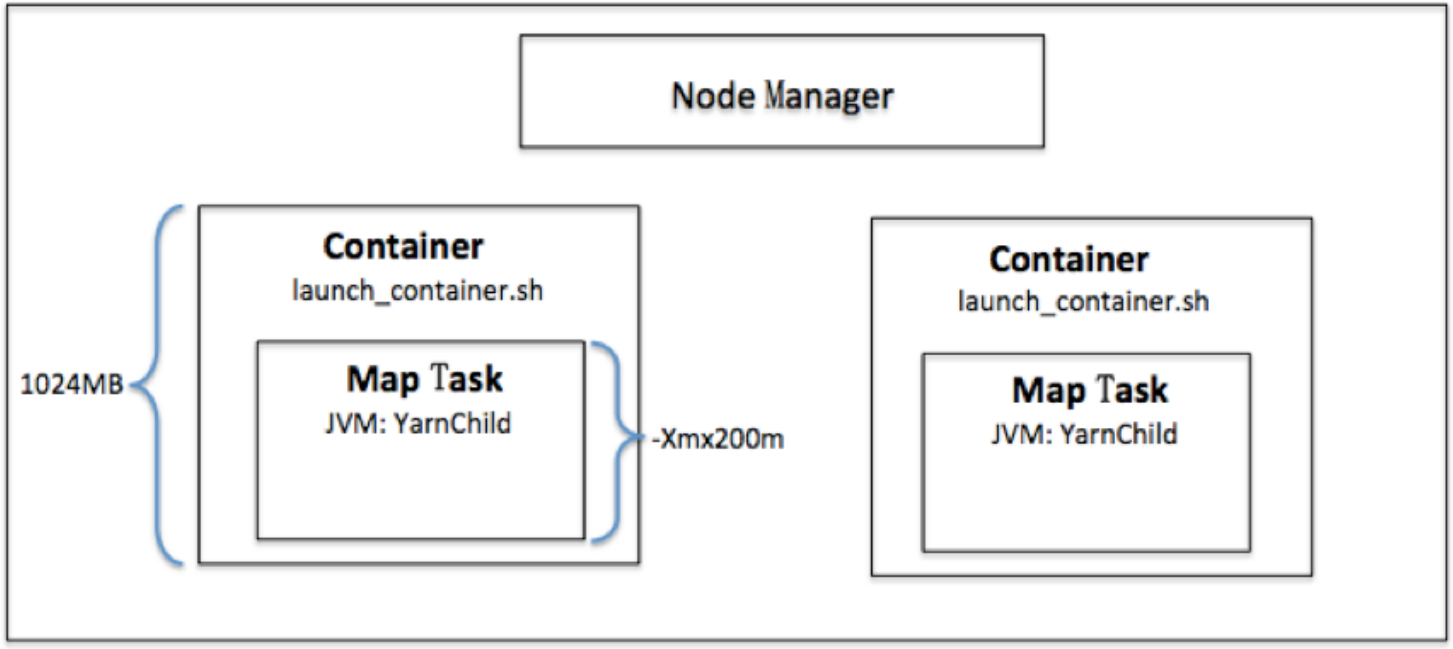

图 3: Container 和 Map Task 的关系图

理解了这一点大家就明白了，mapreduce.map.java.opts 其实就是启动 JVM 虚拟机时，传递给虚拟机的启动参数，而默认值 -Xmx200m 表示这个 Java 程序可以使用的最大堆内存数，一旦超过这个大小，JVM 就会抛出 Out of Memory 异常，并终止进程。而 mapreduce.map.memory.mb 设置的是 Container 的内存上限，这个参数由 NodeManager 读取并进行控制，当 Container 的内存大小超过了这个参数值，NodeManager 会负责 kill 掉 Container。在后面分析 yarn.nodemanager.vmem-pmem-ratio 这个参数的时候，会讲解 NodeManager 监控 Container 内存（包括虚拟内存和物理内存）及 kill 掉 Container 的过程。

紧接着，一些深入思考的读者可能就会提出这些问题了：

Q: 上面说过，Container 只是一个简单的脚本程序，且里面仅运行了一个 JVM 程序，那么为何还需要分别设置这两个参数，而不能简单的设置 JVM 的内存大小就是 Container的大小？

A: YARN 作为一个通用的计算平台，设计之初就考虑了各种语言的程序运行于这个平台之上，而非仅适用 Java 及 JVM。所以 Container 被设计成一个抽象的计算单元，于是它就有了自己的内存配置参数。

Q: JVM 是作为 Container 的独立子进程运行的，与 Container 是两个不同的进程。那么 JVM 使用的内存大小是否受限于 Container 的内存大小限制？也就是说，mapreduce.map.java.opts 参数值是否可以大于 mapreduce.map.memory.mb 的参数值？

A: 这就需要了解 NodeManager 是如何管理 Container 内存的了。NodeManager 专门有一个 monitor 线程，时刻监控所有 Container 的物理内存和虚拟内存的使用情况，看每个 Container 是否超过了其预设的内存大小。而计算 Container 内存大小的方式，是计算 Container 的所有子进程所用内存的和。上面说过了，JVM 是 Container 的子进程，那么 JVM 进程使用的内存大小，当然就算到了 Container 的使用内存量之中。一旦某个 Container 使用的内存量超过了其预设的内存量，则 NodeManager 就会无情地 kill 掉它。

mapreduce.reduce.java.opts 和 mapred.job.reduce.memory.mb

和上面介绍的参数类似，区别就是这两个参数是针对 Reducer 的。

mapred.child.java.opts

这个参数也已经是一个旧的参数了。在老版本的 MR 中，Map Task 和 Reduce Task 的 JVM 内存配置参数不是分开的，由这个参数统一指定。也就是说，这个参数其实已经分成了 mapreduce.map.java.opts 和 mapreduce.reduce.java.opts 两个，分别控制 Map Task 和 Reduce Task。但是为了前后兼容，这个参数在 Hadoop 源代码中仍然被使用，使用的地方上面章节已经讲述过了，这里再把优先级列一下：

```
mapreduce.map.java.opts > mapred.child.java.opts > -Xmx200m
```

yarn.nodemanager.resource.memory-mb

从这个参数开始，我们来看 NodeManager 的配置项。

这个参数其实是设置 NodeManager 预备从本机申请多少内存量的，用于所有 Container 的分配及计算。这个参数相当于一个阈值，限制了 NodeManager 能够使用的服务器的最大内存量，以防止 NodeManager 过度消耗系统内存，导致最终服务器宕机。这个值可以根据实际服务器的配置及使用，适度调整大小。例如我们的服务器是 96GB 的内存配置，上面部署了 NodeManager 和 HBase，我们为 NodeManager 分配了 52GB 的内存。

yarn.nodemanager.vmem-pmem-ratio 和 yarn.nodemanager.vmem-check-enabled

yarn.nodemanager.vmem-pmem-ratio 这个参数估计是最让人困惑的了。网上搜出的资料大都出自官方文档的解释，不够清晰明彻。下面我结合源代码和大家解释一下这个参数到底在控制什么。

首先，NodeManager 接收到 AppMaster 传递过来的 Container 后，会用 Container 的物理内存大小 (pmem) * yarn.nodemanager.vmem-pmem-ratio 得到 Container 的虚拟内存大小的限制，即为 vmemLimit：

```
long pmemBytes = container.getResource().getMemory() * 1024 * 1024L;

float pmemRatio =
container.daemonConf.getFloat(YarnConfiguration.NM_VMEM_PMEM_RATIO,
YarnConfiguration.DEFAULT_NM_VMEM_PMEM_RATIO);

long vmemBytes = (long) (pmemRatio * pmemBytes);
```

然后，NodeManager 在 monitor 线程中监控 Container 的 pmem（物理内存）和 vmem（虚拟内存）的使用情况。如果当前 vmem 大于 vmemLimit 的限制，或者 olderThanAge（与 JVM 内存分代相关）的内存大于限制，则 kill 掉进程：

```
if (currentMemUsage > (2 * vmemLimit)) {

isOverLimit = true;

} else if (curMemUsageOfAgedProcesses > vmemLimit) {

isOverLimit = true;

}
```

kill 进程的代码如下：

```
if (isMemoryOverLimit) {

// kill the container

eventDispatcher.getEventHandler().handle(new ContainerKillEvent(containerId,
msg));
```

```
}
```

上述控制是针对虚拟内存的，针对物理内存的使用 YARN 也有类似的监控，读者可以自行从源码中进行探索。yarn.nodemanager.vmem-check-enabled 参数则十分简单，就是上述监控的开关。

上面的介绍提到了 vmemLimit，也许大家会有个疑问：这里的 vmem 究竟是否是 OS 层面的虚拟内存概念呢？我们来看一下源码是怎么做的。

ContainerMontor 就是上述所说的 NodeManager 中监控每个 Container 内存使用情况的 monitor，它是一个独立线程。ContainerMonitor 获得单个 Container 内存（包括物理内存和虚拟内存）使用情况的逻辑如下：

Monitor 每隔 3 秒钟就更新一次每个 Container 的使用情况；更新的方式是：

1. 查看 /proc/pid/stat 目录下的所有文件，从中获得每个进程的所有信息；
2. 根据当前 Container 的 pid 找出其所有的子进程，并返回这个 Container 为根节点，子进程为叶节点的进程树；在 Linux 系统下，这个进程树保存在 ProcfsBasedProcessTree 类对象中；
3. 然后从 ProcfsBasedProcessTree 类对象中获得当前进程 (Container) 总虚拟内存量和物理内存量。

由此大家应该立马知道了，内存量是通过 /proc/pid/stat 文件获得的，且获得的是该进程及其所有子进程的内存量。所以，这里的 vmem 就是 OS 层面的虚拟内存概念。

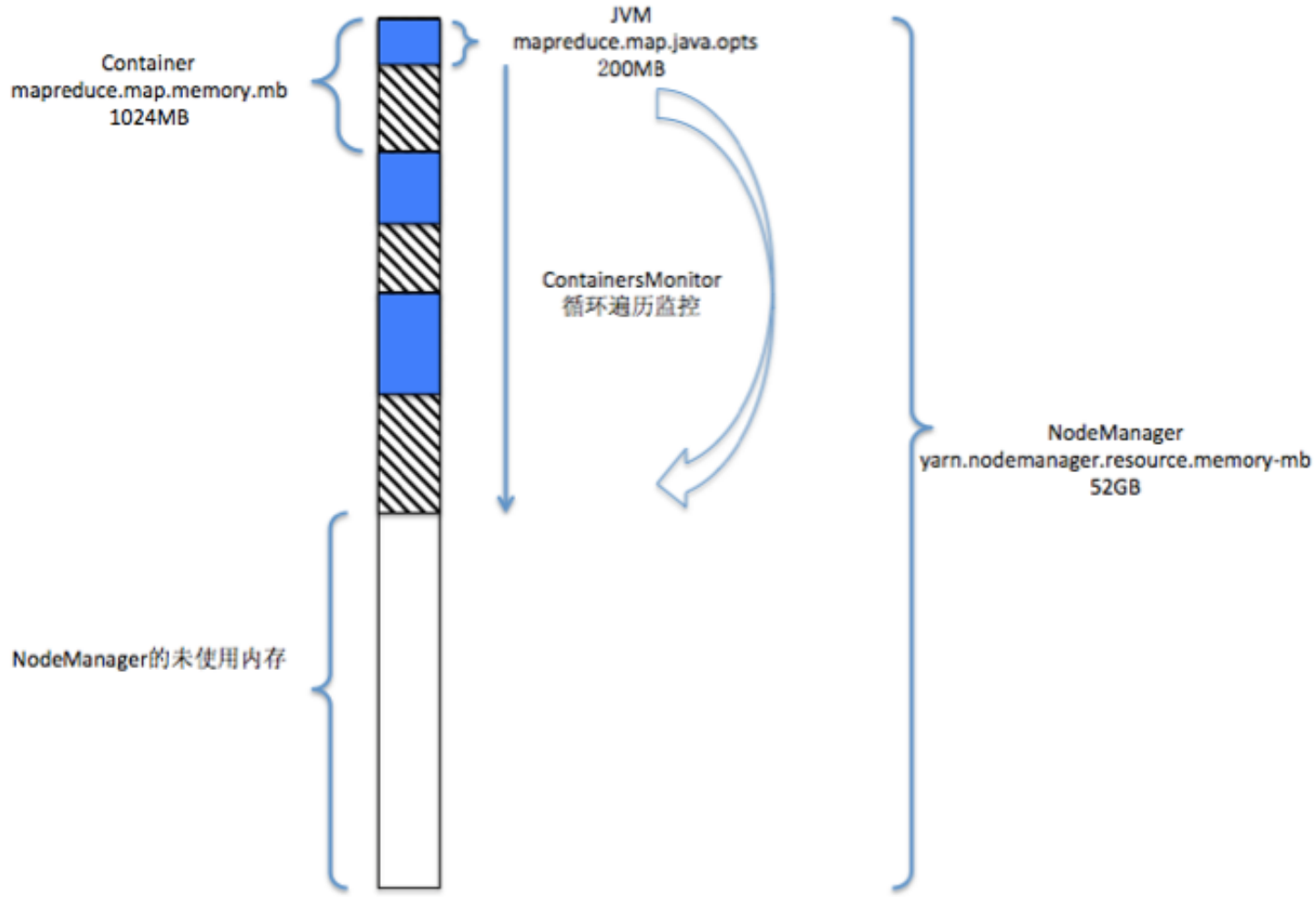

图 4：内存参数的组合示意图

四、结语

本文带大家深入剖析了 YARN 中几个容易混淆的内存参数，大家可以见微知著，从文章分析问题的角度找出同类问题的分析方法，文档与源码相结合，更深入了解隐藏在框架之下的秘密。

标签: hadoop, yarn, mapreduce

好文要顶 关注我 收藏该文

美伊小公主的超级奶爸
关注 - 0
粉丝 - 0
+加关注

0 0

» 下一篇：YARN环境中应用程序JAR包冲突问题的分析及解决

posted on 2016-03-03 13:24 美伊小公主的超级奶爸 阅读(659) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部