[译]Java8官方GC调优指南 --(二) 工效学 - 掘金

juejin.cn/post/6844904053432909832

2020年1月27日

> 本套文章是Java8官方GC调优指南的全文翻译，点击查看原文,原文章名称《Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide 》

2 Ergonomics 工效学

前言

Ergonomics是指JVM和垃圾回收调优的过程。JVM提供平台决定的默认垃圾回收器、heap size、和运行时编译器。其中行为决定的调整是根据Java程序的行为来动态进行的。

这一节描述了这些默认回收器和基于行为的调优。先了解这些默认配置再去调整细节。

Garbage Collector, Heap, and Runtime Compiler Default Selections

服务器级别的机器被定义为如下配置：

- 2或2个以上的物理CPU核数
- 2或2GB以上的物理内存

在服务器级别的物理机上，有下面几个默认配置：

- 吞吐量优先的垃圾回收器
- 初始化堆内存空间是物理机物理内存的1/64，最大1GB
- 最大堆内存空间是物理机内存的1/4，最大1GB
- Server runtime compiler

关于64位系统的初始堆和最大堆的细节，可以查看The Parallel Collector的Default Heap Size小节。

除了32位Windows操作系统之外，其他的所有平台都有服务器级的JVM定义。 下表表示了不同平台的不同的runtime compiler。

Platform	Operating System	Default	Default if Server-Class
i586	Linux	Client	Server
i586	Windows	Client	Client
SPARC(64-bit)	Solaris	Server	Server
AMD(64-bit)	Linux	Server	Server
AMD(64-bit)	Windows	Server	Server

Behavior-Based Tuning

对于Parallel Collector，Java SE提供了两个回收器参数来达到想要的效果：

- 最大停顿时间目标
- 应用吞吐量目标

具体可以看The Parallel Colletor章节。(其他的回收器没有这两个参数)。注意这两个目标不一定总能同时满足。 应用需要一个足够大的堆，能够承载所有存活对象。另外，减小堆的大小可能会达不到调优的效果

Maximum Pause Time Goal 最大停顿时间优先

暂停时间就是应用运行期垃圾回收器暂停应用去回收内存空间的时间，此时应用是不可用的。最大停顿时间 目标就是去限制这些停顿的最大时间。垃圾回收器来保证暂停的平均时间(只是平均时间，没办法限制每次的停顿时间都符合预期)。**因为要尽量降低最大停顿时间，所有可能会触发更多的GC，也就是会产生更多的停顿次数。如果这样下来平均停顿时间还是比目标停顿时间大，那么这个目标也就没有达成**，用户可以自己取舍，如果超出的时间没有很多，那也可以接受，毕竟光靠这个参数，不能说百分百能够符合预期，JVM只能尽量去限制最大停顿时间。

最大停顿时间目标可以使用一个参数来指定：`-XX:MaxGCPauseMillis=<nnn>`。这个参数提示垃圾回收器，用户希望停顿时间小于`<nnn>`毫秒。垃圾回收器会调整Java堆大小和其他的参数来保证停顿时间小于预期时间。默认是没有最大停顿时间目标的。因为这会带来更多的GC次数，所以应用吞吐量可能会降低。

Throuhput Goal 吞吐量优先

吞吐量是由GC时间和GC外时间来计算的。这个目标可以通过参数: `-XX:GCTimeRatio=<nnn>` 来指定。GC时间与应用时间比率由这个公式计算得来: `1/(1+<nnn>)`。例如，`-XX:GCtimeRatio=19` 时，公式就是 `1/20`，也就是使用5%的时间来做垃圾回收。

Footprint Goal 内存使用量优先

如果吞吐量和最大停顿时间都已经达到预期，垃圾回收器就会减少堆内存直到满足其中一个目标(总会优先满足吞吐量优先)。

Tuning Strategy 调优策略

不要设定堆的最大值，除非你确定需要一个比默认值更大的堆。选择吞吐量优先的策略对你的应用来说一般就足够了。

选择吞吐量优先的话，堆内存会增大或缩小。应用的行为会决定堆内存是增大还是减小。例如，如果应用高频率的使用内存，heap size就会动态增大。

如果堆内存涨到最大值，但是吞吐量目标还是没有达成，那么这个堆的大小就是不合适的。将heap size设置到接近物理内存的大小，再重启应用。如果吞吐量的目标还是没有达成，说明机器到瓶颈了（根据我个人的经验，更可能的原因是代码写的烂）。

如果吞吐量优先的目标无法达成，但是有些GC停顿时间过长，就选择最大停顿时间目标。有时候就是需要对应用最一些权衡。

垃圾回收器为了满足优化目标时，heap size会动态的调整大小，这种情况是很平常的。即使应用到达了一个稳态，heap size也依然会震荡。