[译]Java8官方GC调优指南 --(一) 引子 - 掘金

juejin.cn/post/6844904053432909831

2020年1月27日

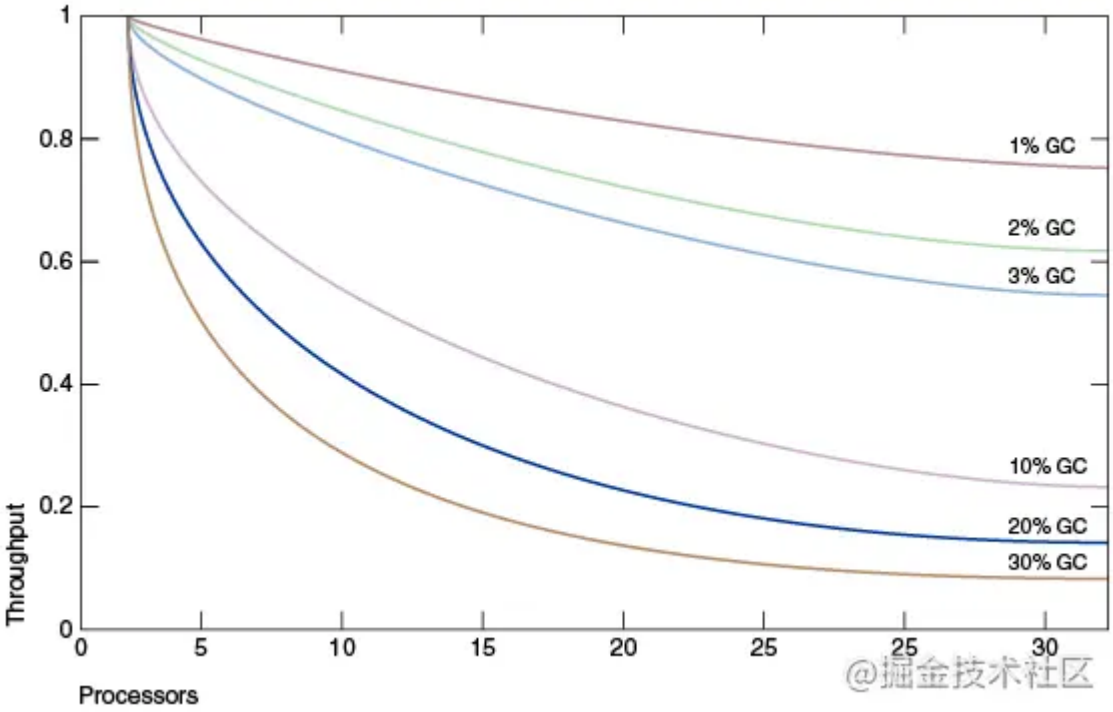
> 本套文章是Java8官方GC调优指南的全文翻译，点击查看原文,原文章名称《Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide 》

1 Inroduction 引子

Java程序有大有小，从小的桌面程序到大型工业级服务端应用。为了支持如此广的应用范围，JVM提供多个垃圾回收器，每一种都可以满足不同的需求。选择哪种回收器，取决于不同的应用场景。为了达到性能适合应用场景，我们需要不断的调整回收器和回收器的性能参数。这篇文档提供了一些帮助信息。 一个垃圾回收器(GC)是一个内存管理工具。通过下面的操作，GC可以实现内存的自动管理：

- 在年轻代分配对象和将年老的对象提升到老年代
- 通过并发标记来发现老年代中存活的对象。当Java Heap区使用量超过默认阈值时，JVM会触发标记操作。详情参考Concurrent Mark Sweep (CMS) Collector and Garbage-First Garbage Collector(G1).
- 使用parallel复制压缩存活对象来释放内存。详情参考The Parallel Collector and Garbage-First Garbage Collector

阿姆达尔定律表明很多工作量无法没完美的并行化；有些工作本身就是串行的，没有办法使用并行化来加速执行。Java也是如此。尤其是，java 1.4的jvm没有平行回收器，所以回收器在多处理器系统上的性能提升是相对的，不一定并发收集就特别好。 下图说明了gc时间消耗与cpu核数的关系，这是一个理想模型。每条线都代表一个Java程序，这条线后面的N%GC表示它在单核CPU的情况下花费多少时间去做GC，例如红色的线表示这个应用在单核CPU的系统上运行时只花费1%的时间来做垃圾回收。但是当系统CPU核数扩展到32时，它就需要花费20%以上的时间去做GC了，也就是说扩展CPU核数(使用并行计算)并不能提高程序的吞吐量。再看品红的那条线，单核时使用30%的时间去做GC，但是提升到32核时吞吐量已经不足15%，性能下降的十分剧烈。



这表示一个小问题在系统扩展后可能会被放大成整个系统的瓶颈(不知道阿里的JDK是不是做了优化)。这也说明你的参数调优在大型系统的场景下会有很大的性能提升，这是很有必要的。所以对于大型Java应用，有必要去选择正确的垃圾回收器并对参数进行调优。

serial collector 串行回收器对于那些堆内存在100MB左右的小型Java应用来说是足够的。其他的collectors有一些额外的复杂性和性能支出，因为这些回收器都会有一些特定的个性化行为。如果你的程序足够简单，你不需要这些特定的回收器特性，那么使用serial collector就足够了。不过serial collector不适合那些大型应用（运行时线程比较多，而且机器内存也很大，CPU核数在2个以上）。如果Java程序运行在服务器上，JVM默认使用parallel collector。详情可以看Ergonomics.

这篇文档是基于Java SE 8 和Solaris 操作系统，仅供参考。然而，这些概念和一些推荐配置适用所有的平台，包括Linux，Windows，和OS X。**另外，所有的命令行参数适用于所有的平台，不过对于每个平台，默认参数可能会不同。**