关注性能:调优垃圾收集Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E5_85_B3_E6_B3_A8_E6_80_A7_E8_c104_644965.htm 随着网志作为公共日记的流行，网志主机迅速地增长。所以对于 Blog-City 的人来说，非常清楚他们的站点需要发展和提高。为了满足其增长的需要，该公司最近刚刚推出了 Blog-City version 2.0。正像经常出现的情况那样，当新的应用程序转入运行阶段时，由于各种原因，其性能无法完全满足期望的要求，突然出现随机的长时间应用程序被挂起的现象还不是最坏的情况。 在其核心，Blog-City 依靠 Blue Dragon Servlet 引擎（CFML 引擎）和数据库。令人惊讶的是，所有这些软件都宿主在运行 Red Hat Linux 的相当老的 P3 机器上。这台机器具有单个硬盘和 512MB 内存，这对于过去的负载来说是足够强大的，但它正在承受不断增长的负载。Blog-City 的运作方式很成功，但其资源限制却成了其成功路上的绊脚石。尽管如此，这就是未来还要继续使用一段时间的所有硬件。 问题定义 整个过程的第一步是确定突然出现应用程序减慢的原因。首先我们怀疑的对象是垃圾收集。正如我们在 本专栏的上月文章中所论述的那样，确定垃圾收集和内存利用问题是否对应用程序产生负面影响的最容易的方式是，设置 -verbose:gc JVM 选项，并检查日志输出。因此我们重新启动应用程序，打开冗长的垃圾收集日志选项，然后耐心地等待应用程序的性能降低。我们的耐心换来的是非常详细的垃圾收集日志文件。 从对日志文件的最初分析中看，在这一应用程序中垃圾收集的瓶颈是显而易见的。种种迹象包括垃圾收集的频率、持续时间和总

体效率都已表明这一点。高于普通垃圾收集频率的常见原因是，堆的大小刚好足以适应所有当前正在使用的运行对象，无法适应新的正被创建的对象。虽然应用程序消耗大量堆可能有许多原因，但主要原因可能是没有足够内存而导致垃圾收集器运行，因为它设法满足当前需要。换句话说，应用程序试图分配新对象，但失败了，如果失败的话，将触发垃圾收集程序。如果垃圾收集失败而无法恢复足够内存，它将迫使另一个花费更大的垃圾收集程序发生。即使 GC 恢复了足够的空间来满足瞬间需求，可以肯定的是，在应用程序程序另一次分配失败，触发另一个 GC 之前，时间不会很长。因此，应该关注重复扫描空闲堆空间的无效任务，而不是服务于应用程序的 JVM。 应用程序逐步消耗所有可用的堆空间可能有许多原因，但如果有更多内存的话，临时解决方案就是配置更大的堆。假设应用程序没有内存泄漏（或者也就是我们常说的“ 无意识地保留对象 ”），它将找到一个“ 自然 ”级别的堆消耗，在这个级别中，GC 将能够很适应地得到维持（除非对象创建的速度过快，以至 GC 总是处于赛跑状态）。在这种情况下，以及无意识地保留对象的情况下，我们需要对应用程序做一些变动，以便获得某些改进。 如果仅仅是这样，那就太简单了 遗憾的是，我们必须面对严酷的现实因素正在运行的机器只有 512 MB 内存。更糟的是，我们必须与数据库和其他运行在机器中的进程共享该空间。要完整理解这一点为什么至关重要，首先您必须明确理解垃圾收集的基本知识，以及它如何与底层操作系统进行交互。 虚拟内存不再是灵丹妙药 操作系统已经使用虚拟内存许多年了。正如您所知道的，虚拟内存使操作系统的内存看起来比实际的内存

要多，这允许计算机运行那些所需内存比可用物理内存更大的程序，不使用内存的应用程序部分将保存在磁盘上。为了进一步简化，操作系统同时按页管理内存。页通常包含 512 字节到 8 KB，所有页的组合就组成了一个虚拟地址空间。操作系统维持一个页表，用于告诉操作系统如何映射虚拟地址到物理地址。当应用程序要求某个内存位置的内容时，操作系统（或硬件）将识别包含虚拟地址的页面。然后确定该页面是否在内存中，如果不在，将会报告页面错误。但是有许多种方式来处理页面错误，最终的结果是，页面必须从磁盘载入到内存中。这样应用程序就可以访问到有效虚拟地址的内容。 如果相关对象总是在内存的同一页面上聚合，那么 GC 的连续工作很可能出现困难。但是现实世界中，相关对象很少（如果有的话）出现聚合现象。实际结果是，依靠虚拟内存的系统将导致操作系统将页从内存中换入和换出，因为它标记然后废弃堆空间，而当聚合现象发生时，GC 将很多时间花在等待页面从磁盘换入而不是实际恢复内存上。因此，应用程序正在等待 GC，而 GC 正在等待磁盘，其间未完成任何真正的工作。由于本系统只有一个磁盘，并且它还需要支持数据库，因此我们在解决问题时处于两难境地。一方面，我们需要增加内存数量，这样我们可以减少 GC 的频率，但另一方面，我们还需要确保数据库的完好运行，而数据库也是内存的消耗大户。因此，我们需要了解应用程序所需的最小内存数量。 正如我们在上月看到的，在冗长的 GC 日志中这一信息可以很容易得到，无需为这一信息而扫描整个日志，我们使用免费的 JTune 工具（请参阅 参考资料）来解释冗长的 GC 日志。图 1 显示了经过垃圾收集之后的内存利用情

况，其中我们将 -Xmx 设置为 256 MB。图 1. 垃圾收集之后的内存利用情况 