HZ BOOKS 华章IT

云计算与虚拟化技术丛书

PEARSON vmware PRESS

Virtualizing and Tuning Large-Scale Java Platforms

大规模Java平台虚拟化与调优

[美] Emad Benjamin 著 张卫滨 文建国 译

VMware公司首席架构师亲笔撰写，全方位解读在VMWare vSphere上优化企业级Java应用的最佳技术实践和实用技巧

从计算机硬件体系结构到Java虚拟机的内存管理优化，从内存数据库的设计到整个应用分层的部署，详细讨论Java平台虚拟化和调优的原则、策略和技巧，包含大量实例

机械工业出版社
China Machine Press

云计算与虚拟化技术丛书

Virtualizing and Tuning Large-Scale Java Platforms

大规模Java平台虚拟化与调优

[美] Emad Benjamin 著　张卫滨 文建国 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

大规模 Java 平台虚拟化与调优 /（美）本杰明（Benjamin, E.）著；张卫滨，文建国译 .
—北京：机械工业出版社，2015.4
（云计算与虚拟化技术丛书）
书名原文：Virtualizing and Tuning Large-Scale Java Platforms

ISBN 978-7-111-49594-9

I. 大… II. ① 本… ② 张… ③ 文… III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2015）第 062067 号

本书版权登记号：图字：01-2014-2027

大规模 Java 平台虚拟化与调优

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：关 敏　　责任校对：董纪丽
印　刷：北京市荣盛彩色印刷有限公司　　版　次：2015 年 5 月第 1 版第 1 次印刷
开　本：186mm × 240mm 1/16　　印　张：12.5（含彩插 0.5 印张）
书　号：ISBN 978-7-111-49594-9　　定　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066　　投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259　　读者信箱：hzjsj@hzbook.com

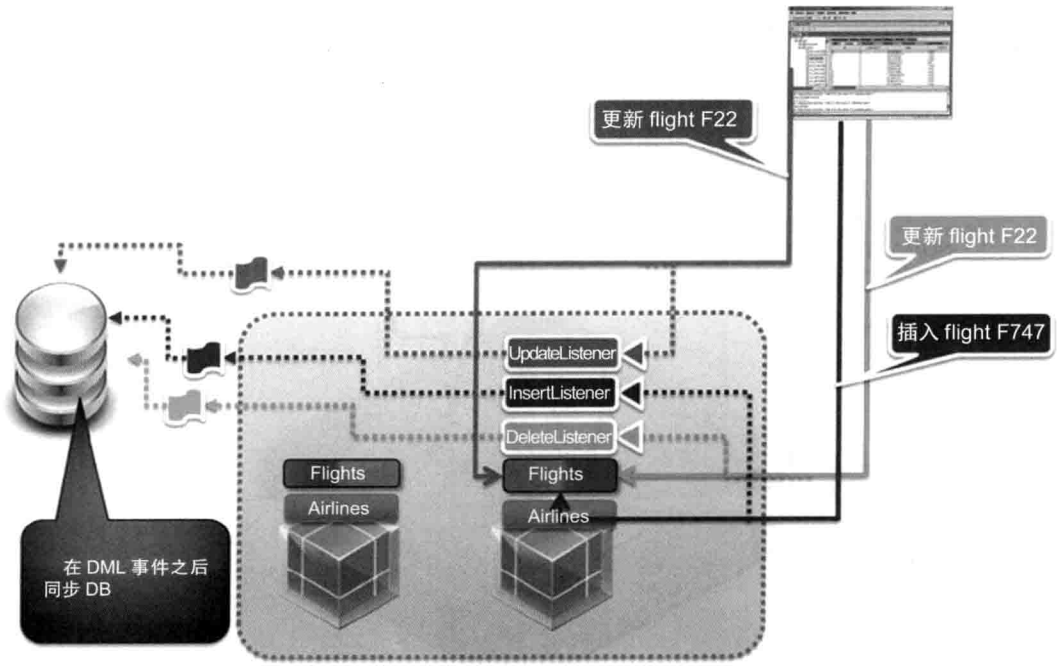


图 2-18　vFabric SQLFire SQL DML 监听器

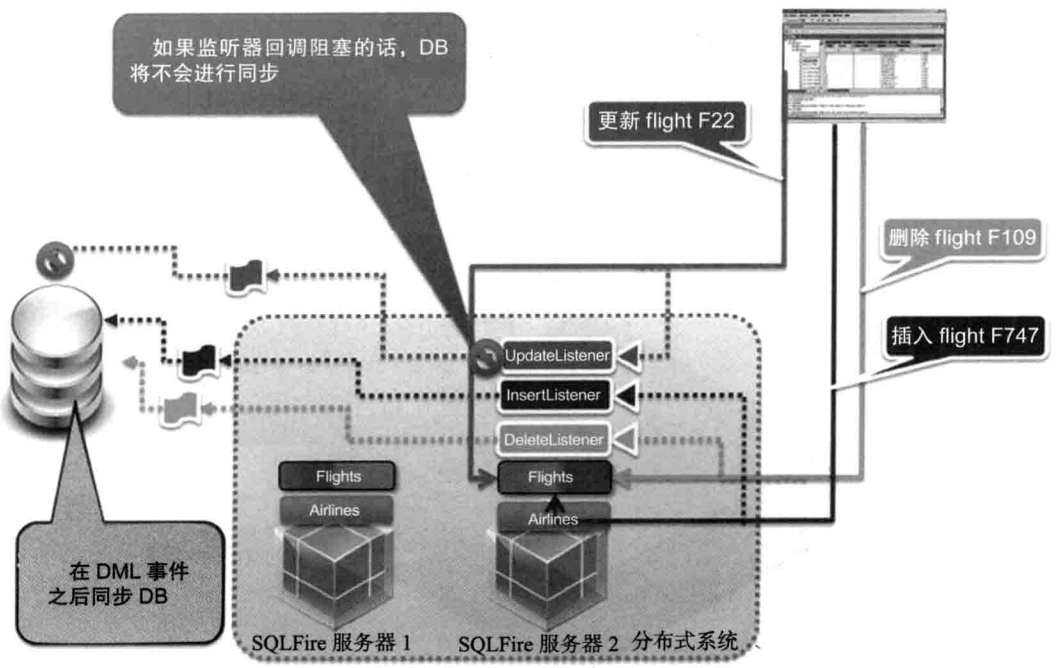


图 2-19　`UpdateListener` 回调处理器的阻塞

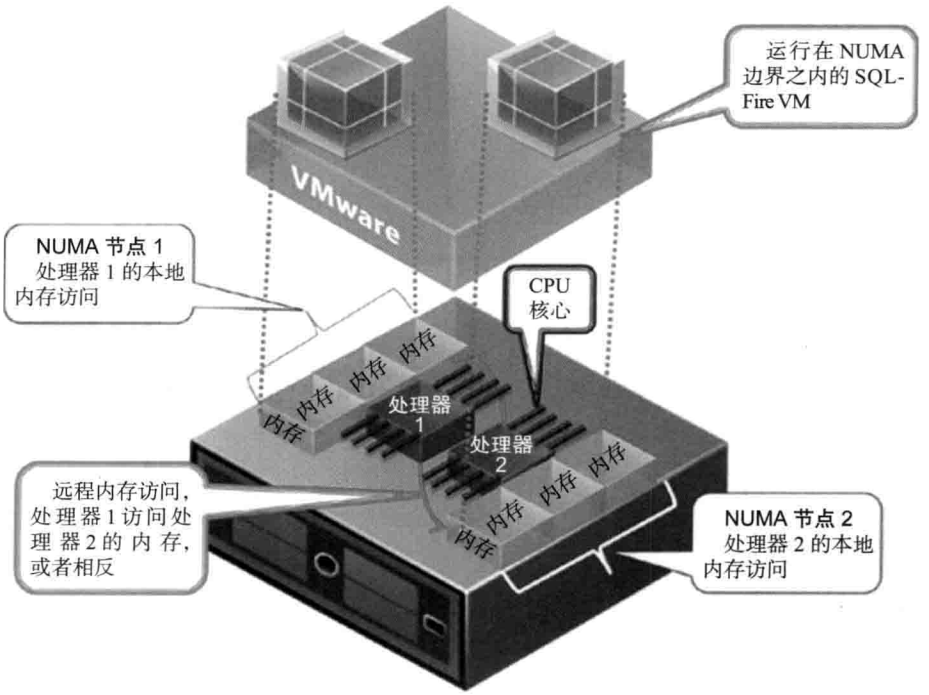


图 1-4　双插槽 8 核心的 vSphere 主机，具有两个 NUMA 节点，每个 NUMA 节点有一个 VM

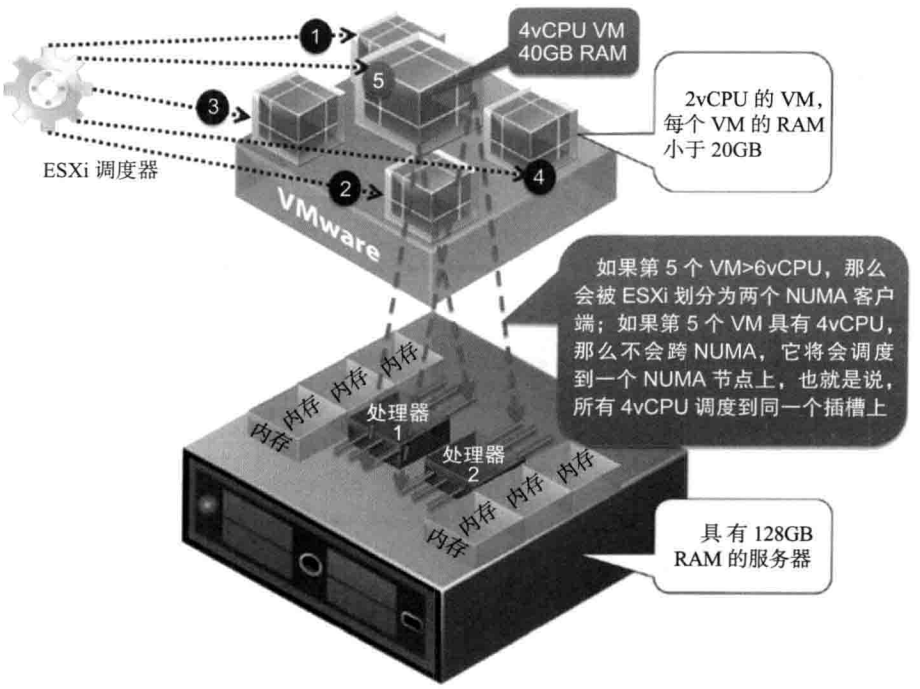


图 1-5　ESXi NUMA 对双插槽 6 核心服务器的调度

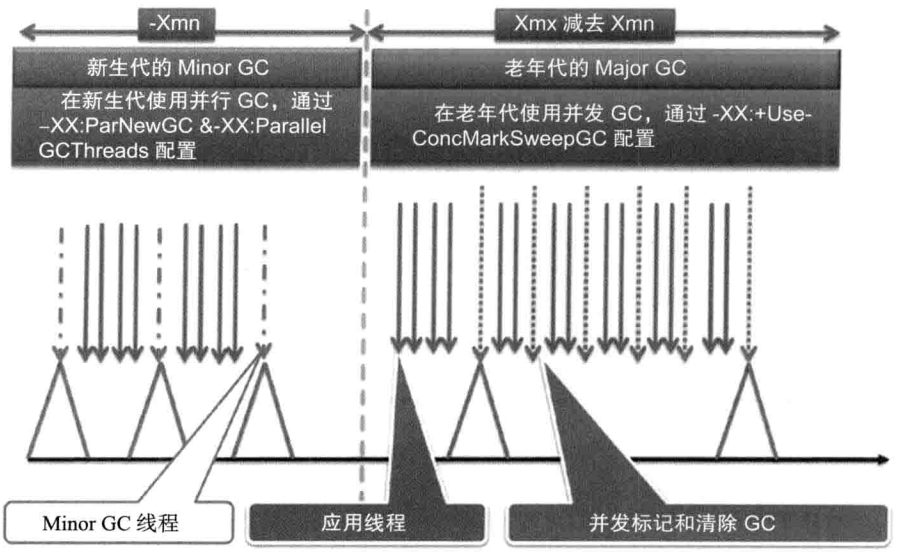


图 3-1　新生代中使用并行 GC 和老年代中使用 CMS

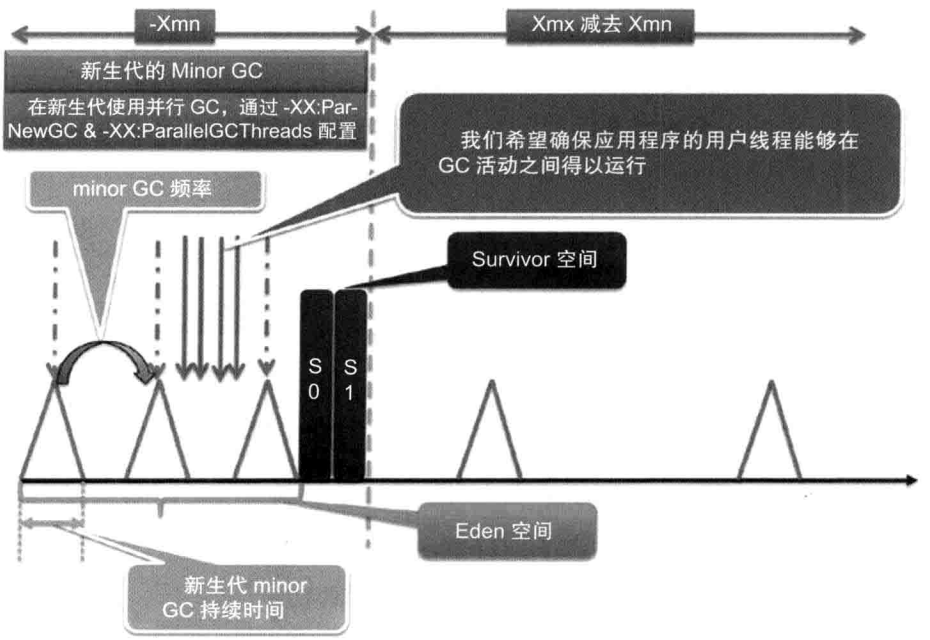


图 3-3　测量新生代中 minor GC 的持续时间和频率

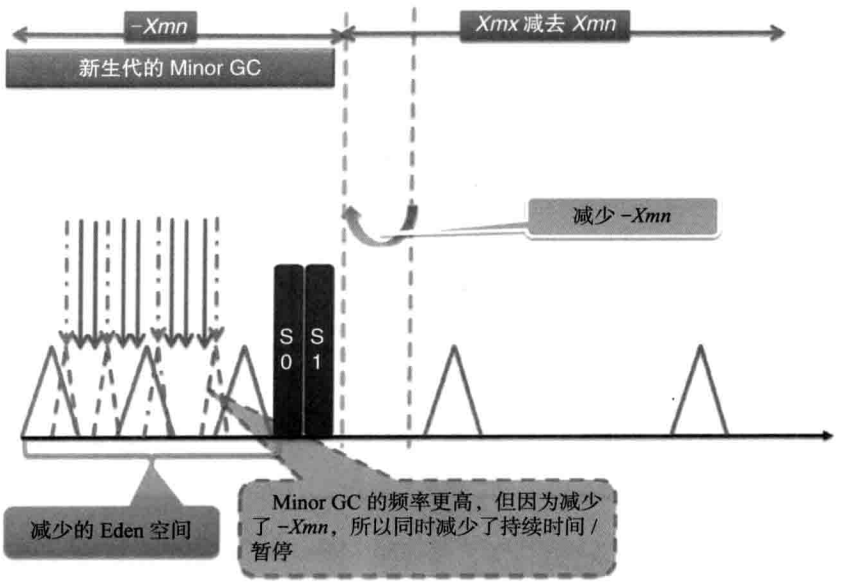


图 3-4　减小 *-Xmn* 的影响

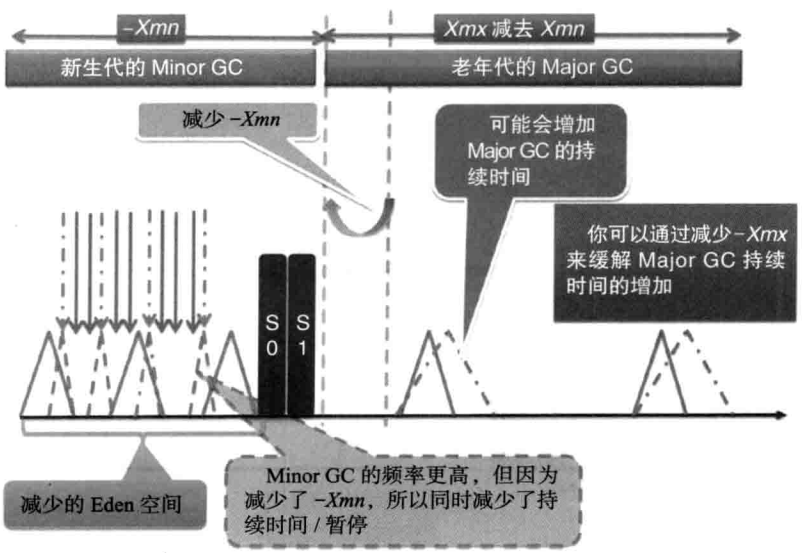


图 3-6　减小新生代对老年代的影响

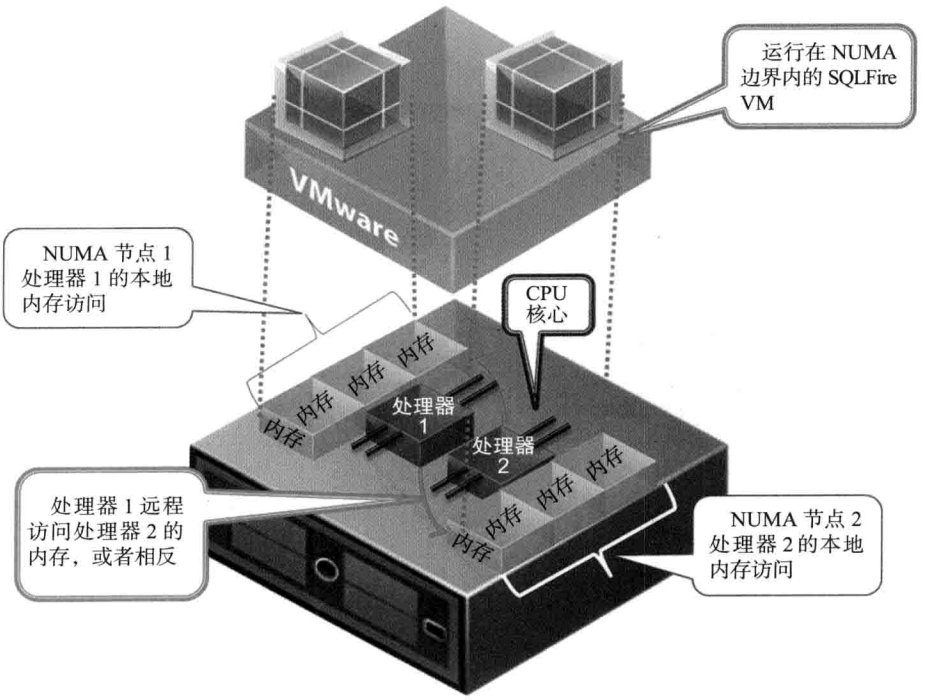


图 4-9　具有 2 个 vFabric SQLFire 的双插槽 NUMA 服务器（可选配置方案 1）

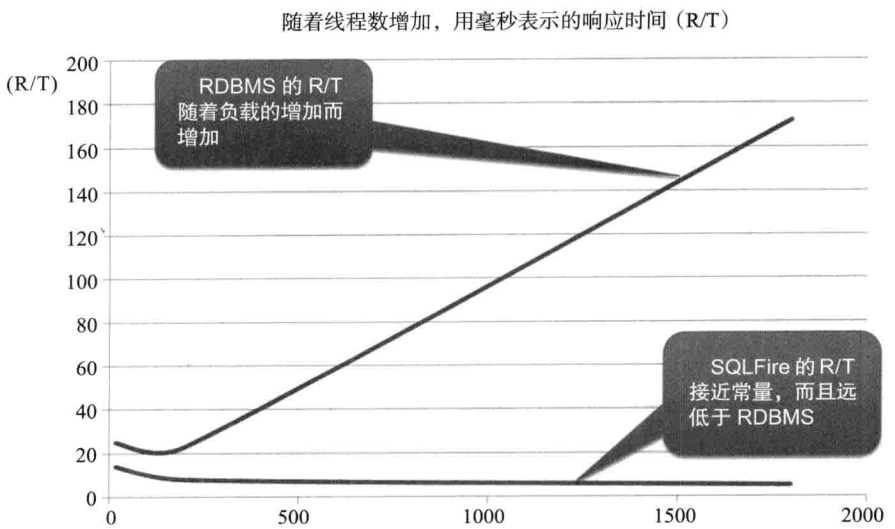


图 5-2　Spring Travel 响应时间和并发线程数

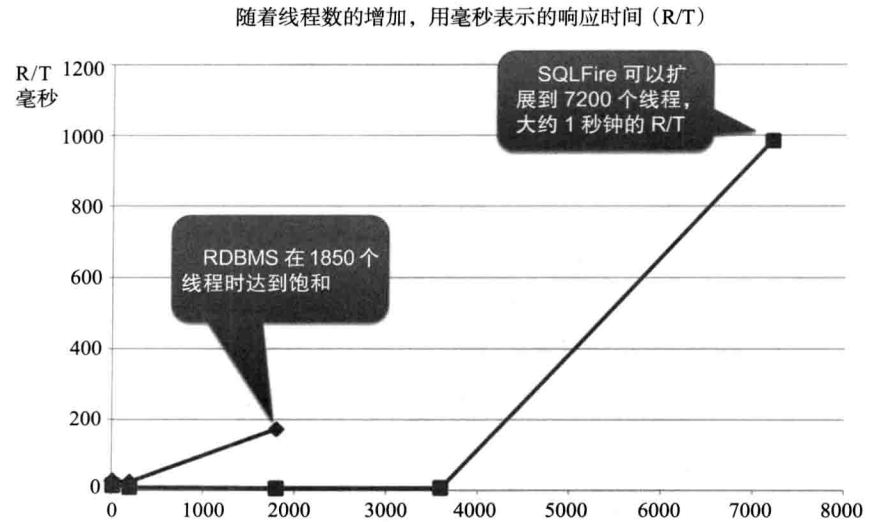


图 5-3　Spring Travel 响应时间和并发线程数：可扩展性测试

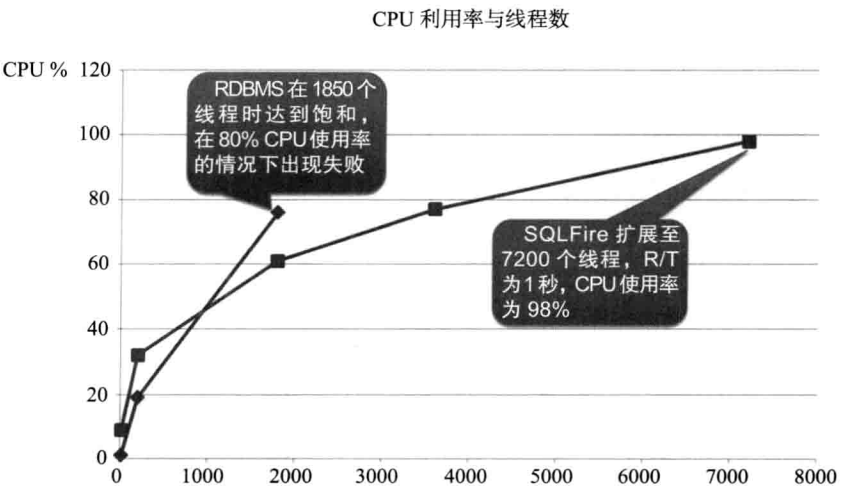


图 5-4　Spring Travel 应用程序的 CPU 和并发线程数

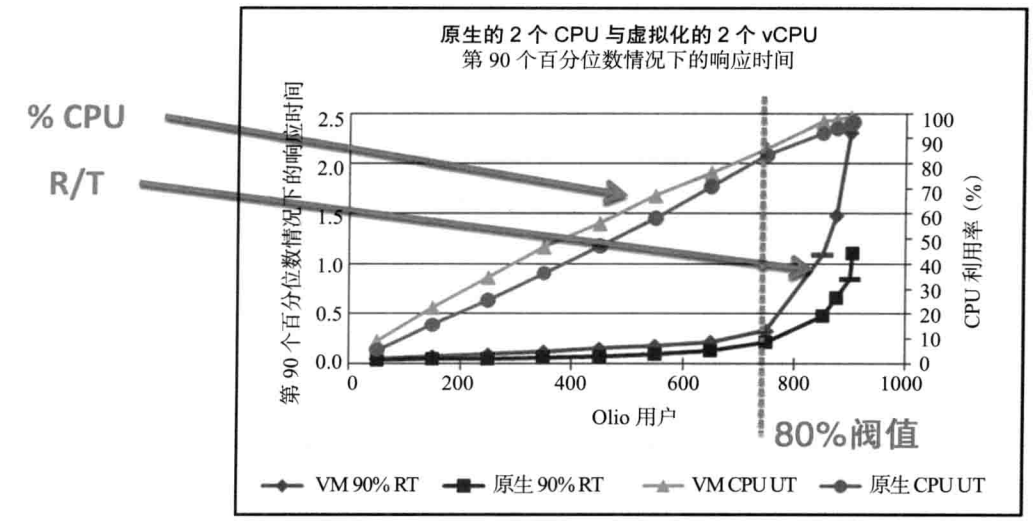


图 5-6　原生的 2 个 CPU 与虚拟化的 2 个 vCPU

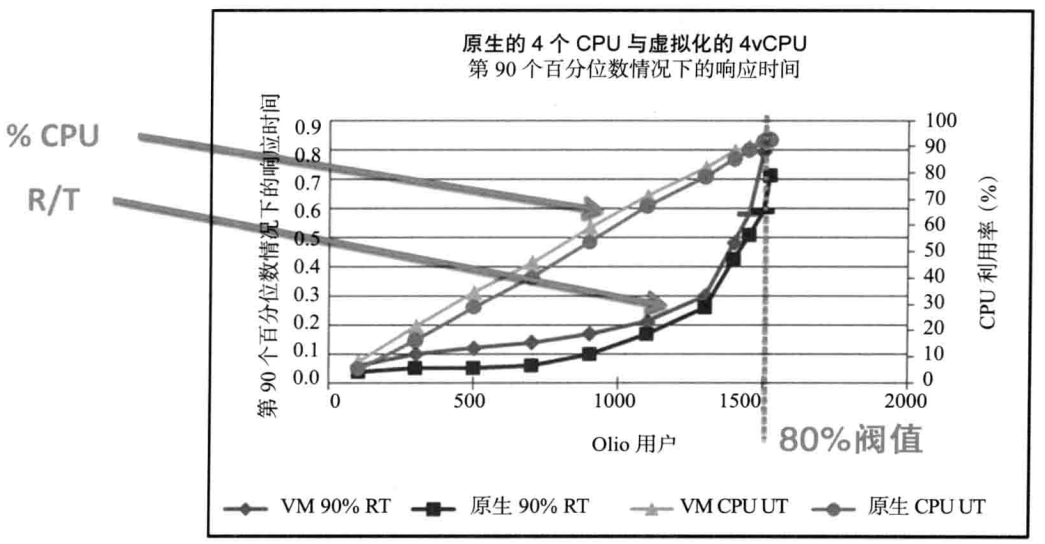
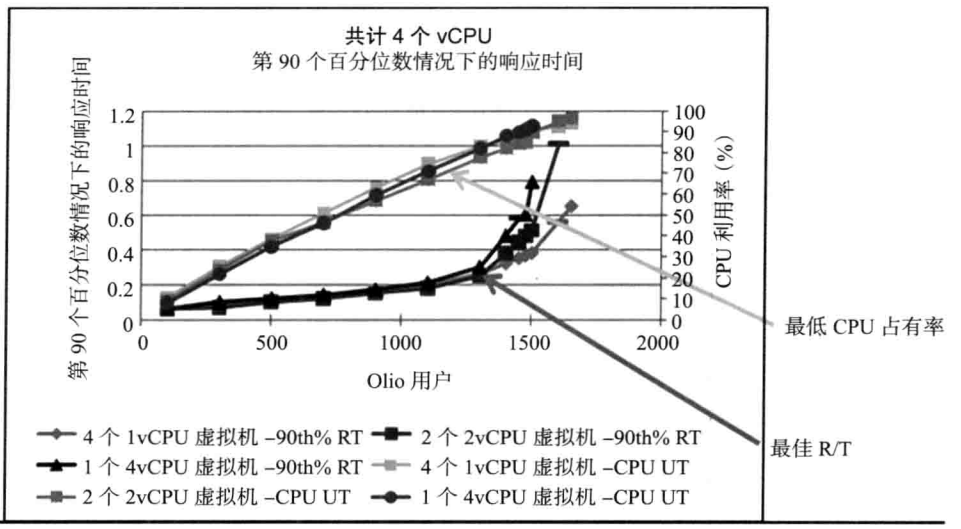


图 5-7　原生的 4 个 CPU 与虚拟化的 4 个 vCPU



每个 VM 上的 vCPU 数量	VM 的数量	每个 VM 上堆的最大值	实现 4 vCPU 的堆总额	
1	4	2GB	8GB	
2	2	2.5GB	5GB	最佳场景
4	1	4GB	4GB	

图 5-9　3 种配置选择：4 个 1-vCPU 的 VM、2 个 2-vCPU 的 VM 和 1 个 4-vCPU 的 VM

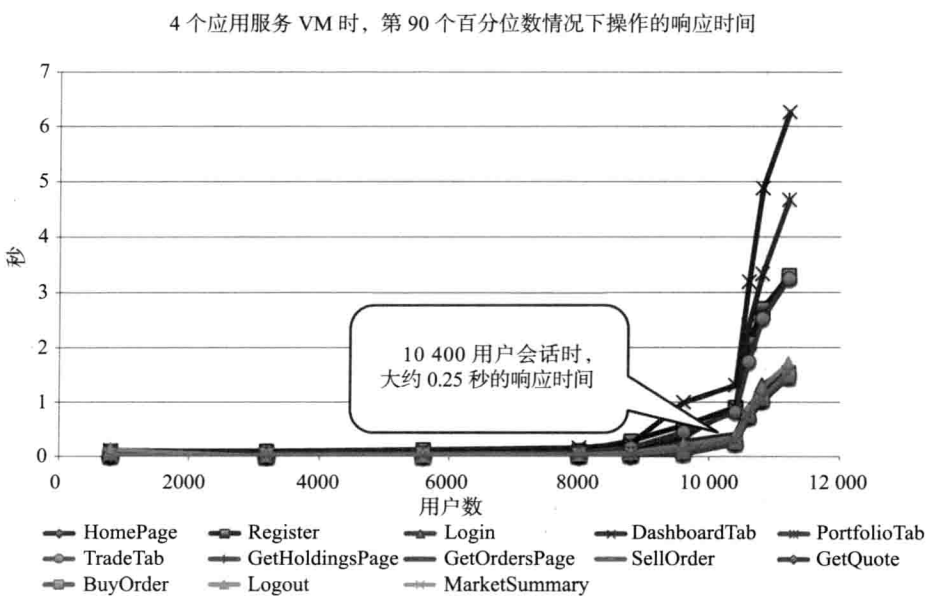


图 5-15　响应时间和用户数量

The Translator's Words 译者序

几年前，业界对于虚拟化和云计算是否能够在商业中真正应用还有许多争论，但目前，云计算和虚拟化已经广泛应用于蓬勃发展的互联网领域以及传统的企业级软件解决方案之中，软件开发和部署正在经历着剧烈的变革。借助云计算的浪潮，一些公司脱颖而出，站在了时代的前沿，这其中就包括 VMware——完整的虚拟化解决方案使其成为行业的领导者。

虚拟化在带来成本节省和运维便利的同时，也带来了一些新的挑战，对架构师和运维人员提出了新的要求。在解决了物理设施的虚拟化问题之后，我们的目光可能就会转移到应用程序本身上来，因为这才是真正为用户交付价值的关键所在。部署在虚拟化环境上的 Java 应用与物理环境上的应用有什么区别？对不同类型的 Java 应用如何进行优化？新的数据存储模式是什么？如何在虚拟化环境下最优化应用部署？对于这些问题，本书都给出了详尽的解答。本书的作者拥有 VMware 虚拟化的丰富经验，所传授的知识都来源于一线的实战经验，相信这些知识对于要进行虚拟化的架构师和运维工程师都会有很大帮助。

在翻译本书的过程中，深感作者知识领域的深厚和本书涉及内容的广泛，从计算机硬件体系结构到 Java 虚拟机的内存管理优化，从内存数据库的设计到整个应用分层的部署，本书都有介绍，因此，在翻译时我查阅了许多资料，这对于个人知识领域的拓展也有很大帮助。

在此要感谢关敏编辑所提供的帮助和指导，感谢我的搭档文建国同学，还要感谢一直以来给我支持和鼓励的朋友与家人。

尽管在翻译的过程中，我们力争达到准确和通畅，但限于水平和时间，书中肯定还有许多不足或纰漏之处，热忱期待你提出意见，希望本书能对你有所帮助！

张卫滨

前　言 Preface

本书是9年来我在VMware vSphere上运行Java应用的经验结晶，这些经验来源于VMware本身以及VMware的众多客户。实际上，很多VMware客户都在VMware vSphere上运行企业级的核心Java应用，并取得了效果更好的总拥有成本（total cost of ownership，TCO）以及服务水平协议（service level agreement，SLA）。我的第一本书是《Enterprise Java Applications Architecture on VMware》（VMware上的企业级Java应用架构），在那本书中很好地阐述了Java虚拟化的主题，其中既包括高层次的架构视角，也包括深入介绍分区大小设置和最佳实践的技术章节。为了保证第一本书在价格上更为实惠，我将一部分章节放到了第二本书，也就是你现在读到的这本书中。这两本书在很多方面都是互补的。在第一本书中有一些高屋建瓴的章节，是针对架构师、工程师以及管理者的，他们第一次考虑虚拟化方案并且可能会问“为什么这样做”的问题。而本书是关于如何做和做什么才能调整至最佳性能的。

限制第一本书的范围是个不错的主意，这样能让第一次构建Java虚拟化项目的人快速读完该书。第一本书出版至今已经有近2年的时间了，从那时到现在，我已经收到了近300条读者的反馈，这些反馈有助于进一步分析书中所给出的指导建议。其中有些反馈涉及大规模的Java平台，这些反馈极大地丰富了本书中的细节。本书会详细讨论分区设置以及小规模和大规模虚拟化Java平台的调优——从100个Java虚拟机（Java Virtual Machine，JVM）到10 000个JVM，JVM堆的大小从1GB到128GB。我最近的经验以及过去15年来优化Java平台所取得的经验都包含在本书中，我将这些经验进行了总结，以一种最实用并且能够立即应用于多种Java负载类型的形式进行了阐述。你可以改进本书所给出的建议、部署配置以及垃圾收集（garbage collection，GC）的优化知识来应对糟糕的GC行为，或者在整体上设计并确定Java平台的规模。本书中所强调的最佳实践可以应用于物理环境、虚拟化环境或者两者组合的环境之中。

撰写本书的动力

在过去的9年中，我在VMware担任不同的职位以确保所有内部的企业级Java应用都被虚拟化，以此向VMware的客户展现这种方式所能带来的收益。就在那个时候，我开始相信我们在生产环境下根据试验数据所得到的最佳实践应该分享给VMware社区。我收到了很多的反馈，要求我将在VMware上运行企业级Java应用方面所学到的经验以及获取成功的各种技巧进行文档化。这就是写作第一本书《Enterprise Java Applications Architecture on VMware》(https://www.createspace.com/3632131）的动力。

延续了写作第一本书的动力，本书（也就是第二本书）主要关注要调整什么、能调整到什么程度以及大型虚拟化Java平台该是什么样子的。实际上，第一本书的内容回答了“为什么虚拟化”以及“要做什么/如何来实现虚拟化”的问题。而本书探讨了“你能虚拟化多大规模的应用以及你对平台能够优化到什么程度”。

写作第一本书是非常令人兴奋的，因为我们试图让VMware的广大用户群了解Java虚拟化是完全可行的并且能够带来很明显的收益。在这本书中，我们将为一些客户提供帮助，这些客户的诉求可能是“现在请帮助我们将规模提升到一个新的水平”。在过去的2年间，我们帮助客户虚拟化了很多大规模的JVM平台，有些甚至达到10 000个JVM，有些涉及大数据平台领域（在一组集群JVM之中，将多个TB的数据存放于内存之中）。在你深入学习本书之前，需要记住的一点是：尽管本书中展现了很多的最佳实践，这些实践代表了最佳的配置指导，但是这些并不是强制性的要求。按照经验，我们发现大多数企业级Java应用的虚拟化很容易，并不需要担心太多的特定配置。实际上，在各种类型的企业级产品化应用之中，Java应用是进行虚拟化的最佳候选，因为它很容易取得成功。通过分享我们的经验，希望能够帮助你避免在虚拟化大规模Java平台时我们曾经遇到过的陷阱。

为了阐述在虚拟化环境中部署Java平台有多么棒（同时也为了消除虚拟化平台会成为问题的误解），我们构建了同时适用于物理化与虚拟化环境的最佳实践。在设计上，本书包含了针对物理化和虚拟化Java平台的最佳实践，所以能够帮助读者在进行虚拟化之前，纠正他们在物理化Java平台中所遇到的问题。当然，这并不是强制性的要求，在迁移到虚拟化的时候，客户可以选择继续保留物理化Java部署时的遗留问题。但是，他们至少能够意识到其物理化Java平台在设计和部署上的不足，这是他们希望将来要进行修正的。

这样的经历是很重要的一个历练过程，能够让我们清楚地认识到：问题实际上存在于客户自己的物理化环境中。因此，客户能够理解维护物理化Java平台上遗留问题的成本。例如，我们经常发现很多Java物理化平台有着糟糕的架构和错误的拓扑结构，很多有着上千个

混乱且不必要的 JVM。当我们与这些客户交流时，不管他们是将 Java 应用部署在物理环境中还是迁移到虚拟化环境中，我们都会带领他们了解这些最佳实践并确保这些环境要进行正确地规模划分和调优。再强调一遍，客户可以忽视我们所给出的建议，对代码和平台不做太多的修改或变更就将遗留的 Java 物理化平台部署到对等的虚拟化环境之中。但是，在迁移到虚拟化平台时，客户最近越来越认识到我们（以及其他人）所给出的最佳实践对于提升 Java 部署范式的价值。

我们学到的经验教训分为以下几类：

- 在生产环境下，肯定会出错，出错只是一个时间问题。所以，你必须要一丝不苟地考虑哪里可能会出错并制定前滚（roll-forward）和回滚（rollback）计划。这个计划中的练习过程有助于进一步强化 QA（质量保证）的测试计划。要注意的一点是，这并不是虚拟化环境所特有的。实际上，不管你处理的是物理化的还是虚拟化的基础设施，这都是同等严格的需求。但实际情况是，虚拟化为你提供了一种快速处理问题的机制（与之形成对比的是，在物理化场景下，你所能取得的灵活性会受到一定的限制，你必须围绕你所拥有的计算资源规划灵活性）。
- 当进行虚拟化的时候，企业级 Java 应用是最容易取得成功的。
- 每个人都在 Java 的各个分层中工作，这些分层了形成各种筒仓（silo），人们在技术和组织流程方面说着不同的术语。显然这是老式物理化（非虚拟化）范式下的做法，这些技术和组织上的筒仓是过去 10 多年来所形成的。但是，在 VMware 上虚拟化 Java 时，很重要的一个方面就是跨团队协作，它会驱动很多团队互相交流以促成最佳的设计。来自开发和运维的团队会多次坐到桌边进行讨论。
- 客户有时候会为其环境中遗留的问题进行辩解。由此导致的结果就是，客户需要付出额外的成本来管理物理环境中庞杂的 JVM，如果不进行及时补救，这些成本也将会带到虚拟化系统中。例如，你真的需要 5000 个具备 1GB 堆空间的 JVM 吗？它们能够进行合并吗？它们绝对可以合并，我们将向你展示如何通过减少许可证成本以及提高管理效率（因为你需要管理的 JVM 更少）实现费用的节省。
- 性能问题。客户经常盲目地将所有问题都归咎于虚拟化或 GC 的问题。但实际上，虚拟化并不是什么问题，而 GC 有时候的确会成为问题。如果存在 GC 问题，这并不是虚拟化所特有的，实际上这样的问题通常在物理化部署环境中同样存在。
- 在物理化 Java 平台中，使用大内存数据库是否可行（确实可以达到 1TB 的内存集群）？绝对是可以的，如果主要的目标是不惜任何代价实现事务，同时还要实现尽可能快的速度，那么对你来说这是正确的架构。我发现很多客户对此持怀疑的态度。他们在对这些类型的环境进行规模划分时并没有考虑底层的平台，对于他们来说这是一个糟糕的开始。当对这些数据平台进行规模划分（稍后会进行讨论）时，你必须

关注服务器的机器架构。在有些客户那里，我看到另外一个糟糕的实践就是他们试图将环境划分为 30 个左右的 JVM。这并不是正确的方式，因为让众多 JVM 相互之间以较高的速度频繁交流可能会导致整体延迟更多。显然，这些都是对延迟敏感且依赖内存（memory-bound）的负载，使用数量更少且内存更大的 JVM 部署模式所取得的效果会更好。如果比较两种不同部署方式的内存数据库性能，其中一种使用 30 个 JVM，另一种使用 8 个更大的 JVM，你会发现 8 个更大 JVM 的配置方案会更好一些。当然，这里要提示的一点是，更大的 JVM 要针对 NUMA 进行正确的规模划分，并且要进行恰当的 GC 调优。

- 当虚拟化的时候，大的内存数据库方案真的可行吗？在过去的几年间，我们发现在客户中有一种日渐增长的趋势就是虚拟化 Java 应用服务器。具体来讲，就是具有几千个 JVM 的大规模平台正在被虚拟化。有一种特殊的负载类型，那就是内存数据管理系统，它需要 TB 级别的内存并且对延迟是敏感的。对于这些内存集群，我们发现尽管 JVM 的数量更少了，但它们会有较大的堆空间，通常 JVM 的大小是 8 ～ 128GB（JVM 的数量通常会少于 12）。当然，12 这个数字本身并没有什么魔力，少的话可以是 3 个，而多的话可以达到 30 个。但是，你所拥有的 JVM 越多，在延迟性方面出现风险的可能性就越大，这是因为会出现额外的网络传输。在本书后面，你会学习如何对这种负载进行规模划分和调优。

很多 Java 应用开发人员很了解开发过程，他们知道如何编写 Java 代码以及如何对 JVM 进行调优。但是，这些信息通常只存留在开发人员那里，并没有分享（或传递）给应用的管理人员。通常，运行 Java 平台所需要的技能在开发人员和管理人员之间是分割的，没有一个人能够完全理解这两个方面。但是，这种孤立式的理解正在发生着变化，因为有更多的人开始去理解如何编写和部署 Java 代码、调优 JVM 并认识虚拟化的各个方面和复杂的服务器硬件架构。所以本书的另外一个目标就是，当读者学习这些技能时，鼓励他们遵循这样的职业发展路径。

我非常真诚地希望本书能够帮助那些具有 Java 开发、基础设施运维以及虚拟化背景的人。你可以将本书作为应对日常情况的指导，也可以作为构建 Java 平台架构策略的帮助手册。

预备知识

本书假设读者已经在整体上了解 Java、JVM GC、服务器硬件架构以及虚拟化技术。这里的知识都是关于运行大规模 Java 平台的。在深入学习本书的内容之前，你可能希望先学习一下虚拟化，但是大多数 Java 的高级专家借助本书就足以掌握关于虚拟化及虚拟化 Java 应用的知识。实际上，通过回答下面的这个问题就能掌握继续阅读本书所需的虚拟化背景知

识。这个问题是一位刚刚接触虚拟化的人某一天提出的：“Java 是不是能够独立于操作系统及 hypervisor？”

本书假设读者已经掌握了一些关于 Java 语言特别是 JVM 架构的背景知识。本书尽可能地概述了 JVM 架构以及具体的调优方法，但是本书不能代替专门的 JVM 优化图书。我只想说，刚刚接触 JVM 调优的 vSphere 管理员能够从本书中学习到足够的知识，以便于他们与从事 Java 的同事进行技术交流。除此之外，vSphere 和 Java 管理员可以使用并修改本书的调优建议，并将其应用于自己的环境之中。JVM 调优建议的各个章节在编写上比同类图书更为简单易懂，目的是为了让 vSphere 和 Java 管理员能够快速了解并有效地运用设计和优化策略。众多运行 Java 应用的 VMware 客户使用本书中所讨论的优化参数并立即得到了性能方面的提升。

你首先需要知道些什么

你在阅读本书就意味着你正在修正你的 Java 平台。可能你已经得出这样的结论，那就是 Java 平台的调优不能被忽略或轻视。但是，即便你刚刚涉足 VMware 虚拟化或者是在物理化系统中对 Java 进行调优，那么本书也是适合你的。作为补习内容（对于刚刚接触这些领域的人来说也可以作为入门介绍），以下的章节简要介绍了虚拟化与大规模 Java 平台调优相关的重要概念。

4GB 的 Java 堆是不是相当于新的 1GB 的 Java 堆呢，为什么

在过去的 2 年中，我参与了超过 290 个客户电话和研讨会，了解到很明显的一点是，运行在 Java 平台上的 40% 的工作负载都部署到 1 ～ 4GB 的 JVM 上。我进而发现有大量小于 1GB 的 JVM，大约占到了我接触客户的另外 40%。而剩余的 20% 范围在 4 ～ 360GB 之间。是的，这个 360GB 的 JVM 是一个监控系统，它不能够水平扩展，所以客户只能使用一个 JVM。尽管这听起来有点令人难以置信，但这确实是 Java 产品化平台的现状。不过，使用 1GB 堆的 JVM 会导致 JVM 实例数量的膨胀，在自身的管理方面这会成为令人头痛的问题。例如，你可能希望提供共计 1TB 的堆空间，如果你只使用 1GB 的 JVM 堆，那么这就意味着你有几千个 JVM 实例。这可能带来什么问题呢？你是否能够通过 250 个 4GB 的 JVM 得到 1TB 呢？当然可以，但因为在你的组织中，依然存在来自于古老的 32 位 JVM 的遗留规则，所以你还得继续使用小于 1GB 的 JVM。更为现实的原因是，人们广泛认为更大的 JVM 会导致更长时间的 GC 停顿。这种说法并不完全正确，但也并不能说完全错误。的确，这会产生更大的停顿，但是随着 64 位 JVM 以及 CMS（concurrent mark-sweep）GC 的最新发展，规模