

Linux 性能和 调优指南

操作系统调优方法

性能监控工具

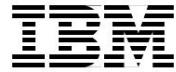
性能分析



爱德华多·西利恩多
竹近国正

红皮书

Machine Translated by Google



国际技术支持组织

Linux 性能和调优指南

2007年7月

注意 :在使用此信息及其支持的产品之前 ,请阅读第 vii 页 “声明”中的信息。

第一版 (2007 年 7 月)

此版本适用于内核 2.6 的 Linux 发行版。

本文于2008年4月25日更新。

內容

2.1 简介	40
2.2 工具功能概述..	40
2.3 监控工具... 2.3.1 top .. 2.3.2 vmstat..	41
.....	41
.....	42
2.3.3 正常运行时间.. 2.3.4 ps	43
和 pstree .. 2.3.5 空闲. 2.3.6 iostat.	44
.....	46
.....	48
2.3.7 SAR。 2.3.8	50
mpstat。 2.3.9 努马司他。	51
2.3.10 pmap。 2.3.11 网络统计。	52
2.3.12 iptraf。 2.3.13	52
tcpdump/ethereal。 2.3.14	53
纳米。	54
.....	55
.....	58
2.3.15 strace。	59
2.3.16 Proc 文件系统。	60
2.3.17 KDE 系统防护..	62
2.3.18 Gnome 系统监视器..	67
2.3.19 容量管理器...	67
2.4 基准工具...	70
2.4.1 LMbench	71
2.4.2 IO 区。 2.4.3 网络性能。.	72
.....	73
2.4.4 其他有用的工具..	76
第 3 章分析性能瓶颈..	77
3.1 识别瓶颈..	78
3.1.1 收集信息。	78
3.1.2 分析服务器的性能。	80
3.2 CPU 瓶颈..	81
3.2.1 查找 CPU 瓶颈..	81
3.2.2 对称多处理器 (SMP)。	81
3.2.3 性能调整选项.	82
3.3 内存瓶颈。	82
3.3.1 查找内存瓶颈...	82
3.3.2 性能调整选项.	84
3.4 磁盘瓶颈..	84
3.4.1 查找磁盘瓶颈。	84
3.4.2 性能调整选项.	87
3.5 网络瓶颈。	87
3.5.1 查找网络瓶颈..	87
3.5.2 性能调整选项.	89
第 4 章调整操作系统...	91
4.1 调整原则	92
4.1.1 变更更管理。	92
4.2 安装注意事项...	92
4.2.1 安装...	92
4.2.2 检查当前配置..	94
4.2.3 尽量减少资源使用..	97

4.2.4 SELinux...	102
4.2.5 编译内核...	104
4.3 更改内核参数。	104
4.3.1 参数存储在哪里。	106
4.3.2 使用 sysctl 命令。	107
4.4 调整处理器子系统...	107
4.4.1 调整进程优先级。	108
4.4.2 中断处理的 CPU 亲和性。	108
4.4.3 NUMA 系统的注意事项。	108
4.5 调整 vm 子系统..	109
4.5.1 设置内核交换和 pdflush 行为。	109
4.5.2 交换分区...	110
4.5.3 大型TLBfs...	111
4.6 调整磁盘子系统。	112
4.6.1 安装 Linux 之前的硬件考虑。 ...	113
4.6.2 I/O电梯调整与选择。	115
4.6.3 文件系统的选择和调整。	120
4.7 调整网络子系统..	124
4.7.1 交通特性的考虑。	124
4.7.2 速度和双工。	125
4.7.3 MTU 大小。	126
4.7.4 增加网络缓冲区。	126
4.7.5 额外的 TCP/IP 调整...	128
4.7.6 Netfilter 的性能影响。 ...	132
4.7.7 卸载配置。 .	133
4.7.8 增加数据包队列..	135
4.7.9 增加传输队列长度..	135
4.7.10 减少中断..	135
附录 A. 测试配置..	137
硬件和软件配置...	138
Linux 安装在来宾 IBM z/VM 系统上。 ...	138
安装在 IBM System x 服务器上的 Linux。	138
缩写和首字母缩略词。	141
相关出版物。	143
IBM 红皮书...	143
其他出版物..	143
在线资源。	143
如何获取 IBM 红皮书。	145
IBM 的帮助...	145
指数。	147

通知

此信息是针对在美国提供的产品和服务而制定的

IBM 可能不会在其他国家/地区提供本文档中讨论的产品、服务或功能。请咨询您当地的 IBM 代表，了解您所在地区当前提供的产品和服务。任何对 IBM 产品、程序或服务的引用并非明示或暗示只能使用该 IBM 产品、程序或服务。任何功能等效且不侵犯 IBM 知识产权的产品、程序或服务均可替代。但是，用户有责任评估和验证任何非 IBM 产品、程序或服务的运行情况。

IBM 可能拥有或正在申请与本文档所述主题相关的专利。提供本文档并不授予您使用这些专利的任何许可。您可以将许可咨询以书面形式发送至：IBM 许可总监，IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 USA

以下段落不适用于英国或此类条款与当地法律相抵触的任何其他国家/地区：国际商业机器公司“按原样”提供本出版物，不附带任何种类的（无论是明示的还是默示的）保证，包括但不限于默示的有关非侵权、适销性或适用于特定用途的保证。某些州/地区不允许在某些交易中免除明示或默示的保证，因此本声明可能不适用于您。

本信息可能包含技术错误或印刷错误。本出版物中的信息会定期更改；这些更改将纳入出版物的新版本中。IBM 可能随时对本出版物中描述的产品和/或程序进行改进和/或更改，恕不另行通知。

本信息中对非 IBM 网站的任何引用仅为方便起见而提供，并不以任何方式构成对此类网站的认可。此类网站上的资料不属于本 IBM 产品资料的一部分，使用此类网站的风险由您自行承担。

IBM 可以以其认为适当的任何方式使用或分发您提供的任何信息，而无需对您承担任何义务。

有关非 IBM 产品的信息均来自这些产品的供应商、其已发布的公告或其他公开来源。IBM 尚未测试这些产品，因此无法确认其性能、兼容性或任何其他与非 IBM 产品相关的声明的准确性。如有任何关于非 IBM 产品功能的疑问，请咨询这些产品的供应商。

本信息包含日常业务运营中使用的数据和报告示例。为了尽可能完整地说明，示例中包含个人、公司、品牌和产品的名称。

所有这些名称都是虚构的，与实际商业企业使用的名称和地址的任何相似之处纯属巧合。

版权许可：

本信息包含源语言编写的示例应用程序，用于演示各种操作平台上的编程技术。您可以以任何形式复制、修改和分发这些示例程序，而无需向 IBM 付费，用于开发、使用、营销或分发符合示例程序所针对的操作平台的应用程序编程接口 (API) 的应用程序。这些示例程序并未在所有条件下进行全面测试。因此，IBM 无法保证或暗示这些程序的可靠性、适用性或功能。

商标

以下术语是国际商业机器公司在美国和/或其他国家或地区的商标：

红皮书 (徽标)® 

eServer™

xSeries®

z/OS®

AIX®

DB2®

DS8000™

IBM®

动力™

红皮书®

ServeRAID™

System i™

系统 p™

System x™

系统 z™

系统存储™

TotalStorage®

以下术语是其他公司的商标：

Java、JDBC、Solaris 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和/或其他国家或地区的商标。

Excel、Microsoft、Windows 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

Intel、Itanium、Intel 徽标、Intel Inside 徽标和 Intel Centrino 徽标是英特尔公司或其子公司在美国和/或其他国家或地区的商标或注册商标。

UNIX 是 The Open Group 在美国和其他国家的注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他人的商标或服务标志。

前言

Linux® 是一个由世界各地的人们开发的开源操作系统。其源代码可免费获取，并可在 GNU 通用公共许可证下使用。Red Hat 和 Novell 等公司以发行版的形式向用户提供该操作系统。一些桌面 Linux 发行版可以从网上免费下载，但服务器版本通常需要购买。

过去几年，Linux 已成功进入全球众多企业的数据中心。Linux 操作系统深受科研和企业用户的青睐。如今，Linux 已成为功能最为丰富的操作系统。您可以在防火墙、手机和大型机等嵌入式设备上找到 Linux 的身影。因此，Linux 操作系统的性能自然也成为科研和企业用户关注的热点话题。

然而，计算全球天气预报和托管数据库对操作系统的要求有所不同。Linux 必须以最佳性能适应所有可能的使用场景。大多数 Linux 发行版都包含通用的调优参数，以适应所有用户的需求。

IBM® 认为 Linux 是一款适合在 IBM 系统上运行企业级应用程序的操作系统。目前，大多数企业应用程序都可以在 Linux 上运行，包括文件和打印服务器、数据库服务器、Web 服务器以及协作和邮件服务器。

在企业级服务器中使用 Linux 需要监控性能，并在必要时调优服务器以消除影响用户的瓶颈。这份 IBM 红皮书介绍了可用于调优 Linux 的方法、可用于监控和分析服务器性能的工具，以及特定服务器应用程序的关键调优参数。本文旨在解释如何分析和调优 Linux 操作系统，以便为您计划在这些系统上运行的任何类型的应用程序提供卓越的性能。

我们测试环境中使用的调优参数、基准测试结果和监控工具均在 IBM System x™ 服务器和 IBM System z™ 服务器上运行的 Red Hat 和 Novell SUSE Linux 内核 2.6 系统上执行。不过，本文中的信息对所有 Linux 硬件平台都适用。

本文的结构

为了帮助 Linux 或性能调优新手快速入门，我们按照以下方式构建了本书：

第 1 章，第 1 页上的“了解 Linux 操作系统”

本章介绍影响系统性能的因素以及 Linux 操作系统管理系统资源的方式。本章还将介绍量化系统性能所需的几个重要性能指标。

第 2 章，“监控和基准测试工具”（第 39 页）

第二章介绍了可用于 Linux 测量和分析系统性能的各种实用程序。

第 77 页上的第 3 章“分析性能瓶颈”

本章介绍识别和分析系统瓶颈的过程。

第 91 页上的第 4 章 “调整操作系统”

掌握了操作系统工作原理和使用性能测量实用程序的基本知识后,您就可以探索 Linux 操作系统中可用的各种性能调整了。

撰写本文的团队

本文由国际技术支持组织罗利中心的来自世界各地的专家团队撰写。



队员:拜伦、爱德华多、竹近

Eduardo Ciliendo是一位 IT 顾问专家,在 IBM 瑞士公司担任 IBM 大型机系统性能专家。他在计算机科学领域拥有超过 10 年的经验。Eddy 曾在苏黎世大学学习计算机与商业科学,并拥有日语学后学位。Eddy 是 zChampion 团队的成员,拥有多项 IT 认证,包括 RHCE 认证。作为 IBM System z™ 的系统工程师,他致力于 z/OS® 和 Linux for System z 的容量规划和系统性能研究。Eddy 撰写过多部关于系统性能和 Linux 的出版物。

Takechika Kunimasa是 IBM 日本全球服务部的助理 IT 架构师。他毕业于千叶大学电气电子工程专业,拥有超过 10 年的 IT 行业经验。他曾担任网络工程师五年,并一直从事 Linux 技术支持工作。他的专业领域包括 Linux on System x™、Linux on System p™、Linux on System z、高可用性系统、网络和基础架构设计。他是思科认证网络专家 (Cisco Certified Network Professional) 和红帽认证工程师 (Red Hat Certified Engineer)。

拜伦·布拉斯韦尔 (Byron Braswell) 是国际技术支持组织 (ITSO) 罗利中心的网络专家。他拥有德克萨斯农工大学物理学士学位和计算机科学硕士学位。他在网络、应用集成中间件和个人计算机软件领域撰写了大量文章。加入 ITSO 之前,拜伦曾在 IBM 学习服务开发部门从事网络教育开发工作。

感谢以下人员对本项目的贡献：

玛格丽特·蒂克诺

卡罗琳·布里斯科

国际技术支持组织罗利中心

罗伊·科斯塔

迈克尔·B·施瓦茨

弗里德·哈姆

国际技术支持组织,波基普西中心

克里斯蒂安·埃尔哈特

马丁·卡默勒

IBM 德国伯布林根

埃尔万·奥弗雷特

IBM 法国

成为一名出版作家

加入我们为期两到六周的驻留项目!参与撰写 IBM 红皮书,涵盖特定产品或解决方案,同时获得前沿技术的实践经验。您将有机会与 IBM 技术专家、业务合作伙伴和客户合作。

您的努力将有助于提高产品认可度和客户满意度。此外,您还将在 IBM 开发实验室建立人脉网络,提升您的生产力和市场竞争力。

了解有关居住计划的更多信息,浏览居住索引,并在线申请:

ibm.com/redbooks/residencies.html

欢迎评论

您的评论对我们很重要！

我们希望我们的论文能够尽可能地帮助到您。您可以通过以下方式之一向我们发送您对本文或其他 IBM Redbooks® 的评论：

使用在线联系我们审核红皮书表格,网址为：

ibm.com/redbooks

请将您的评论通过电子邮件发送至：

redbooks@us.ibm.com

请将您的意见邮寄至：

IBM 公司国际技术支持组织部 HYTD 邮件站 P099 2455 South Road Poughkeepsie, NY

12601-5400



第 1 章了解 Linux 操作系统

本文首先概述 Linux 操作系统如何处理其任务以完成与硬件资源的交互。性能调优是一项具有挑战性的任务，需要深入了解硬件、操作系统和应用程序。如果性能调优很简单，我们即将探讨的参数就会被硬编码到固件或操作系统中，您也就不会读到这些内容了。然而，如图 1-1 所示，服务器性能受多种因素影响。

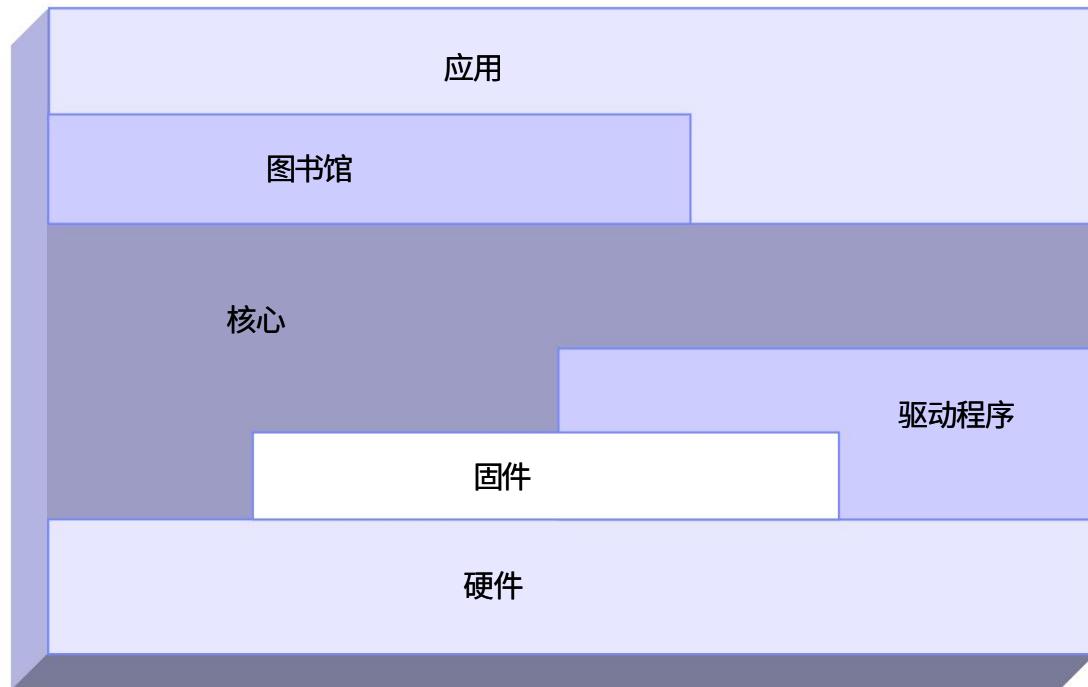


图1-1 不同性能组件交互示意图

如果一台拥有 20,000 个用户的数据库服务器的磁盘子系统只包含一个 IDE 驱动器 ,那么您可能需要花费数周时间对 I/O 子系统进行调优,但结果可能毫无意义。通常情况下,更新驱动程序或更新应用程序就能带来显著的性能提升。在讨论具体细节时,请始终牢记系统性能的整体情况。了解操作系统管理系统资源的方式有助于我们理解在任何应用场景下需要调优哪些子系统。

以下部分简要介绍 Linux 操作系统的架构。Linux 内核的完整分析超出了本文的讨论范围。您可以参考内核文档,获取 Linux 内核的完整参考信息。

注:本文主要关注Linux操作系统的性能。

本章将介绍:

- 1.1、第 2 页上的“Linux 进程管理”
- 1.2、第 10 页“Linux 内存架构”
- 1.3、第 15 页上的“Linux 文件系统”
- 1.4、第 19 页上的“磁盘 I/O 子系统”
- 1.5、“网络子系统” 第 26 页
- 1.6, “了解 Linux 性能指标” (第 34 页)

1.1 Linux进程管理

进程管理是任何操作系统最重要的功能之一。有效的进程管理使应用程序能够稳定有效地运行。

Linux 进程管理实现与 UNIX® 实现类似,包括进程调度、中断处理、信号发送、进程优先级排序、进程切换、进程状态、进程内存等。

在本节中,我们将讨论 Linux 进程管理实现的基础知识。它可以帮助您了解 Linux 内核如何处理会影响系统性能的进程。

1.1.1 什么是进程?

进程是在处理器上运行的执行实例。进程使用 Linux 内核能够处理的任何资源来完成其任务。

Linux 操作系统上运行的所有进程都由task_struct结构管理,该结构也称为进程描述符。进程描述符包含单个进程运行所需的所有信息,例如进程标识、进程属性以及构建进程所需的资源。如果您了解进程的结构,就能理解哪些因素对进程的执行和性能至关重要。图 1-2 显示了与进程信息相关的结构概要。

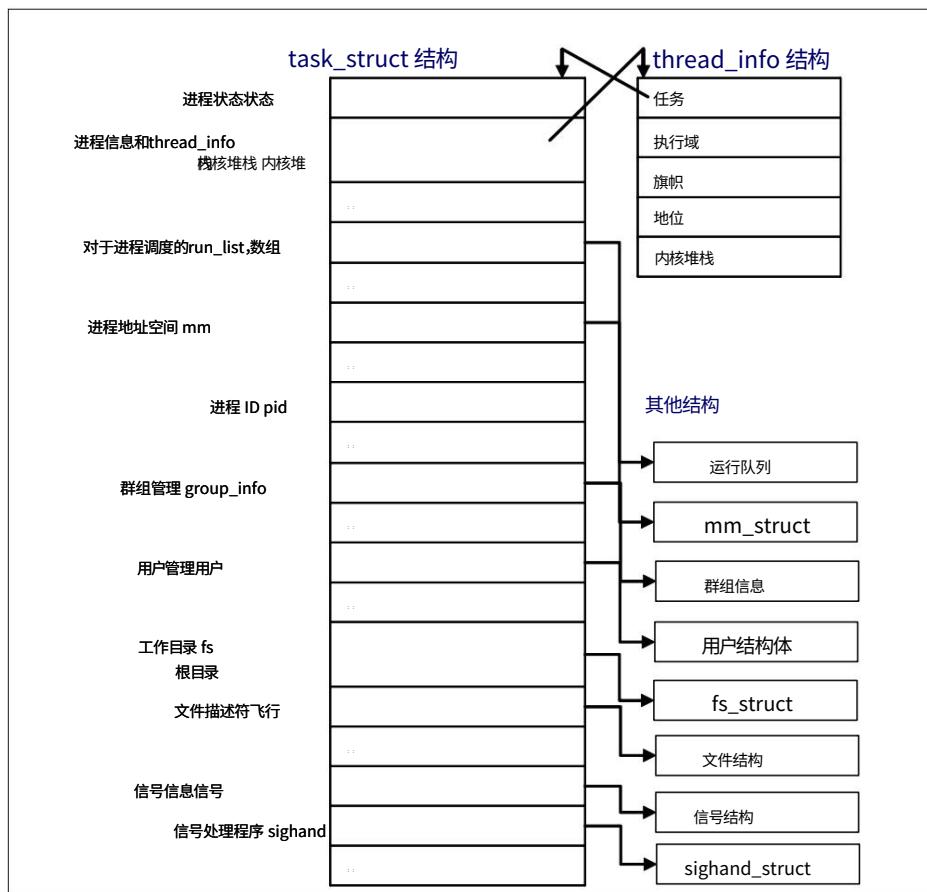


图1-2 task_struct 结构

1.1.2 进程的生命周期

每个进程都有自己的生命周期,例如创建、执行、终止和删除。
只要系统正常运行,这些阶段就会重复数百万次。因此,从性能角度来看,进程生命周期非常重要。

图1-3显示了进程的典型生命周期。

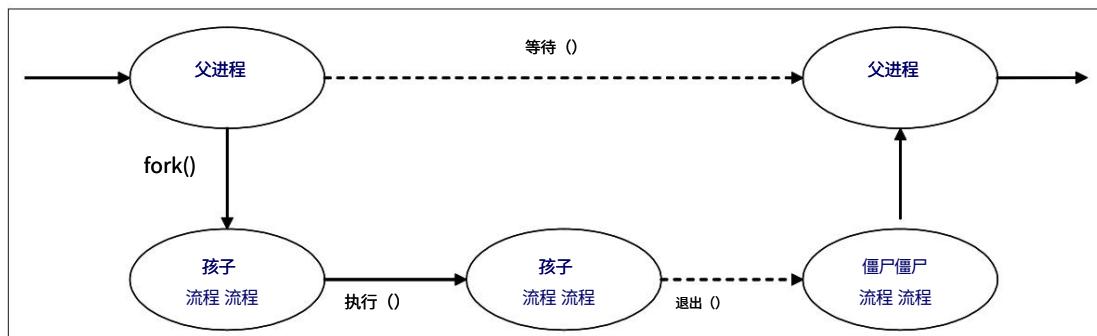


图1-3 典型进程的生命周期

当一个进程创建一个新进程时,创建进程 (父进程)会发出fork()系统调用。fork ()系统调用会获取新创建进程 (子进程)的进程描述符,并设置一个新的进程 ID。它会复制

将父进程的进程描述符复制到子进程的描述符中。此时,父进程的整个地址空间不会被复制;两个进程共享相同的地址空间。

`exec()`系统调用将新程序复制到子进程的地址空间。

由于两个进程共享相同的地址空间,写入新的程序数据会导致页面错误异常。此时,内核会将新的物理页面分配给子进程。

这种延迟操作称为“写时复制”。子进程通常执行自己的程序,而不是与父进程执行相同的程序。此操作避免了不必要的开销,因为复制整个地址空间是一个非常缓慢且低效的操作,会占用大量的处理器时间和资源。

程序执行完成后,子进程将通过`exit()`系统调用终止。`exit()`系统调用会释放进程的大部分数据结构,并通过发送信号通知父进程进程已终止。此时的进程被称为僵尸进程(请参阅第7页的“僵尸进程”)。

子进程只有在父进程通过`wait()`系统调用获知其子进程终止后才会被完全删除。一旦父进程收到子进程终止的通知,它将删除子进程的所有数据结构并释放进程描述符。

1.1.3 线程

线程是单个进程中生成的执行单元。它与同一进程中的其他线程并行运行。它们可以共享相同的资源,例如内存、地址空间、打开的文件等。它们可以访问同一组应用程序数据。线程也称为轻量级进程(LWP)。由于它们共享资源,因此每个线程不应同时更改其共享资源。互斥、锁定、序列化等功能的实现由用户应用程序负责。

从性能角度来看,创建线程比创建进程的开销更小,因为线程在创建时不需要复制资源。另一方面,进程和线程在调度算法方面具有相似的特性。内核以类似的方式处理它们。

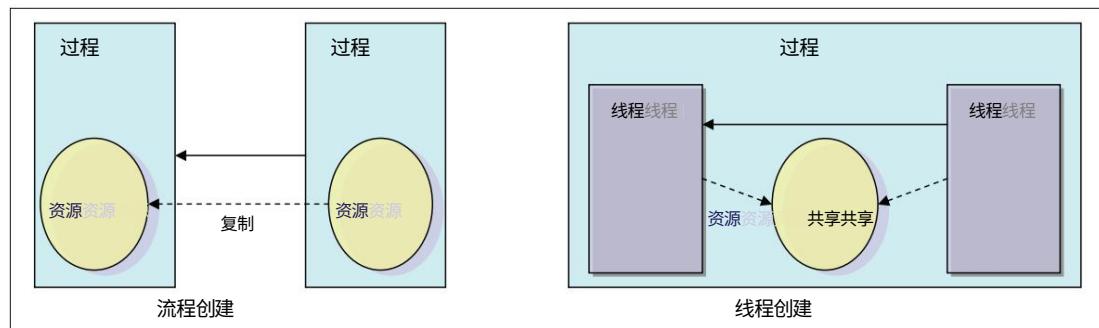


图1-4 进程与线程

在当前的Linux实现中,线程由可移植UNIX操作系统接口(POSIX)兼容库(pthread)支持。Linux操作系统中提供了多种线程实现。以下是一些广泛使用的实现。

Linux线程

LinuxThreads 自 Linux 内核 2.0 以来一直是默认的线程实现。LinuxThread 的一些实现与 POSIX 标准存在一些不兼容之处。原生 POSIX 线程库 (NPTL) 正在取代 LinuxThreads。企业 Linux 发行版的未来版本将不再支持 LinuxThreads。

原生 POSIX 线程库 (NPTL)

NPTL 最初由 Red Hat 开发。NPTL 更符合 POSIX 标准。它利用了内核 2.6 中的增强功能，例如新的 `clone()` 系统调用、信号处理实现等，比 LinuxThreads 具有更好的性能和可扩展性。

NPTL 与 LinuxThreads 存在一些不兼容性。依赖 LinuxThread 的应用程序可能无法与 NPTL 实现兼容。

下一代 POSIX 线程 (NGPT)

NGPT 是 IBM 开发的 POSIX 线程库版本。目前处于维护阶段，没有进一步的开发计划。

使用 `LD_ASSUME_KERNEL` 环境变量，您可以选择应用程序应该使用哪个线程库。

1.1.4 进程优先级和nice级别

进程优先级是一个数字，它决定了进程被 CPU 处理的顺序，由动态优先级和静态优先级决定。进程优先级越高，获得处理器运行许可的机会就越大。

内核会根据进程的行为和特性，使用启发式算法根据需要动态地调高或调低动态优先级。用户进程可以通过使用进程的 nice 级别间接更改静态优先级。静态优先级较高的进程将拥有更长的时间片（即进程在处理器上运行的时间）。

Linux 支持从 19（最低优先级）到 -20（最高优先级）的 nice 等级。默认值为 0。要将程序的 nice 等级更改为负数（以提高其优先级），需要登录或以 root 身份使用 `su` 命令。

1.1.5 上下文切换

在进程执行期间，正在运行的进程的信息存储在处理器的寄存器及其缓存中。为执行进程加载到寄存器中的数据集称为上下文。要切换进程，需要存储正在运行的进程的上下文，并将下一个正在运行的进程的上下文恢复到寄存器中。进程描述符和称为内核模式堆栈的区域用于存储上下文。此切换过程称为上下文切换。过多的上下文切换是不可取的，因为处理器每次都必须刷新其寄存器和缓存，为新进程腾出空间。这可能会导致性能问题。

图 1-5 说明了上下文切换的工作原理。

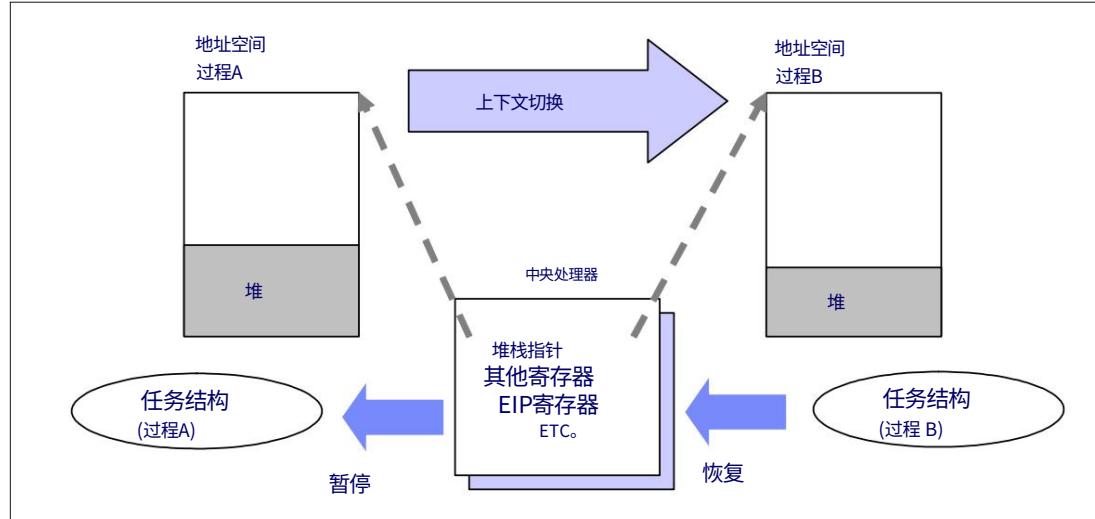


图1-5 上下文切换

1.1.6 中断处理

中断处理是最高优先级的任务之一。中断通常由 I/O 设备产生,例如网卡、键盘、磁盘控制器、串行适配器等。

中断处理程序会将事件（例如键盘输入、以太网帧到达等）通知 Linux 内核。它会通知内核中断进程执行并尽快执行中断处理,因为某些设备需要快速响应。这对于系统稳定性至关重要。当中断信号到达内核时,内核必须将当前执行的进程切换到新的进程来处理中断。这意味着中断会导致上下文切换,因此大量的中断可能会导致性能下降。

在 Linux 实现中,有两种类型的中断。硬中断用于需要响应的设备（磁盘 I/O 中断、网络适配器中断、键盘中断、鼠标中断）。软中断用于可延迟处理的任务（TCP/IP 操作、SCSI 协议操作等）。您可以在 /proc/interrupts 中查看与硬中断相关的信息。

在多处理器环境中,中断由每个处理器处理。将中断绑定到单个物理处理器可以提高系统性能。更多详细信息,请参阅第 108 页上的 4.4.2 节“中断处理的 CPU 亲和性”。

1.1.7 进程状态

每个进程都有自己的状态,显示进程中当前正在发生的事情。

流程状态在流程执行过程中发生变化。一些可能的状态如下：

任务运行

在此状态下,进程正在 CPU 上运行或在队列（运行队列）中等待运行。

任务停止

被某些信号（例如SIGINT、SIGSTOP）暂停的进程处于此状态。该进程正在等待诸如 SIGCONT 之类的信号恢复。

任务可中断

在此状态下,进程被暂停并等待某个条件满足。如果进程处于 TASK_INTERRUPTIBLE 状态并收到停止信号,则进程状态将改变,操作将被中断。TASK_INTERRUPTIBLE 进程的一个典型示例是等待键盘中断的进程。

任务不可中断

与 TASK_INTERRUPTIBLE 类似。处于 TASK_INTERRUPTIBLE 状态的进程可以被中断,而发送信号对处于 TASK_UNINTERRUPTIBLE 状态的进程没有任何作用。

TASK_UNINTERRUPTIBLE 状态的一个典型例子是等待磁盘 I/O 操作的进程。

任务僵尸

当一个进程通过exit()系统调用退出后,其父进程应该知道该进程已终止。在 TASK_ZOMBIE 状态下,进程正在等待其父进程收到通知以释放所有数据结构。

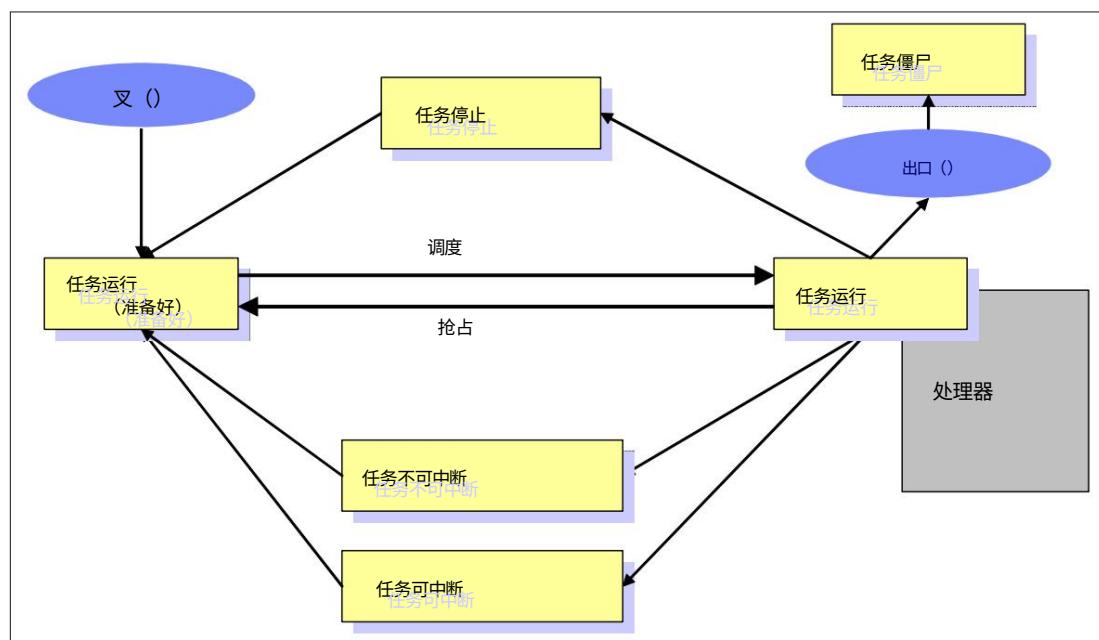


图1-6 进程状态

僵尸进程

当一个进程收到终止信号后,通常会需要一些时间来完成所有任务(例如关闭打开的文件),然后再自行结束。在这段通常很短的时间内,该进程处于僵尸状态。

进程完成所有这些关闭任务后,会向父进程报告即将终止。有时,僵尸进程无法自行终止,在这种情况下,它会显示状态为Z(僵尸)。

无法使用kill命令终止这样的进程,因为它已经被视为已死亡。如果无法清除僵尸进程,可以终止其父进程,这样僵尸进程也会随之消失。但是,如果父进程是 init 进程,则不应终止它。init 进程非常重要,因此可能需要重启才能清除僵尸进程。

1.1.8 进程内存段

进程使用其自己的内存区域来执行工作。具体工作内容会根据具体情况和进程使用情况而有所不同。进程可能具有不同的工作负载特征和不同的数据大小需求。进程必须处理各种大小的数据。为了满足这一需求,Linux 内核为每个进程使用动态内存分配机制。进程内存分配结构如图 1-7 所示。

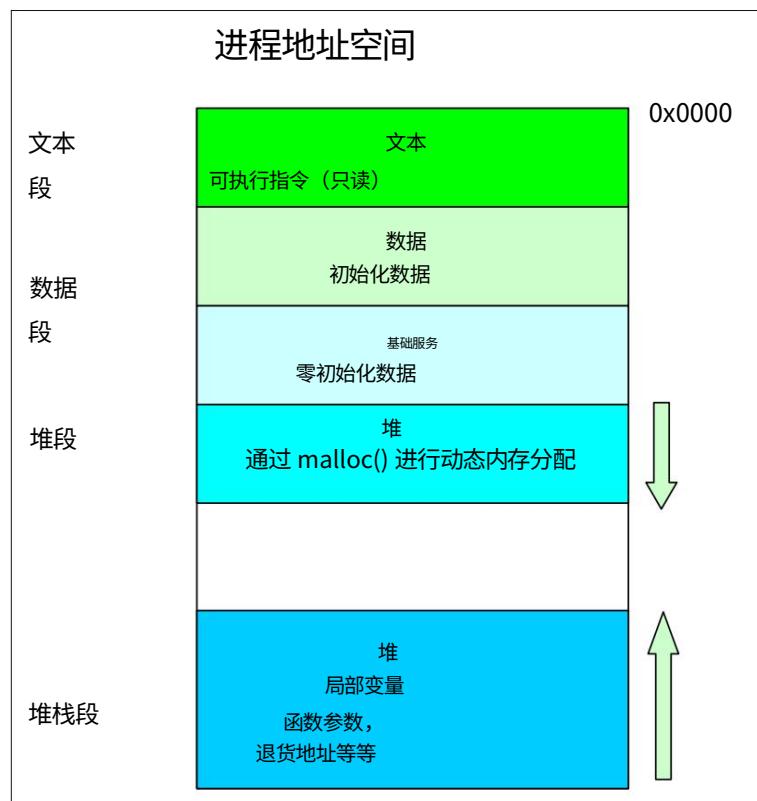


图1-7 进程地址空间

进程内存区域由这些段组成

文本段

存储可执行代码的区域。

数据段

数据段由这三个区域组成。

- 数据：存储静态变量等初始化数据的区域。
- BSS：存储零初始化数据的区域。数据被初始化为零。
- 堆： malloc() 根据需求分配动态内存的区域。
堆向更高的地址增长。

堆栈段

存储局部变量、函数参数以及函数返回地址的区域。堆栈向低地址方向增长。

您可以使用 pmap 命令显示用户进程地址空间的内存分配情况。您可以使用 ps 命令显示段的总大小。请参阅第 52 页 2.3.10 节 “pmap” 和第 44 页 2.3.4 节 “ps 和 pstree” 。

1.1.9 Linux CPU调度程序

任何计算机的基本功能,简而言之,就是计算。为了能够进行计算,必须有一种方法来管理计算资源(即处理器)和计算任务(也称为线程或进程)。得益于Ingo Molnar的杰出贡献,Linux内核采用了O(1)算法,而不是之前CPU调度程序所用的O(n)算法。O(1)指的是一种静态算法,这意味着无论进程数量多少,选择一个进程执行所需的时间都是恒定的。

新的调度程序无论进程数量或处理器数量如何,都具有极好的扩展性,并且对系统造成的开销很小。该算法使用两个进程优先级数组:

积极的
已到期

调度程序会根据进程的优先级和先前的阻塞率为其分配一个时间片,然后根据进程的优先级,将它们放入active数组中的一个进程列表中。当进程的时间片到期时,会为其分配一个新的时间片,并将其放入expired数组中。

当active数组中的所有进程都用完其时间片时,两个数组将交换位置,并重新启动算法。对于一般的交互式进程(而非实时进程),这会导致高优先级进程(通常具有较长的时间片)比低优先级进程获得更多的计算时间,但不会达到使低优先级进程完全饿死的程度。这种算法的优势在于,它极大地提高了Linux内核对企业级工作负载的可扩展性,这些工作负载通常包含大量线程或进程以及大量处理器。新的O(1)CPU调度器是为内核2.6设计的,但已移植到2.4内核系列。第9页上的图1-8展示了Linux CPU调度器的工作原理。

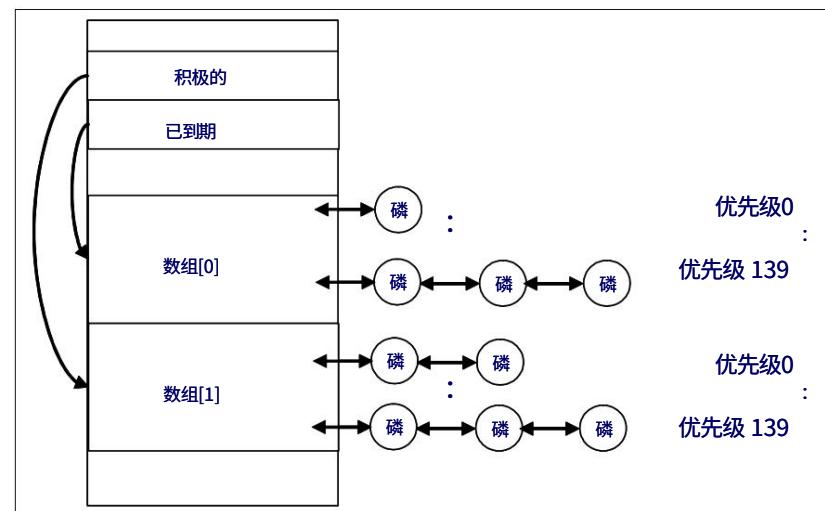


图1-8 Linux内核2.6 O(1)调度器

新调度程序的另一个显著优势是支持非统一内存架构(NUMA)和对称多线程处理器,例如英特尔®超线程技术。

改进的NUMA支持确保除非某个节点负担过重,否则NUMA节点之间不会发生负载平衡。此机制可确保NUMA系统中相对较慢的可扩展链路上的流量最小化。虽然调度器域组中处理器之间的负载平衡会在每个调度器周期内进行,但

仅当该节点过载并要求负载平衡时,才会发生跨调度程序域的工作负载。

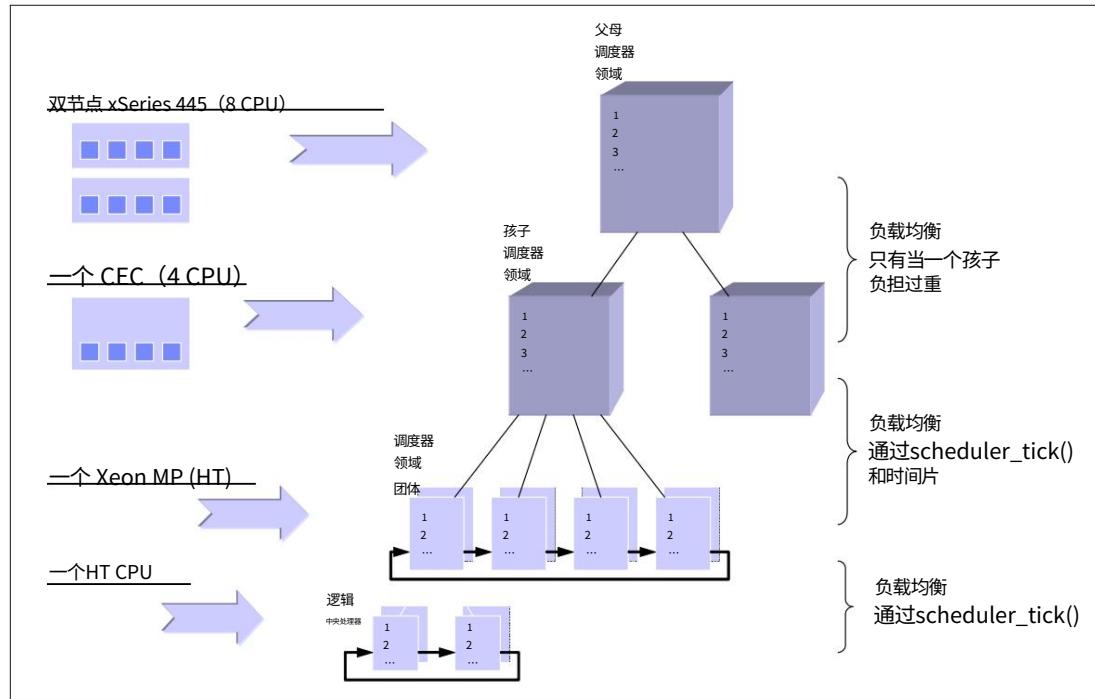


图 1-9 基于 8 路 NUMA 的系统上的 O(1) CPU 调度程序架构
启用超线程

1.2 Linux 内存架构

为了执行一个进程,Linux 内核会为请求的进程分配一部分内存区域。进程使用该内存区域作为工作空间并执行所需的工作。这类似于您拥有自己的办公桌,然后使用桌面来存放纸张、文档和备忘录以完成工作。不同之处在于,内核必须以更动态的方式分配空间。运行的进程数量有时会达到数万个,而内存量通常有限。因此,Linux 内核必须高效地处理内存。在本节中,我们将描述 Linux 内存架构、地址布局以及 Linux 如何高效地管理内存空间。

1.2.1 物理内存和虚拟内存

如今,我们面临着 32 位系统和 64 位系统的选择。对于企业级客户端来说,最重要的区别之一是虚拟内存寻址 4 GB 以上的可能性。从性能的角度来看,了解 Linux 内核在 32 位和 64 位系统上如何将物理内存映射到虚拟内存是很有意义的。

正如第 11 页的图 1-10 所示,Linux 内核在 32 位和 64 位系统中寻址内存的方式存在明显差异。详细探讨物理到虚拟的映射超出了本文的讨论范围,因此我们重点介绍 Linux 内存架构中的一些细节。

在 32 位架构 (例如 IA-32) 上,Linux 内核只能直接寻址第一个 GB 的物理内存 (考虑到保留范围,为 896 MB)。高于

所谓的 ZONE_NORMAL 必须映射到较低的 1 GB。这种映射对应用程序完全透明，但是在 ZONE_HIGHMEM 中分配内存页会导致轻微的性能下降。

另一方面，在 64 位架构（例如 x86-64（也包括 x64））中，ZONE_NORMAL 会一直扩展到 64 GB，在 IA-64 系统中则扩展到 128 GB。由此可见，使用 64 位架构可以消除将内存页从 ZONE_HIGHMEM 映射到 ZONE_NORMAL 的开销。

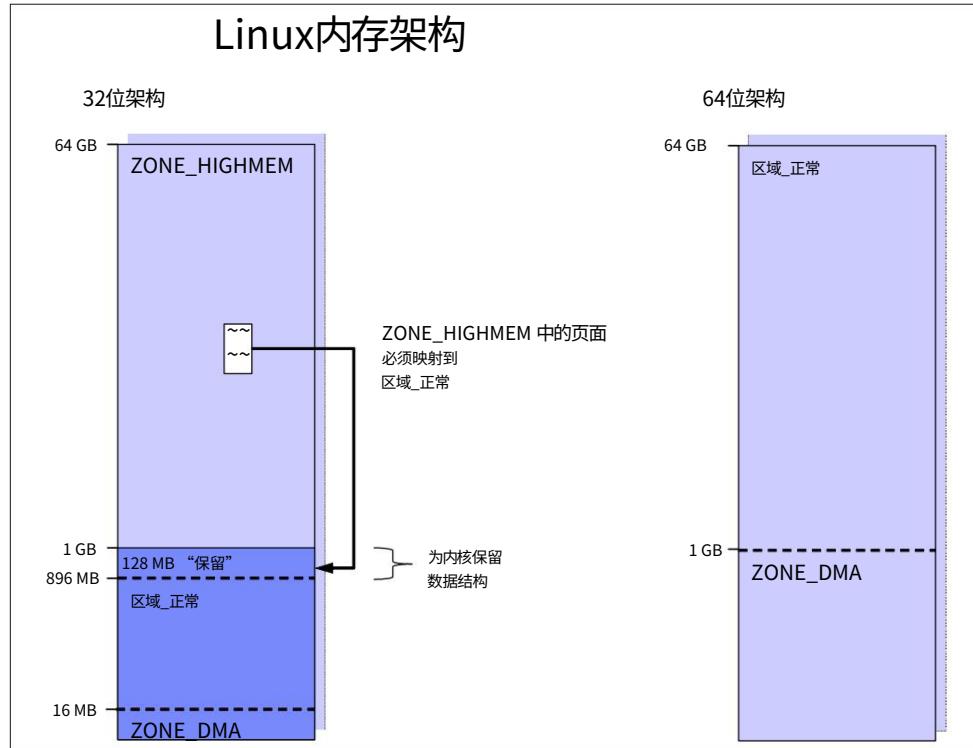


图1-10 32位和64位系统的Linux内核内存布局

虚拟内存寻址布局

图 1-11 显示了 32 位和 64 位架构的 Linux 虚拟寻址布局。

在 32 位架构上，单个进程可以访问的最大地址空间为 4GB。

这是源自 32 位虚拟寻址的限制。在标准实现中，虚拟地址空间被划分为 3 GB 的用户空间和 1 GB 的内核空间。此外，还有一些变体，例如 4 GB/4 GB 的寻址布局实现。

另一方面，在 x86_64 和 ia64 等 64 位架构上，不存在这样的限制。

每个单独的进程都可以受益于广阔而巨大的地址空间。

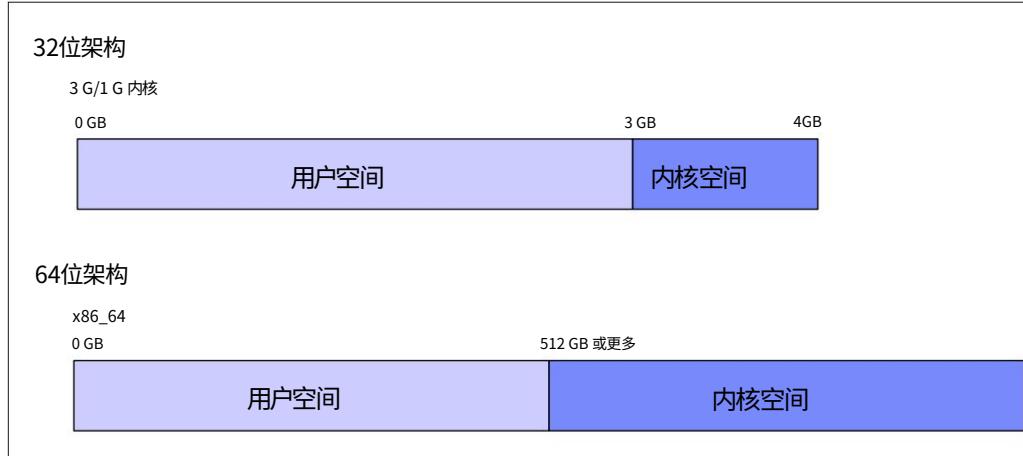


图1-11 32位和64位架构的虚拟内存寻址布局

1.2.2 虚拟内存管理器

操作系统的物理内存架构通常对应用程序和用户隐藏,因为操作系统会将任何内存映射到虚拟内存中。如果我们想要了解 Linux 操作系统内部的调优可能性,就必须了解 Linux 如何处理虚拟内存。如第 10 页 1.2.1 节 “物理内存和虚拟内存”中所述,应用程序不分配物理内存,而是向 Linux 内核请求一定大小的内存映射,并作为交换接收虚拟内存中的映射。如图 1-12 所示,虚拟内存不一定非要映射到物理内存中。如果应用程序分配了大量内存,则其中一些内存可能会映射到磁盘子系统上的交换文件。

图 1-12 显示,应用程序通常不会直接写入磁盘子系统,而是写入缓存或缓冲区。然后,当 pdflush 内核线程有时间或文件大小超出缓冲区缓存时,它会将缓存/缓冲区中的数据刷新到磁盘。请参阅第 22 页的“刷新脏缓冲区”。

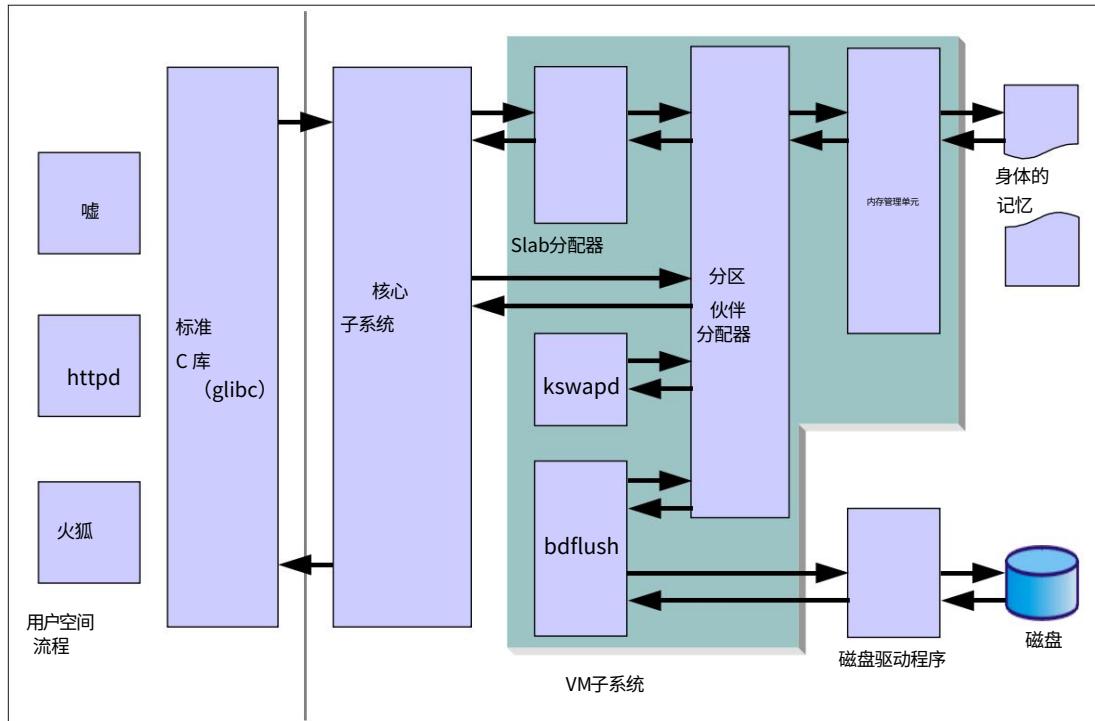


图1-12 Linux虚拟内存管理器

Linux 内核管理磁盘缓存的方式与 Linux 内核处理物理磁盘子系统写入的方式密切相关。其他操作系统仅分配特定比例的内存作为磁盘缓存，而 Linux 处理内存资源的效率则高得多。虚拟内存管理器的默认配置会将所有可用的空闲内存空间分配为磁盘缓存。因此，高效的 Linux 系统拥有数 GB 的内存，但只有 20 MB 可用，这种情况并不罕见。

同样，Linux 也能非常高效地处理交换空间。交换空间的使用并不代表内存瓶颈，而是证明了 Linux 处理系统资源的效率。更多详细信息，请参阅第 14 页的“页框回收”。

页框分配

页是物理内存（页框）或虚拟内存中一组连续的线性地址。Linux 内核以此页为单位处理内存。页通常大小为 4 KB。当一个进程请求一定数量的页面时，如果有可用页面，Linux 内核会立即将其分配给该进程。否则，必须从其他进程或页面缓存中获取页面。内核知道有多少可用的内存页面以及它们的位置。

伙伴系统

Linux 内核使用一种称为伙伴系统的机制来维护其空闲页面。伙伴系统负责维护空闲页面，并尝试为页面分配请求分配页面。它会尽力保持内存区域的连续性。如果小页面被随意分散，可能会导致内存碎片化，并且将大量页面分配到连续区域中会变得更加困难。这可能会导致内存使用效率低下和性能下降。

图 1-13 说明了伙伴系统如何分配页面。

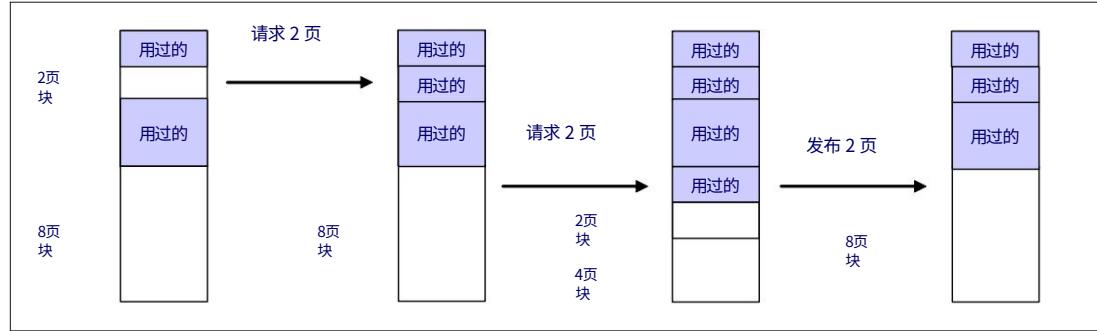


图 1-13 伙伴系统

当页面分配失败时,将激活页面回收机制。请参阅第 14 页的“页面帧回收”。

您可以通过 /proc/buddyinfo 查找有关伙伴系统的信息。有关详细信息,请参阅第 47 页上的“区域中的内存使用情况”。

页框回收

如果进程请求映射一定数量的页面时,没有可用的页面,Linux 内核会尝试释放某些页面 (这些页面之前使用过,但现在不再使用,并且根据某些原则仍被标记为活动页面),并将内存分配给新进程,从而为新请求获取页面。此过程称为页面回收。kswapd 内核线程和 try_to_free_page () 内核函数负责页面回收。

kswapd 通常处于任务可中断状态,但当区域中的可用页面未达到阈值时,伙伴系统会调用它。它会尝试根据最近最少使用(LRU)原则从活动页面中查找要移除的候选页面。最近最少使用的页面应首先释放。活动列表和非活动列表用于维护候选页面。kswapd 会扫描活动列表的一部分,检查页面的最近使用情况,并将最近未使用的页面放入非活动列表。您可以使用 vmstat -a 命令查看有多少内存被视为活动和非活动。有关详细信息,请参阅第 42 页上的 2.3.2 节“vmstat”。

kswapd 还遵循另一项原则。页面主要用于两个目的:页面缓存和进程地址空间。页面缓存是映射到磁盘文件的页面。属于进程地址空间的页面 (称为匿名内存,因为它未映射到任何文件,也没有名称) 用于堆和堆栈。请参阅第 8 页上的 1.1.8 “进程内存段”。当 kswapd 回收页面时,它宁愿缩小页面缓存,也不愿将进程拥有的页面换出 (或交换出)。

页面调出和交换出：“页面调出”和“交换出”这两个词有时会让人混淆。“页面调出”的意思是将部分页面 (整个地址空间的一部分) 调入交换空间,而“交换出”的意思是将整个地址空间调入交换空间。

它们有时可以互换使用。

回收的页面缓存和进程地址空间的很大一部分可能取决于使用场景,并且会影响性能。您可以使用 /proc/sys/vm/swappiness 来控制此行为。有关调整的详细信息,请参阅第 109 页上的 4.5.1 “设置内核交换和 pdflush 行为”。

交换

如前所述,当发生页面回收时,非活动列表中属于进程地址空间的候选页面可能会被调出。使用交换空间本身并不成问题。在其他操作系统中,交换空间只不过是应对主内存过度分配的一种保障,而 Linux 则更高效地利用交换空间。如第 13 页图 1-12 所示,虚拟内存由物理内存和磁盘子系统(或交换分区)组成。如果 Linux 中的虚拟内存管理器发现某个内存页面已被分配但长时间未使用,它会将该内存页面移至交换空间。

您经常会看到一些守护进程(例如 getty),它们会在系统启动时启动,但几乎不会使用。看来,将昂贵的主内存中这些页面释放出来,并将其移至交换分区会更高效。这正是 Linux 处理交换分区的方式,因此,如果您发现交换分区已满 50%,也无需惊慌。

交换空间被使用这一事实并不表示存在内存瓶颈;相反,它证明了 Linux 处理系统资源的效率。

1.3 Linux 文件系统

Linux 作为开源操作系统的一大优势在于它为用户提供了多种受支持的文件系统。现代 Linux 内核几乎可以支持计算机系统使用的所有文件系统,从基本的 FAT 支持到高性能文件系统,例如日志文件系统(JFS)。但是,由于 Ext2、Ext3 和 ReiserFS 是大多数 Linux 发行版都支持的原生 Linux 文件系统(ReiserFS 仅在 Novell SUSE Linux 上提供商业支持),因此我们将重点介绍它们的特性,并仅概述其他常用的 Linux 文件系统。

有关文件系统和磁盘子系统的更多信息,请参阅第 112 页上的 4.6 “调整磁盘子系统”。

1.3.1 虚拟文件系统

虚拟文件系统(VFS)是一个抽象接口层,位于用户进程和各种 Linux 文件系统实现之间。VFS 提供了通用的对象模型(例如 i 节点、文件对象、页面缓存、目录项等)以及访问文件系统对象的方法。它向用户进程隐藏了各个文件系统实现之间的差异。借助 VFS,用户进程无需了解要使用哪个文件系统,也无需了解应该为每个文件系统发出哪个系统调用。

第 16 页的图 1-14 说明了 VFS 的概念。

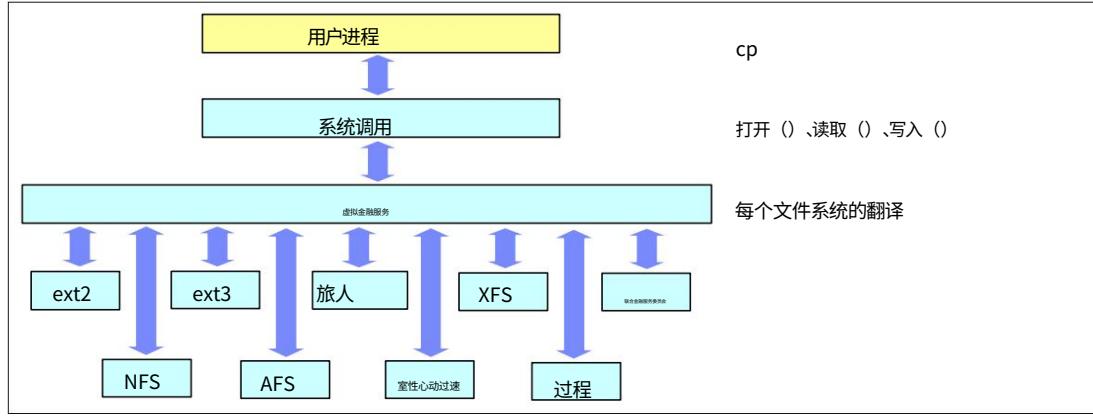


图1-14 VFS概念

1.3.2 日志

在非日志文件系统中，当对文件系统执行写入操作时，Linux 内核会首先更改文件系统元数据，然后再写入实际的用户数据。此操作有时会导致数据完整性丢失的可能性更高。如果系统在执行写入文件系统元数据的操作时因某种原因突然崩溃，则文件系统的一致性可能会被破坏。fsck会通过检查所有元数据来修复不一致性，并在下次重启时恢复一致性。但是，当系统容量较大时，此操作需要很长时间才能完成。在此过程中，系统无法运行。

日志文件系统通过将待更改的数据写入实际文件系统之前，先将其写入称为日志区域的区域来解决此问题。日志区域可以位于文件系统内部，也可以位于文件系统外部。写入日志区域的数据称为日志。它包含对文件系统元数据的更改以及实际文件数据（如果支持）的更改。

由于日记功能会在将实际用户数据写入文件系统之前写入日志，因此与无日记功能的文件系统相比，它可能会导致性能开销。为了维持更高的数据一致性而牺牲的性能开销取决于在写入用户数据之前写入磁盘的信息量。我们将在第 18 页的 1.3.4 节“Ext3”中讨论此主题。

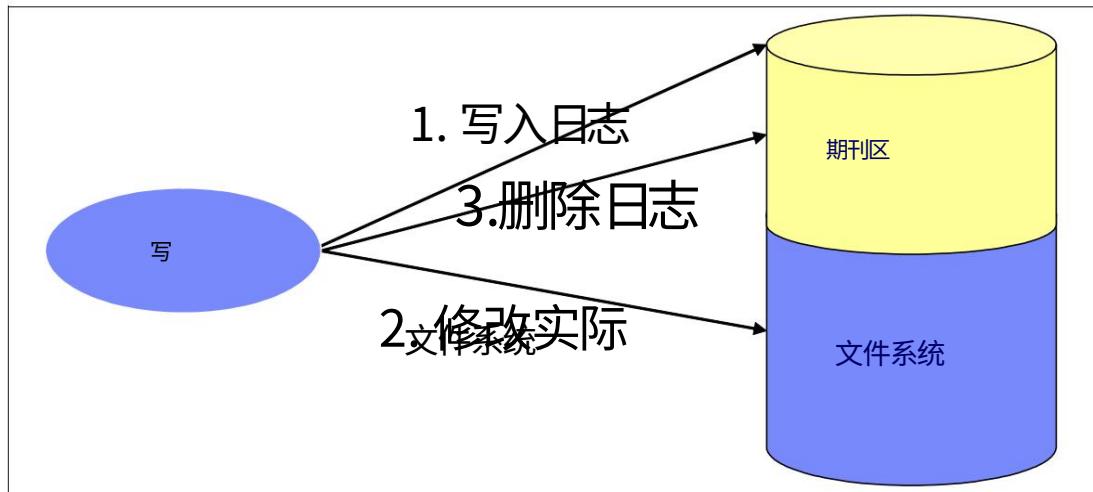


图1-15 日志概念

1.3.3 Ext2

扩展 2 文件系统是扩展 3 文件系统的前身。它是一种快速、简单的文件系统,与大多数其他现有文件系统不同,它不具备日志功能。

图 1-16 显示了 Ext2 文件系统的数据结构。文件系统以引导扇区开始,随后是块组。将整个文件系统拆分为多个小的块组有助于提升性能,因为 i 节点表和保存用户数据的数据块可以在磁盘盘片上更靠近放置,从而减少寻道时间。块组由以下项目组成:

超级块	文件系统的信息存储在这里。超级块的精确副本位于每个块组的顶部。
块组描述符块组的信息存储在这里。	
i节点位图	用于免费i节点管理
i节点表	i 节点表存储在这里。每个文件都有一个对应的 i 节点表,其中包含文件的元数据,例如文件模式、uid、gid、atime、ctime、mtime、ctime 以及指向数据块的指针。
数据块	实际用户数据的存储位置

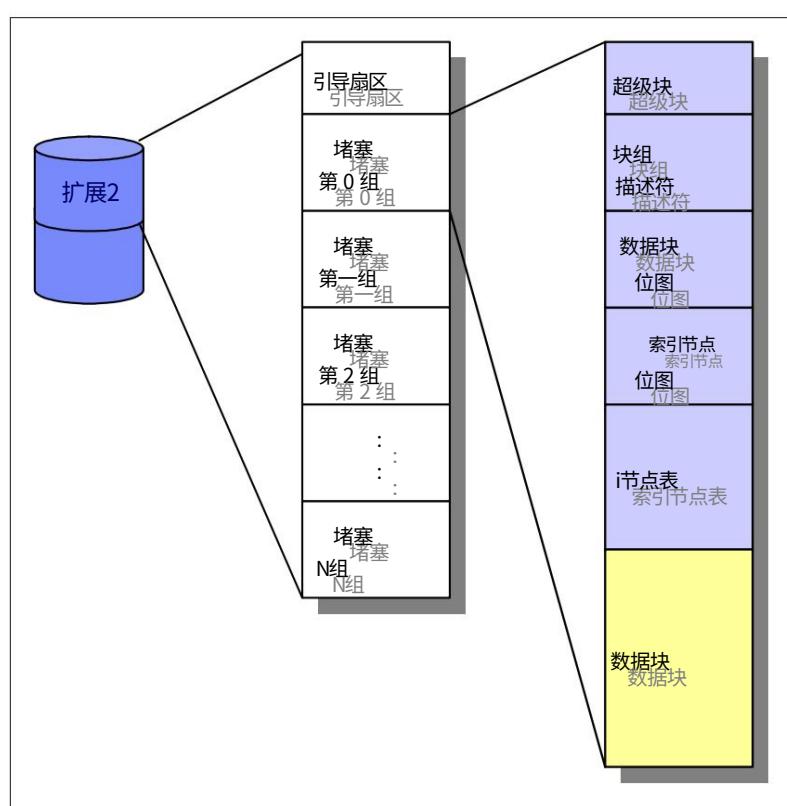


图1-16 Ext2文件系统数据结构

为了查找包含文件的数据块,内核首先搜索文件的 i 节点。当进程发出打开 /var/log/messages 的请求时,内核会解析文件路径,并搜索 / (根目录) 的目录条目,该条目包含其自身 (根目录) 下的文件和目录的信息。然后,内核可以找到 /var 的 i 节点,并查看 /var 的目录条目。该条目也包含其自身下的文件和目录的信息。内核以相同的方式查找文件,直到找到文件的 i 节点。Linux

内核使用文件对象缓存（例如目录项缓存或i节点缓存）来加速查找相应的i节点。

一旦 Linux 内核知道了文件的 i 节点，它就会尝试到达实际的用户数据块。

正如我们之前所述，i 节点包含指向数据块的指针。通过引用它，内核可以找到数据块。对于大型文件，Ext2 实现了对数据块的直接/间接引用。

图 1-17 说明了其工作原理。

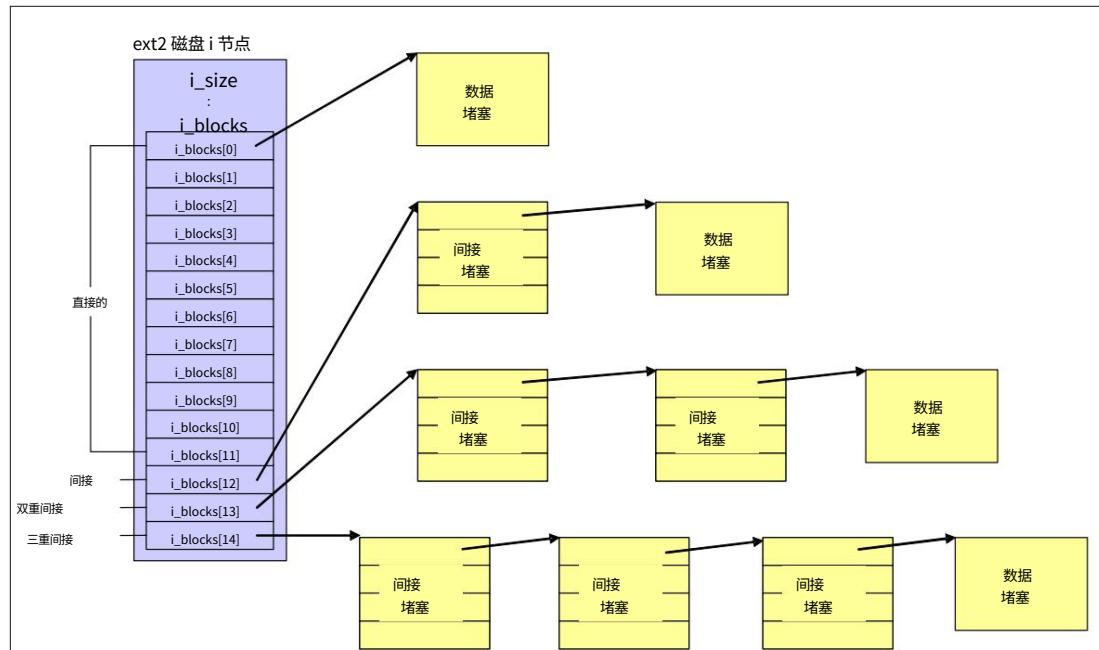


图1-17 Ext2文件系统直接/间接引用数据块

不同文件系统的文件系统结构和文件访问操作各不相同，这赋予了每个文件系统不同的特性。

1.3.4 Ext3

当前的企业 Linux 发行版支持扩展 3 文件系统。这是广泛使用的扩展 2 文件系统的更新版本。虽然其基本结构与 Ext2 文件系统相似，但主要区别在于对日志功能的支持。此文件系统的亮点包括：

可用性:Ext3 始终以一致的方式将数据写入磁盘，因此在发生非正常关机（意外断电或系统崩溃）的情况下，服务器不必花时间检查数据的一致性，从而将系统恢复时间从数小时缩短至数秒。

数据完整性:通过在mount命令上指定日志模式 `data=journal`，所有数据（包括文件数据和元数据）都会被记录下来。

速度:通过指定日志模式 `data=writeback`，您可以根据业务需求，选择速度与完整性的平衡点。这在同步写入频繁的环境中尤为明显。

灵活性:从现有的 Ext2 文件系统升级非常简单，无需重新格式化。只需执行 `tune2fs` 命令并修改 `/etc/fstab` 文件，即可轻松将 Ext2 文件系统升级到 Ext3 文件系统。另请注意，Ext3 文件系统可以作为 Ext2 文件系统挂载，并禁用日志功能。许多第三方供应商的产品都

操作 Ext3 文件系统的能力。例如,PartitionMagic 可以处理 Ext3 分区的修改。

日记模式

Ext3 支持三种日志模式。

杂志

此日志选项通过同时记录文件数据和元数据来提供最高级别的数据一致性。但它的性能开销也更高。

已订购

在此模式下,仅写入元数据。但是,保证首先写入文件数据。

这是默认设置。

写回

此日志选项以牺牲数据一致性为代价,提供最快的数据访问速度。由于元数据仍在记录中,因此可以保证数据的一致性。

但是,没有对实际文件数据进行特殊处理,这可能会导致系统崩溃后旧数据出现在文件中。

1.3.5 ReiserFS

ReiserFS 是一种快速的日志文件系统,具有优化的磁盘空间利用率和快速的崩溃恢复功能。ReiserFS 的开发在很大程度上得到了 Novell 的帮助。ReiserFS 仅在 Novell SUSE Linux 上提供商业支持。

1.3.6 日志文件系统

日志文件系统 (JFS) 是一个完整的 64 位文件系统,可以支持超大文件和分区。JFS 最初由 IBM 为 AIX® 开发,现在已采用通用公共许可证 (GPL) 提供。对于高性能计算 (HPC) 或数据库环境中常见的超大分区和文件大小,JFS 是一个理想的文件系统。如果您想了解更多关于 JFS 的信息,请参阅:

<http://jfs.sourceforge.net>

注意:在 Novell SUSE Linux Enterprise Server 10 中,JFS 不再作为新文件系统受支持。

1.3.7 XFS

扩展文件系统 (XFS) 是由 Silicon Graphics Incorporated 开发的高性能日志文件系统,最初用于其 IRIX 系列系统。它的特性与 IBM 的 JFS 类似,也支持非常大的文件和分区大小。

因此,使用场景与 JFS 非常相似。

1.4 磁盘I/O子系统

在处理器解码并执行指令之前,数据必须从磁盘盘片上的扇区一路检索到处理器缓存及其寄存器。执行结果才能写回磁盘。

我们将研究 Linux 磁盘 I/O 子系统,以便更好地了解对系统性能有重大影响的组件。

1.4.1 I/O子系统架构

图1-18显示了I/O子系统架构的基本概念

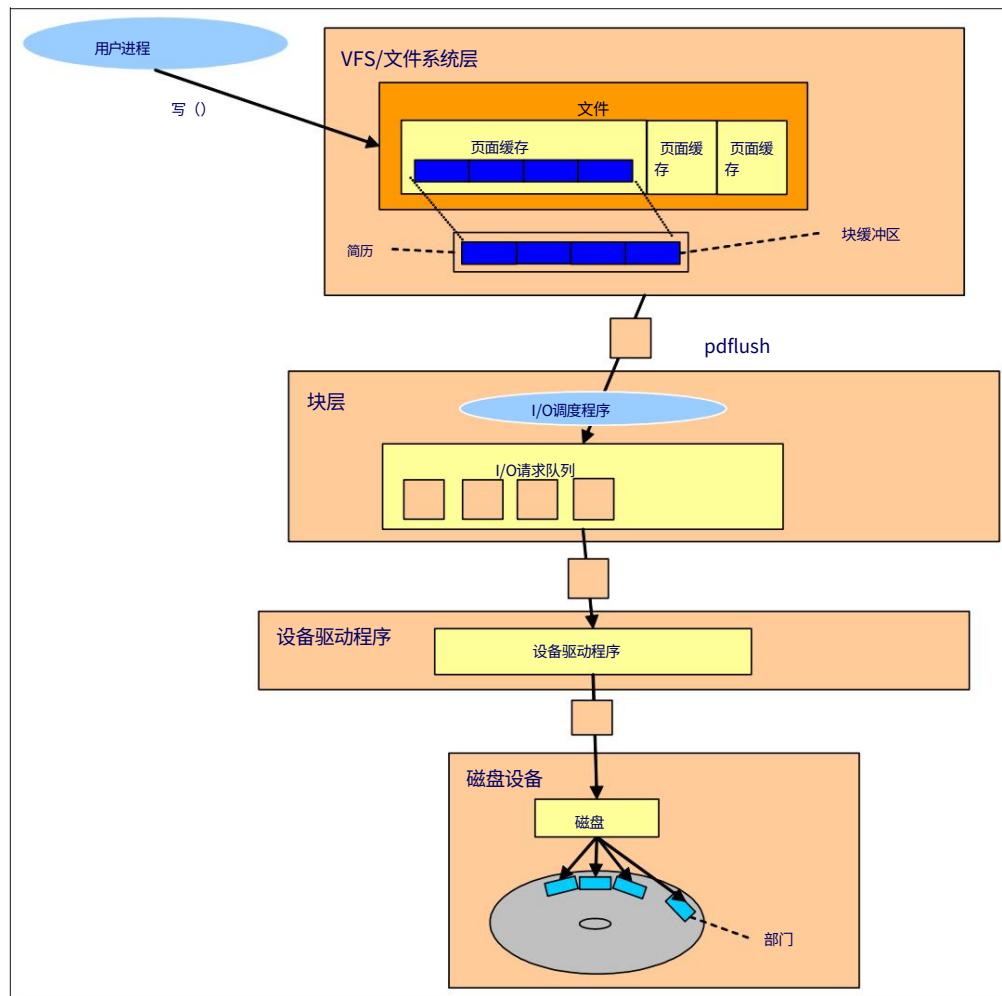


图1-18 I/O子系统架构

为了快速概览 I/O 子系统的整体操作,我们将以将数据写入磁盘为例。以下序列概述了执行磁盘写入操作时发生的基本操作。假设文件数据位于磁盘盘片上的扇区中,已被读取,并且位于页面缓存中。

1. 一个进程通过write()系统调用请求写入文件。
2. 内核更新映射到该文件的页缓存。
3. pdflush内核线程负责将页面缓存刷新到磁盘。
4. 文件系统层将各个块缓冲区组合成一个bio struct (参考1.4.3 “块层”),并向块设备层提交写请求。
5. 块设备层从上层获取请求并执行I/O 提升操作并将请求放入I/O请求队列。

6. 设备驱动程序（例如 SCSI 或其他设备特定驱动程序）将负责写入手术。
7. 磁盘设备固件执行硬件操作，如寻道、旋转和数据转移到盘片上的扇区。

1.4.2 缓存

在过去的20年中，处理器性能的提升已经超过了计算机系统中其他组件（例如处理器缓存、总线、RAM、磁盘等）的性能提升。内存和磁盘访问速度的降低限制了系统的整体性能，因此处理器速度的提升并不能提升系统性能。缓存机制通过将常用数据缓存在更快的内存中来解决此问题，从而减少了访问速度较慢内存的可能性。当前的计算机系统几乎在所有I/O组件中都采用了这种技术，例如硬盘驱动器缓存、磁盘控制器缓存、文件系统缓存以及每个应用程序处理的缓存等等。

内存层次

图 1-19 展示了内存层次结构的概念。由于 CPU 寄存器和磁盘的访问速度差异很大，CPU 会花费更多时间等待来自慢速磁盘设备的数据，从而大大降低了快速 CPU 的优势。

内存分层结构通过在 CPU 和磁盘之间放置 L1 缓存、L2 缓存、RAM 和其他一些缓存来减少这种不匹配。这使得进程访问速度较慢的内存和磁盘的机会更少。靠近处理器的内存速度更快，体积更小。

该技术还可以利用引用局部性原理。在速度更快的内存上，缓存命中率越高，数据访问速度就越快。

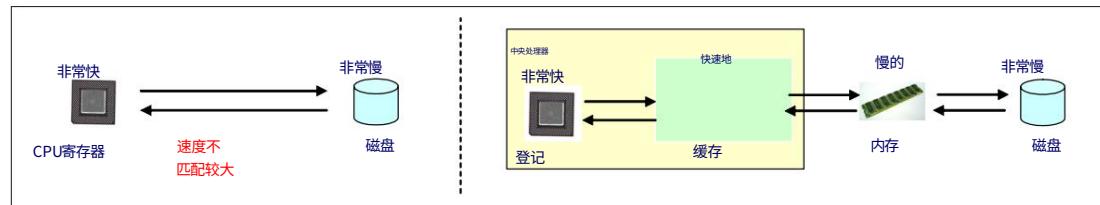


图1-19 内存层次结构

参考地点

正如我们之前在“内存层次结构”中所述，实现更高的缓存命中率是提升性能的关键。为了实现更高的缓存命中率，我们使用了一种称为“引用局部性”的技术。该技术基于以下原则：

最近使用的数据在不久的将来被使用的概率很高（时间局部性）。

靠近已使用数据的数据被使用的概率很高（空间局部性）。

第 22 页的图 1-20 说明了这一原理。

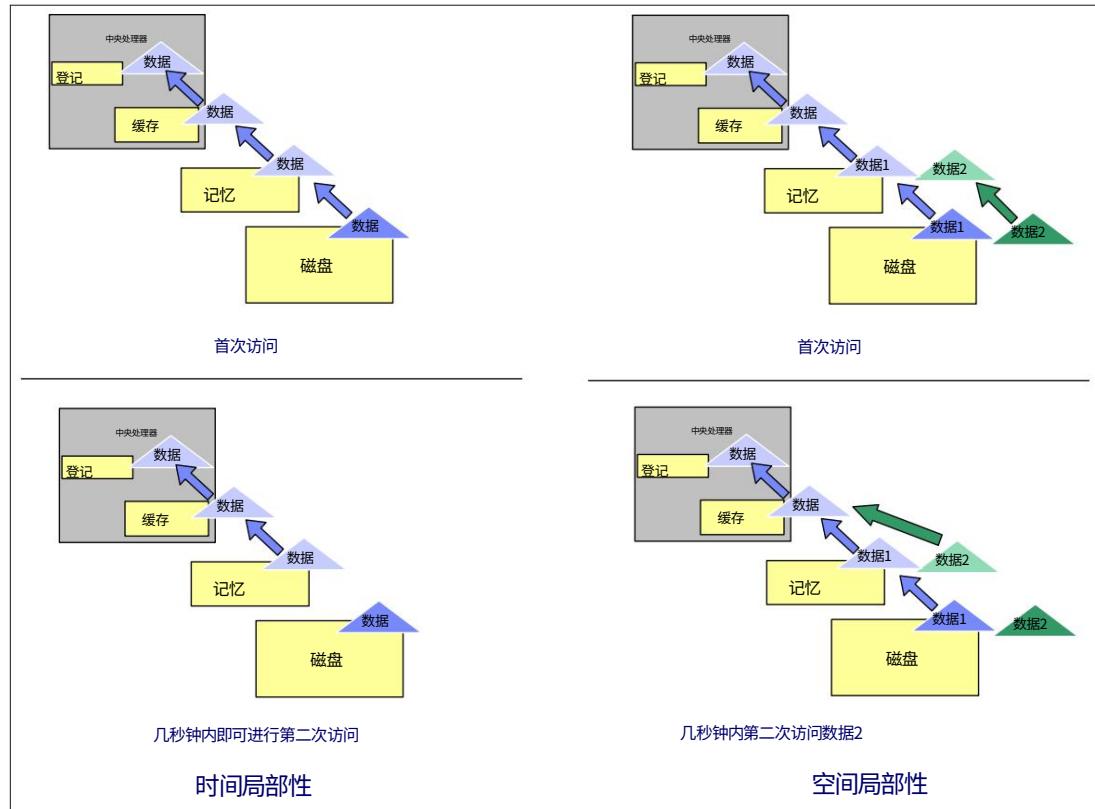


图 1-20 引用局部性

Linux 在许多组件中都使用了这个原则,例如页面缓存、文件对象缓存 (i 节点缓存、目录条目缓存等)、预读缓冲区等。

刷新脏缓冲区

当进程从磁盘读取数据时,数据会被复制到内存中。该进程和其他进程可以从缓存在内存中的数据副本中检索相同的数据。当进程尝试更改数据时,该进程会首先更改内存中的数据。此时,磁盘上的数据和内存中的数据并不完全相同,内存中的数据被称为脏缓冲区。脏缓冲区应尽快与磁盘上的数据同步,否则如果发生突然崩溃,内存中的数据可能会丢失。

脏缓冲区的同步过程称为刷新 (flush)。在 Linux 内核 2.6 实现中, pdflush 内核线程负责将数据刷新到磁盘。刷新会定期发生 (kupdate), 并且当内存中脏缓冲区的比例超过某个阈值时 (bdflush) 也会触发。该阈值可在 /proc/sys/vm/dirty_background_ratio 文件中配置。有关更多信息,请参阅第 109 页上的“4.5.1 设置内核交换和 pdflush 行为”。

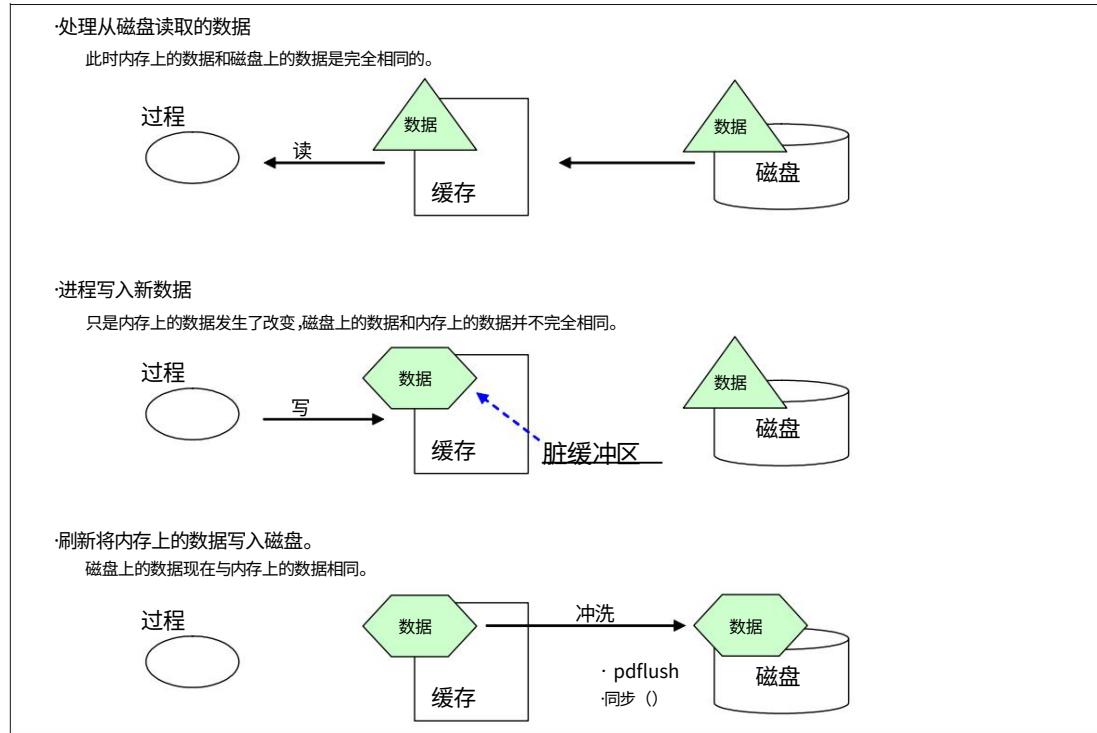


图 1-21 刷新脏缓冲区

1.4.3 区块层

块层处理与块设备操作相关的所有活动（参见第 20 页的图 1-18）。块层中的关键数据结构是 bio 结构体。bio 结构体是文件系统层和块层之间的接口。

当执行写入操作时,文件系统层会尝试写入由块缓冲区组成的页缓存。文件系统层将连续的块组合在一起,形成一个 bio 结构,然后将 bio 发送给块层。(参见第 20 页的图 1-18)

块层处理 bio 请求,并将这些请求链接到一个称为 I/O 请求队列的队列中。此链接操作称为 I/O 电梯。在 Linux 内核 2.6 实现中,有四种类型的 I/O 电梯算法可用。它们是:

块大小

块大小是驱动器可读取或写入的最小数据量,它会直接影响服务器的性能。一般来说,如果您的服务器处理大量小文件,那么较小的块大小会更高效。如果您的服务器专用于处理大文件,那么较大的块大小可能会提高性能。现有文件系统上的块大小无法动态更改。只有重新格式化才能修改当前的块大小。

输入/输出电梯

Linux 内核 2.6 采用了全新的 I/O 电梯模型。Linux 内核 2.4 使用的是单个通用 I/O 电梯模型,而 2.6 内核则提供了四种电梯模型供用户选择。由于 Linux 操作系统可用于执行各种任务,因此 I/O 设备和工作负载特性都会发生显著变化。笔记本电脑的 I/O 需求可能与拥有 10,000 个用户的数据库系统不同。为了适应这种情况,我们提供了四种 I/O 电梯模型。

预期

预期 I/O 提升器是基于块设备只有一个物理寻道头（例如单个 SATA 驱动器）的假设而创建的。预期提升器使用下文将详细描述的截止期限机制以及预期启发式算法。顾名思义，预期 I/O 提升器“预期” I/O 并尝试将其以单个更大的数据流写入磁盘，而不是多次非常小的随机磁盘访问。预期启发式算法可能会导致写入 I/O 延迟。它显然是针对通用系统（例如普通个人计算机）上的高吞吐量进行调整的。在内核版本 2.6.18 之前，预期提升器是标准的 I/O 调度程序。但是，大多数企业 Linux 发行版默认使用 CFQ 提升器。

完全公平排队（CFQ）

CFQ 提升器通过维护每个进程的 I/O 队列来为进程实现 QoS（服务质量）策略。CFQ 提升器非常适合具有大量竞争进程的大型多用户系统。它会积极尝试避免进程资源匮乏，并具有低延迟的特点。从内核版本 2.6.18 开始，改进的 CFQ 提升器是默认的 I/O 调度程序。

根据系统设置和工作负载特性，CFQ 调度器可能会降低单个主应用程序的速度，例如，一个采用公平性导向算法的大型数据库。默认配置基于相互竞争的进程组来处理公平性问题。例如，单个数据库和所有通过页面缓存的写入操作（所有 pdflush 实例都位于一个 pgroup 中）会被 CFQ 视为单个应用程序，该应用程序可能会与许多后台进程竞争。在这种情况下，尝试使用 I/O 调度器子配置和/或 Deadline 调度器会很有帮助。

最后期限

截止期限提升器是一种循环提升器（轮询），其截止期限算法可提供 I/O 子系统的近乎实时的行为。截止期限提升器在保持良好磁盘吞吐量的同时，提供了出色的请求延迟。截止期限算法的实现确保不会发生进程饥饿。

空操作

NOOP 代表无操作（No Operation），其名称解释了它的大部分功能。NOOP 提升器简单精简。它是一个简单的 FIFO 队列，不执行任何数据排序。NOOP 只是合并相邻的数据请求，因此它对磁盘 I/O 的处理器开销非常小。NOOP 提升器假设块设备要么有自己的提升算法（例如 SCSI 的 TCQ），要么没有寻道延迟（例如闪存卡）。

注意：在 Linux 内核版本 2.6.18 中，现在可以根据每个磁盘子系统选择 I/O 提升器，而不再需要在每个系统级别进行设置。

1.4.4 I/O 设备驱动程序

Linux 内核使用设备驱动程序来控制设备。设备驱动程序通常是一个单独的内核模块，为每个设备（或设备组）提供，以使设备可供 Linux 操作系统使用。设备驱动程序加载后，将作为 Linux 内核的一部分运行，并完全控制设备。本文将介绍 SCSI 设备驱动程序。

SCSI

小型计算机系统接口（SCSI）是最常用的 I/O 设备技术，尤其是在企业服务器环境中。在 Linux 内核实现中，

SCSI 设备由设备驱动程序模块控制。它们由以下类型的模块组成。

上层驱动程序 :sd_mod、sr_mod (SCSI-CDROM)、st (SCSI 磁带)、sq (SCSI 通用设备) 等等。

提供支持多种类型的 SCSI 设备 (如 SCSI CD-ROM、SCSI 磁带等) 的功能。

中层驱动程序 :scsi_mod

实现 SCSI 协议和常见的 SCSI 功能。

低级驱动程序

提供对每个设备的底层访问。底层驱动程序基本上是针对特定硬件设备的，并为每个设备提供。例如，用于 IBM ServeRAID™ 控制器的 ips、用于 Qlogic HBA 的 qla2300、用于 LSI Logic SCSI 控制器的 mptscsih 等等。

伪驱动程序 :ide-scsi

用于 IDE-SCSI 仿真。

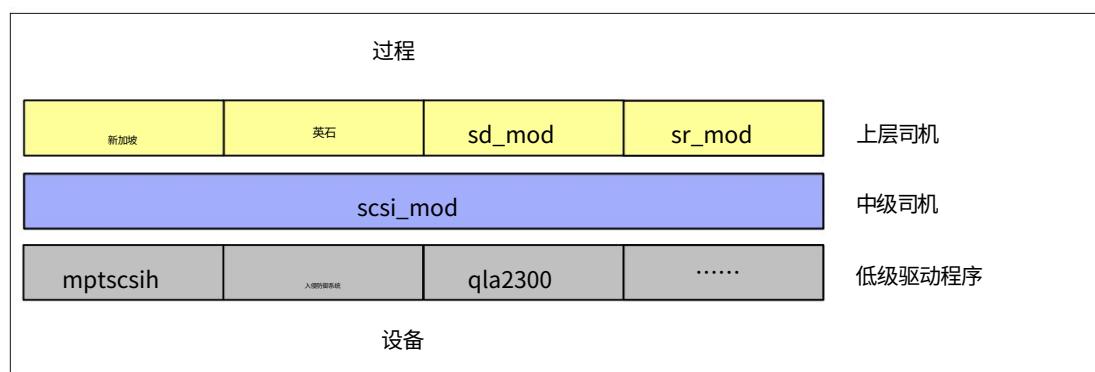


图1-22 SCSI驱动程序结构

如果为设备实现了特定功能，则应在设备固件和底层设备驱动程序中实现。支持的功能取决于您使用的硬件和设备驱动程序的版本。设备本身也应支持所需的功能。特定功能通常通过设备驱动程序参数进行调整。您可以尝试在 /etc/modules.conf 中进行一些性能调整。有关调整提示和技巧，请参阅设备和设备驱动程序文档。

1.4.5 RAID和存储系统

存储系统和 RAID 类型的选择和配置也是影响系统性能的重要因素。Linux 支持软件 RAID，但本文不涉及该主题的详细介绍。我们在第 113 页的 4.6.1 节“安装 Linux 前的硬件注意事项”中包含了一些调优注意事项。

有关可用 IBM 存储解决方案的更多详细信息，请参阅：

调优 IBM System x 服务器性能,SG24-5287

IBM 系统存储解决方案手册,SG24-5250

存储区域网络简介,SG24-5470

1.5 网络子系统

从性能角度来看，网络子系统是另一个重要的子系统。

网络操作会与 Linux 以外的许多组件交互，例如交换机、路由器、网关、PC 客户端等等。虽然这些组件可能不受 Linux 控制，但它们对整体性能有很大影响。请记住，您必须与网络系统上的工作人员密切合作。

这里我们主要关注 Linux 如何处理网络操作。

1.5.1 组网实现

TCP/IP 协议具有与 OSI 分层模型类似的分层结构。Linux 内核网络实现也采用了类似的方法。图 1-23 展示了分层的 Linux TCP/IP 协议栈，并提供了 TCP/IP 通信的概览。

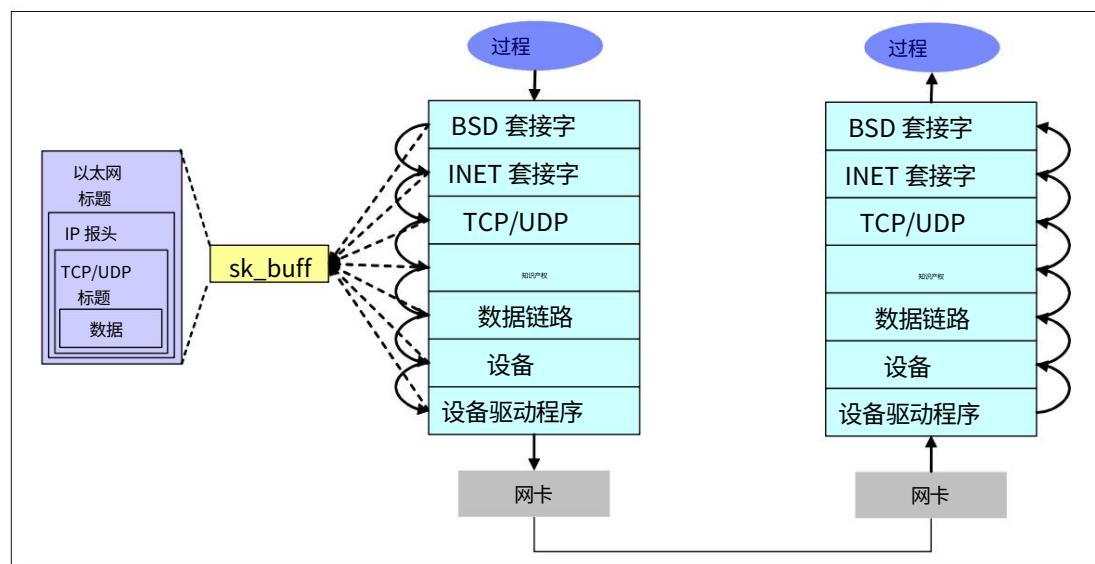


图1-23 网络分层结构及组网运行概况

与许多 UNIX 系统一样，Linux 使用套接字接口进行 TCP/IP 网络操作。

套接字为用户应用程序提供接口。我们将了解网络数据传输过程中基本操作的流程。

1. 当应用程序向其对等主机发送数据时，该应用程序会创建其数据。
2. 应用程序打开socket，并通过socket接口写入数据。
3. 套接字缓冲区用于处理传输的数据。套接字缓冲区具有引用数据并穿过各个层。
4. 在每一层中，执行适当的操作，例如解析报头、添加和修改报头、校验和、路由操作、分片等。

当套接字缓冲区向下传递到各个层级时，数据本身不会在层级之间复制。由于在不同层级之间复制实际数据效率不高，因此内核只需更改套接字缓冲区中的引用并将其传递到下一层，即可避免不必要的开销。

5. 最后，数据从网络接口卡输出到网络上。

6. 以太网帧到达对等主机的网络接口。

7. 如果 MAC 地址与
接口卡的 MAC 地址。
8. 网络接口卡最终将数据包移入套接字缓冲区并在 CPU 上发出硬中断。
9. 然后,CPU 处理该数据包,并将其向上移动,直至到达
(例如)Apache 等应用程序的 TCP 端口。

套接字缓冲区

如前所述,内核使用缓冲区来发送和接收数据。图 1-24 显示了可用于网络的可配置缓冲区。它们可以通过 /proc/sys/net 中的文件进行调整。

```
/proc/sys/net/core/rmem_max
/proc/sys/net/core/rmem_default
/proc/sys/net/core/wmem_max
/proc/sys/net/core/wmem_default
/proc/sys/net/ipv4/tcp_mem
/proc/sys/net/ipv4/tcp_rmem
/proc/sys/net/ipv4/tcp_wmem
```

有时它可能会影响网络性能。我们将在 4.7.4 节 “增加网络缓冲区”中详细介绍。

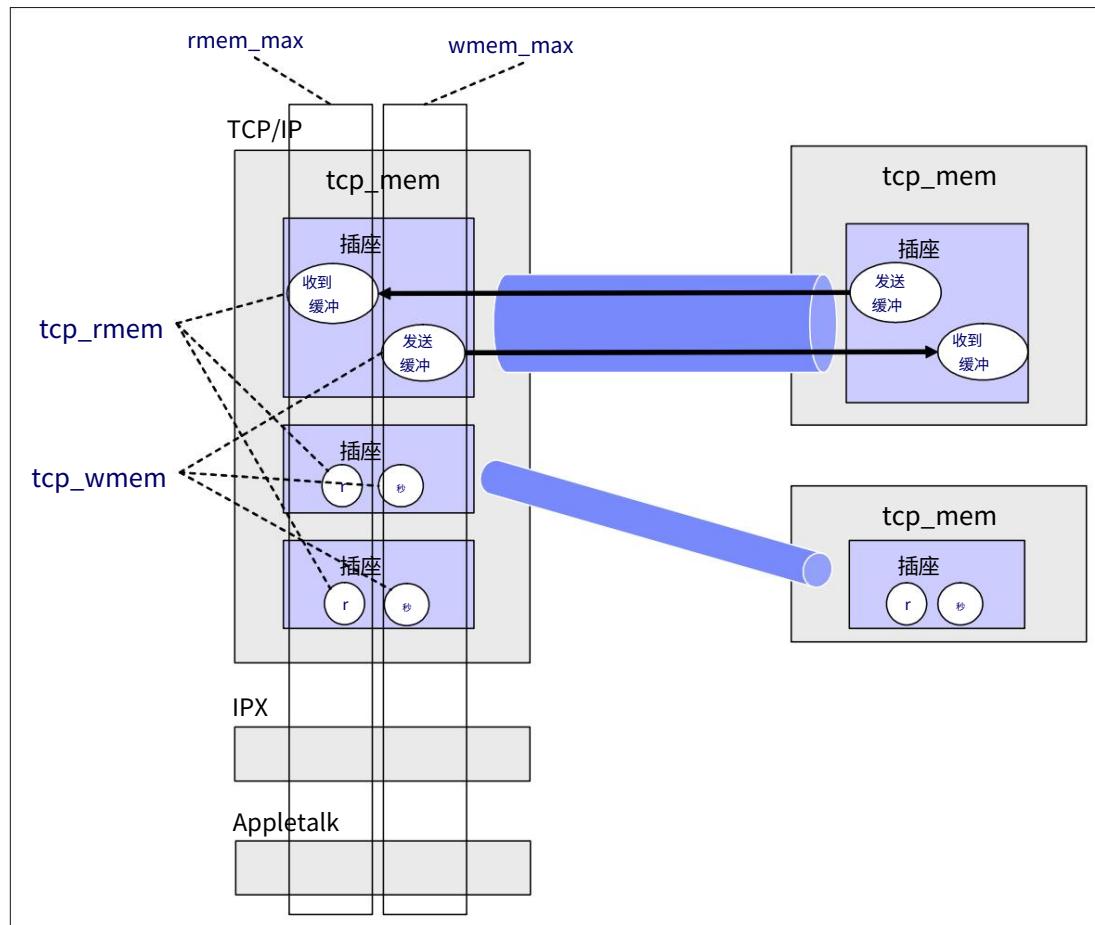


图1-24 套接字缓冲区内存分配

网络 API (NAPI)

随着新网络 API (NAPI) 的引入,网络子系统经历了一些变化。Linux 中网络堆栈的标准实现更注重可靠性和低延迟,而非低开销和高吞吐量。虽然这些特性在创建防火墙时非常有用,但大多数企业应用程序 (例如文件和打印或数据库)的运行速度会比 Windows® 下的类似安装慢。

在处理网络数据包的传统方法中,如图 1-25 中的蓝色箭头所示,网络接口卡最终将数据包移动到操作系统内核的网络缓冲区中,并在 CPU 上发出硬中断。

这只是处理网络数据包过程的简化视图,但它揭示了这种方法本身的缺点之一。每当一个具有匹配 MAC 地址的以太网帧到达接口时,就会发生硬中断。每当 CPU 必须处理硬中断时,它必须停止正在处理的任何任务并处理中断,从而导致上下文切换以及相关的处理器缓存刷新。虽然您可能认为如果只有少量数据包到达接口,这不会造成问题,但千兆以太网和现代应用程序每秒可以创建数千个数据包,从而导致大量的中断和上下文切换。

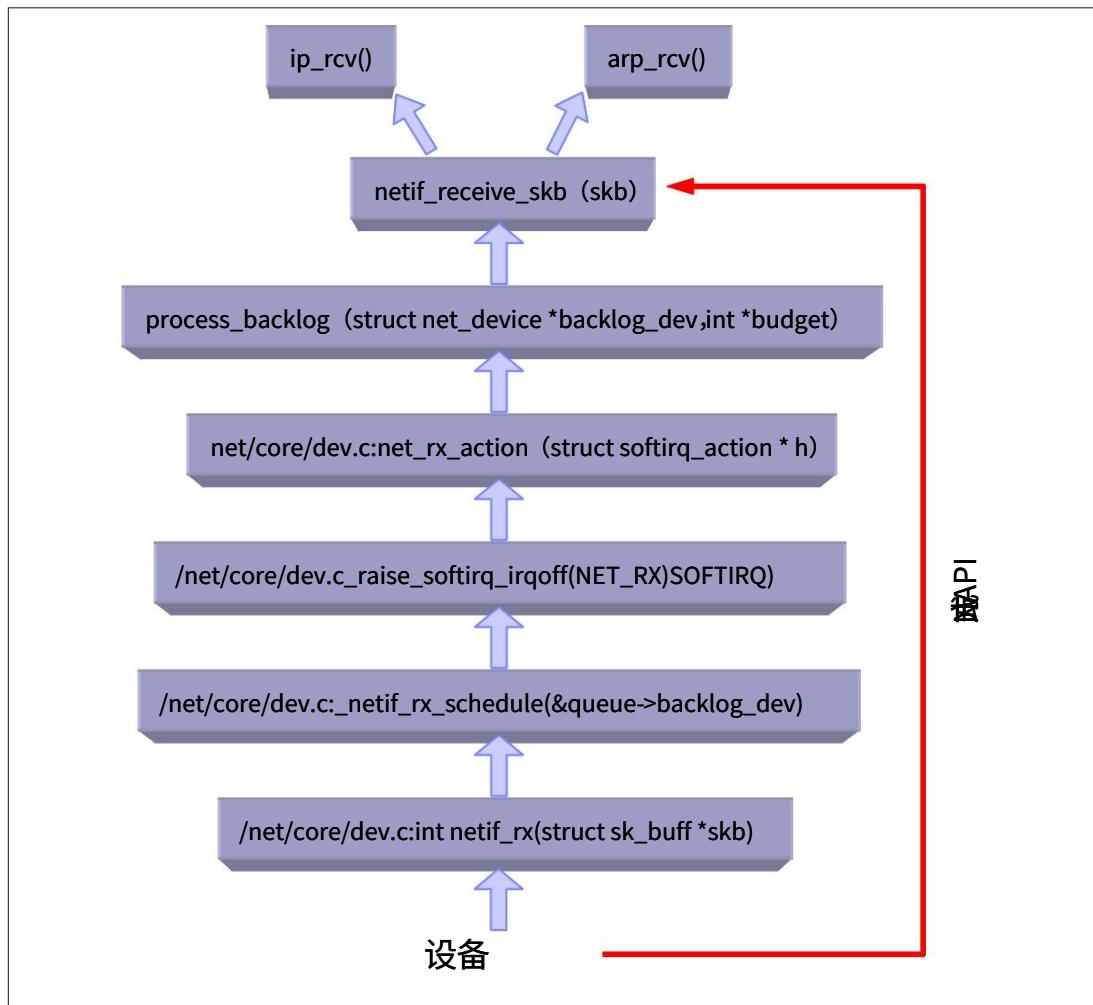


图 1-25 Linux 网络栈

因此,引入了 NAPI 来应对处理网络流量带来的开销。对于第一个数据包,NAPI 的工作原理与传统实现类似,它会为第一个数据包发出中断。但处理完第一个数据包后,接口将进入轮询模式。只要网络接口的 DMA 环形缓冲区中还有数据包,就不会触发新的中断,从而有效地减少了上下文切换及其相关开销。如果处理完最后一个数据包,环形缓冲区被清空,网卡将再次回到中断模式。NAPI 还具有通过创建可由多个处理器处理的软中断来提高多处理器可扩展性的优势。虽然 NAPI 对于大多数企业级多处理器系统来说将是一个巨大的改进,但它需要支持 NAPI 的驱动程序。此外,NAPI 还具有很大的调优空间,我们将在本文的调优部分进行探讨。

Netfilter

Linux 内核内置了高级防火墙功能。该功能由Netfilter模块提供。您可以使用iptables实用程序来操作和配置 Netfilter。

总体来说,Netfilter 提供以下功能。

数据包过滤:如果数据包与规则匹配,Netfilter 会接受或拒绝该数据包,或根据定义的规则采取适当的措施。

地址转换:如果数据包与规则匹配,Netfilter 会改变数据包以满足地址转换要求。

可以使用以下属性定义匹配过滤器。

网络接口

IP 地址、IP 地址范围、子网

协议

ICMP 类型

港口

TCP 标志

状态(请参阅第 30 页上的“连接跟踪”)

图 1-26 概述了数据包如何遍历 Netfilter 链,这些链是按顺序在每个点应用的已定义规则的列表。

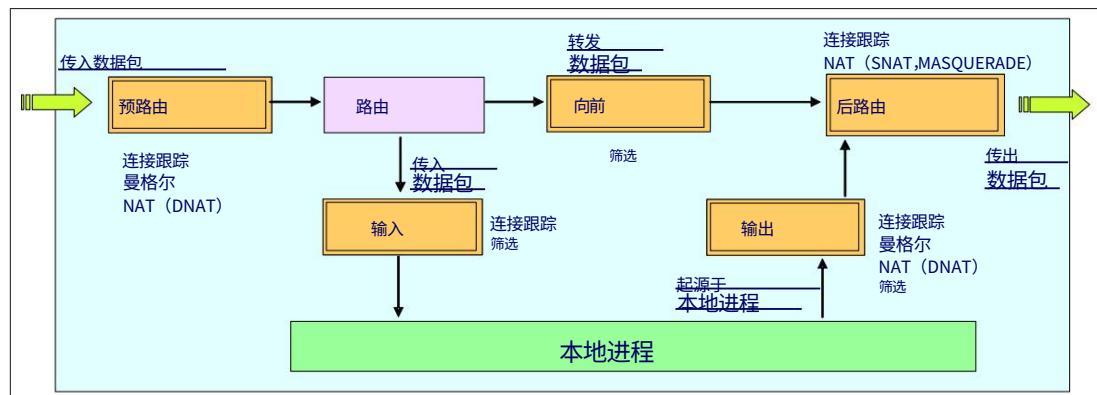


图 1-26 Netfilter 报文流

如果数据包与规则匹配,Netfilter 将采取适当的操作。该操作称为目标。一些可能的目标包括:

接受：	接受数据包并让其通过。
降低：	默默地丢弃该数据包。
拒绝：	通过发回诸如 ICMP 端口不可达之类的数据包来丢弃该数据包。TCP 重置为原始主机。

LOG:记录匹配的数据包。
MASQUERADE、SNAT、DNAT、REDIRECT:地址转换

连接跟踪

为了实现更复杂的防火墙功能,Netfilter 使用了连接跟踪机制来跟踪所有网络流量的状态。Netfilter 使用 TCP 连接状态（请参阅第 30 页的“连接建立”）和其他网络属性（例如 IP 地址、端口、协议、序列号、确认号、ICMP 类型等），将每个数据包分为以下四种状态。

新:数据包尝试建立新连接
ESTABLISHED:数据包通过已建立的连接
RELATED:与先前数据包相关的数据包
INVALID:由于格式错误或无效,数据包处于未知状态

此外,Netfilter 可以使用单独的模块,通过分析协议特定的属性和操作来执行更详细的连接跟踪。例如,FTP、NetBIOS、TFTP、IRC 等都有连接跟踪模块。

1.5.2 TCP/IP

TCP/IP 多年来一直是默认的网络协议。Linux 的 TCP/IP 实现与其标准基本兼容。为了更好地进行性能调优,您应该熟悉基本的 TCP/IP 网络知识。

有关更多详细信息,请参阅 TCP/IP 教程和技术概述,GG24-3376。

建立连接

在传输应用程序数据之前,客户端和服务器之间需要建立连接。连接建立过程称为 TCP/IP 三次握手。

第 31 页的图 1-27 概述了基本连接建立和终止过程。

1. 客户端向对等服务器发送 SYN 数据包（设置了 SYN 标志的数据包）以请求联系。
2. 服务器收到数据包并发回SYN+ACK数据包。
3. 然后客户端向对端发送ACK数据包,完成连接建立。

一旦连接建立,应用程序数据就可以通过该连接进行传输。当所有数据传输完毕后,连接关闭过程开始。

1. 客户端向服务器发送FIN报文,启动连接终止流程。
2. 服务器返回FIN确认,然后将FIN包发送给如果没有数据要发送给客户端。
3. 客户端向服务器发送ACK数据包,完成连接终止。

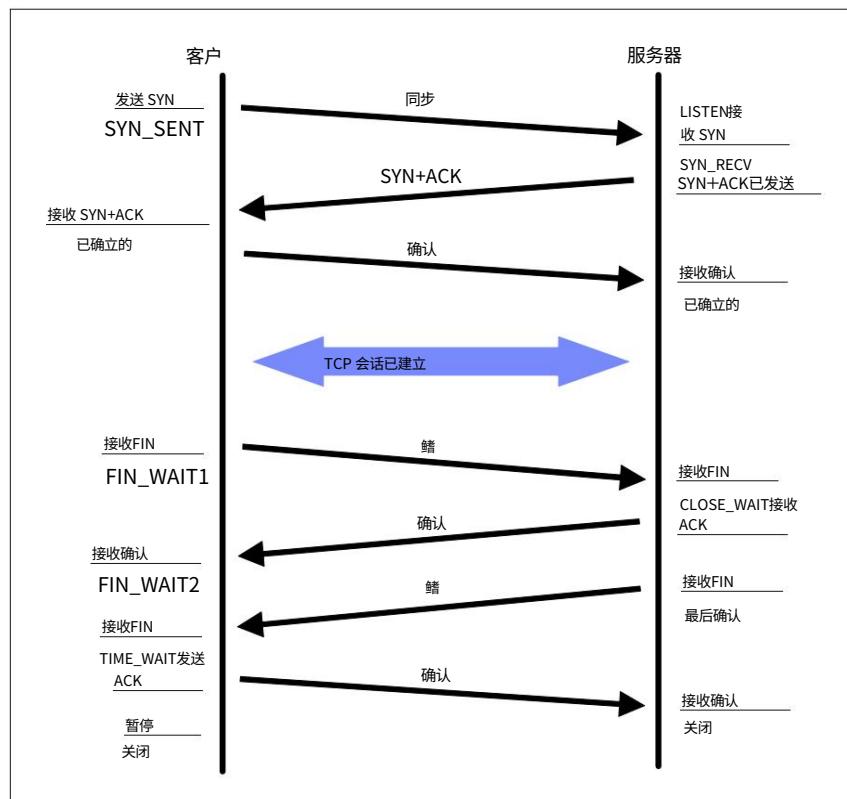


图1-27 TCP三次握手

连接状态在会话期间会发生变化。第 32 页的图 1-28 显示了 TCP/IP 连接状态图。

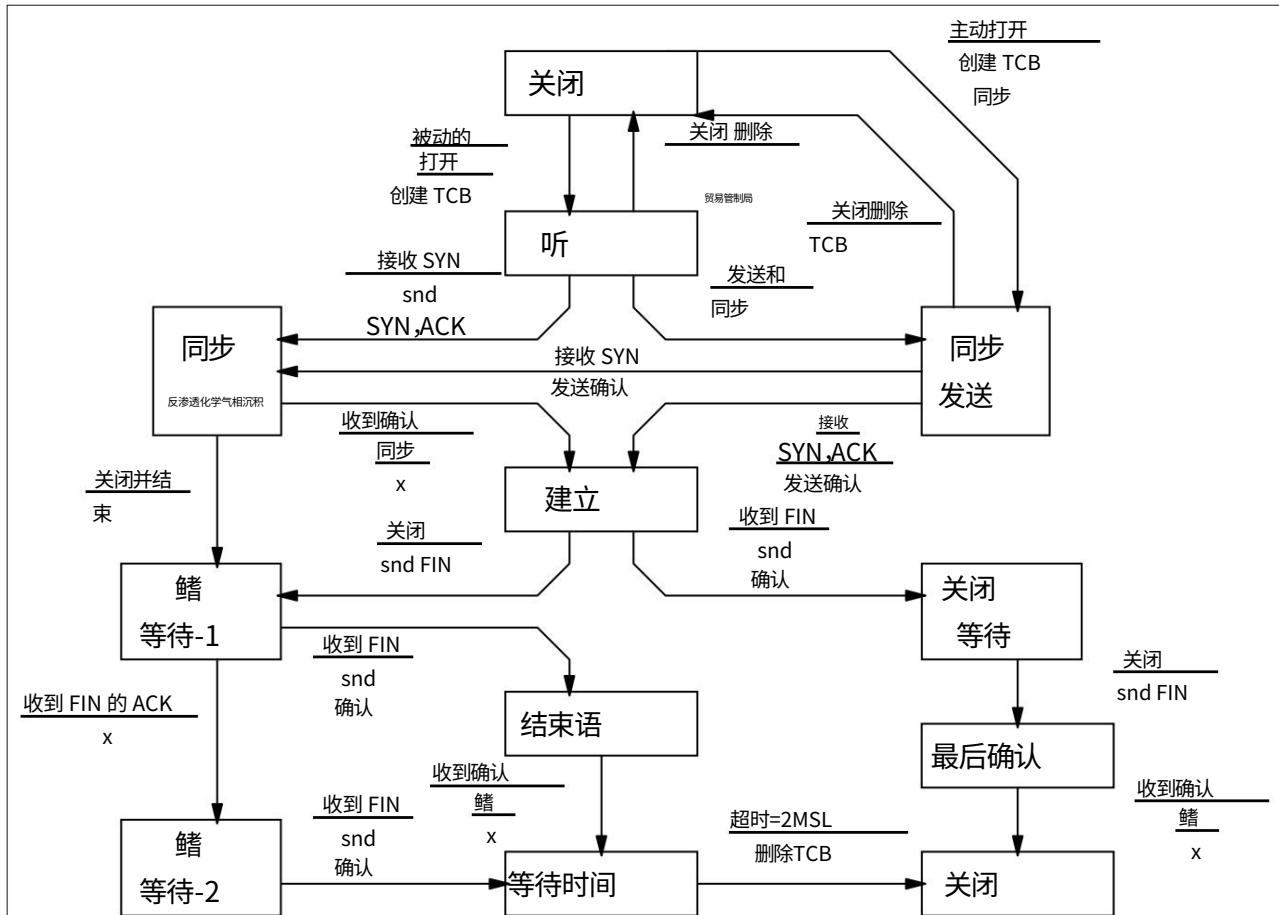


图1-28 TCP连接状态图

您可以使用netstat命令查看每个TCP/IP会话的连接状态。有关更多详细信息,请参阅第53页上的2.3.11“netstat”。

交通管制

TCP/IP实现具有确保高效数据传输的机制,即使在网络传输质量差和拥塞时也能保证数据包的送达。

TCP/IP传输窗口

传输窗口原则是Linux操作系统中TCP/IP实现在性能方面的一个重要方面。简单来说,TCP传输窗口是指在需要连接另一端确认之前,给定主机可以发送或接收的最大数据量。窗口大小由接收主机通过TCP报头中的窗口大小字段提供给发送主机。使用传输窗口,主机可以更有效地发送数据包,因为发送主机不必等待每个发送数据包的确认。这可以提高网络利用率。延迟确认还可以提高效率。TCP窗口最初很小,随着连接另一端每次成功确认而缓慢增大。要优化窗口大小,请参见第126页上的4.7.4“增加网络缓冲区”。

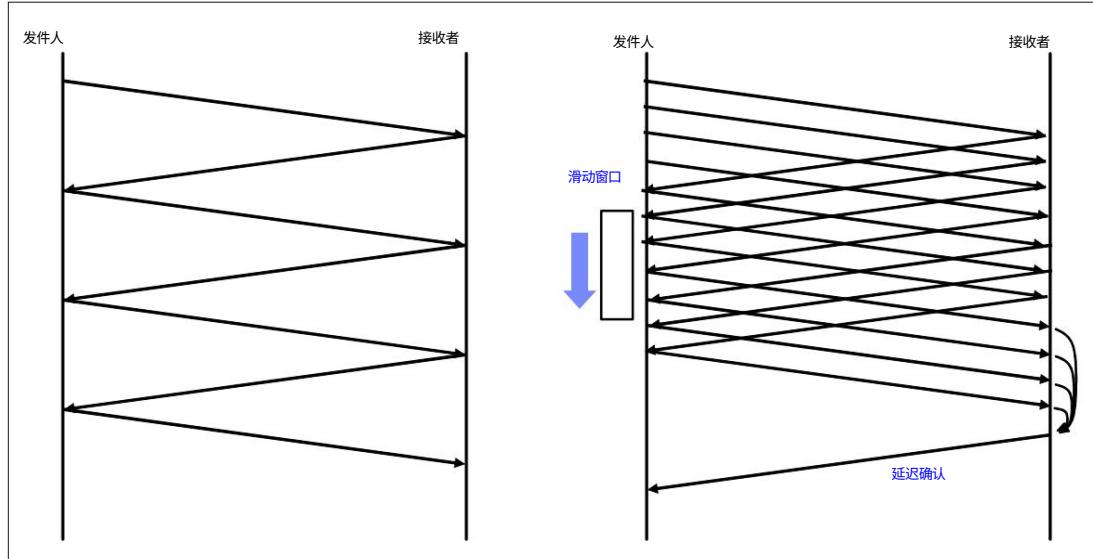


图1-29 滑动窗口和延迟确认

高速网络可以选择使用一种称为窗口缩放的技术来进一步增加最大传输窗口大小。我们将在第 131 页的“调整 TCP 选项”中更详细地分析这些实现的效果。

重传

在连接建立、终止和数据传输过程中,各种原因(例如网络接口故障、路由器速度慢、网络拥塞、网络实现缺陷等等)都可能导致大量超时和数据重传。TCP/IP 通过将数据包排队并多次尝试发送来处理这种情况。

您可以通过配置参数来更改内核的某些行为。例如,在丢包率较高的网络上,您可能需要增加 TCP SYN 连接建立数据包的尝试次数。您还可以通过 /proc/sys/net 下的文件更改某些超时阈值。有关更多信息,请参阅第 130 页上的“调整 TCP 行为”。

1.5.3 卸载

如果系统上的网络适配器支持硬件卸载功能,则内核可以将其部分任务卸载到适配器,从而可以降低 CPU 利用率。

校验和卸载

IP/TCP/UDP 校验是通过比较协议头中校验和字段的值与数据包数据计算出的值来确保数据包正确传输。

TCP分段卸载 (TSO)

当发送到网络适配器的数据大于其支持的最大传输单元 (MTU) 时,数据应该被拆分成 MTU 大小的数据包。适配器会代表内核处理这一操作。

有关更多高级网络功能的信息,请参阅《调整 IBM System x 服务器性能,SG24-5287》中的“10.3. 高级网络功能”一节。

1.5.4 绑定模块

Linux 内核通过使用绑定驱动程序提供网络接口聚合功能。

这是一个独立于设备的 Bonding 驱动程序。(此外,还有一些特定于设备的驱动程序。)Bonding 驱动程序支持 802.3 链路聚合规范以及一些原有的负载均衡和容错实现。它实现了更高级别的可用性和性能改进。请参阅内核文档Documentation/networking/bonding.txt。

1.6 了解 Linux 性能指标

在了解 Linux 操作系统中的各种调优参数和性能测量实用程序之前,有必要先讨论一下各种可用的指标及其与系统性能相关的含义。由于 Linux 是一个开源操作系统,因此有很多性能测量工具可供选择。您最终选择的工具取决于您的个人偏好以及所需的数据量和细节。尽管可用的工具种类繁多,但所有性能测量实用程序测量的指标都相同,因此了解这些指标可以帮助您更好地使用遇到的任何实用程序。因此,我们仅介绍最重要的指标。还有许多更详细的值可能有助于进行更深入的分析,但这超出了本文的讨论范围。

1.6.1 处理器指标

以下是处理器指标:

CPU 利用率

这可能是最直接的指标。它描述的是每个处理器的总体利用率。在 IBM System x 架构上,如果 CPU 利用率持续超过 80%,则很可能存在处理器瓶颈。

用户时间

描述用于用户进程的 CPU 百分比,包括 nice time。用户时间通常值较高,因为在这种情况下,系统会执行实际工作。

系统时间

描述用于内核操作(包括 IRQ 和软中断时间)的 CPU 百分比。

系统时间值过高且持续过高可能表明网络和驱动程序堆栈存在瓶颈。系统通常应该尽可能减少内核时间的占用。

等待

等待 I/O 操作所花费的 CPU 总时间。与阻塞值类似,系统不应花费过多时间等待 I/O 操作;否则,您应该调查相应 I/O 子系统的性能。

空闲时间

描述系统空闲等待任务的 CPU 百分比。

美好时光

描述改变进程执行顺序和优先级的重新调整进程所花费的 CPU 百分比。

平均负载

平均负载不是百分比,而是以下各项总和的滚动平均值:

- 队列中等待处理的进程数
- 等待不可中断任务完成的进程数

即 TASK_RUNNING 和 TASK_UNINTERRUPTIBLE 进程总数的平均值。如果请求 CPU 时间的进程被阻塞（这意味着 CPU 没有时间处理它们）,平均负载就会增加。另一方面,如果每个进程都能立即获得 CPU 时间,并且没有 CPU 周期损失,则负载会降低。

可运行进程

此值表示已准备好执行的进程数。此值在一段持续时间内不应超过物理处理器数量的 10 倍;否则可能会出现处理器瓶颈。

已阻止

等待 I/O 操作完成时无法执行的进程。

阻塞的进程可能会导致 I/O 瓶颈。

上下文切换

系统中线程间切换的次数。大量中断和大量上下文切换可能预示着驱动程序或应用程序存在问题。上下文切换通常不理想,因为每次切换都会刷新 CPU 缓存,但某些上下文切换是必要的。请参阅第 5 页 1.1.5 节“上下文切换”。

中断

中断值包含硬中断和软中断。硬中断对系统性能的影响更大。高中断值表明存在软件瓶颈,无论是内核还是驱动程序。请记住,中断值包含由 CPU 时钟引起的中断。请参阅第 6 页 1.1.6 节“中断处理”。

1.6.2 内存指标

以下是内存指标:

释放内存

与大多数其他操作系统相比,Linux 中的可用内存值无需担心。如第 12 页 1.2.2 节“虚拟内存管理器”中所述,Linux 内核会将大部分未使用的内存分配为文件系统缓存,因此只需从已用内存中减去缓冲区和缓存的总量,即可确定(有效)可用内存。

交换使用情况

此值表示已使用的交换空间量。如第 12 页 1.2.2 节“虚拟内存管理器”中所述,交换空间使用情况只能表明 Linux 的内存管理效率确实很高。交换入/交换出是识别内存瓶颈的可靠方法。如果值持续高于每秒 200 到 300 页,则表示可能存在内存瓶颈。

缓冲区和缓存

缓存分配为文件系统和块设备缓存。

板坯

描述内核的内存使用情况。请注意,内核页面无法被调出到磁盘。

主动记忆与非主动记忆

提供有关系统内存活动使用情况的信息。不活动的内存很可能被 kswapd 守护进程交换到磁盘。请参阅第 14 页的“页框回收”。

1.6.3 网络接口指标

以下是网络接口指标：

接收和发送的数据包

此指标告知您给定网络接口接收和发送的数据包数量。

接收和发送的字节数

该值表示给定网络接口接收和发送的字节数。

每秒碰撞次数

此值指示相应接口所连接的网络上发生的冲突数量。持续的冲突值通常与网络基础设施的瓶颈有关，而非服务器本身。在大多数配置正确的网络中，除非网络基础设施包含集线器，否则冲突很少发生。

数据包丢失

这是由于防火墙配置或由于缺少网络缓冲区而导致内核丢弃的数据包的数量。

超支

溢出表示网络接口耗尽缓冲区空间的次数。

该指标应与丢弃的数据包值结合使用，以识别网络缓冲区或网络队列长度中可能存在的瓶颈。

错误

标记为故障的帧数。这通常是由于网络不匹配或网线部分断裂造成的。对于铜质千兆网络来说，网线部分断裂可能会造成严重的性能问题。

1.6.4 块设备指标

以下是块设备指标：

爱荷华州

CPU 等待 I/O 操作发生的时间。持续较高的值很可能表示存在 I/O 瓶颈。

平均排队长度

未完成的 I/O 请求数量。通常，磁盘队列数量最好为 2 到 3；值过高可能会导致磁盘 I/O 瓶颈。

平均等待时间

衡量处理 I/O 请求所需平均时间（以毫秒为单位）。等待时间包括实际 I/O 操作的时间以及在 I/O 队列中等待的时间。

每秒传输次数

描述每秒执行的 I/O 操作数（读取和写入）。每秒传输量指标与每秒千字节数相结合，可以帮助您

确定系统的平均传输大小。平均传输大小通常应与磁盘子系统使用的条带大小相匹配。

每秒读/写块数

此指标表示自内核 2.6 起，每秒的读取和写入次数（以 1024 字节的块为单位）。早期版本的内核可能会报告不同的块大小，从 512 字节到 4 KB 不等。

每秒读/写千字节数

以千字节为单位的从/向块设备的读取和写入表示传输到/从块设备的实际数据量。



2

第 2 章 监控和基准测试工具

Linux 操作系统的开放性和灵活性催生了大量性能监控工具。其中一些是知名 UNIX 实用程序的 Linux 版本,还有一些是专为 Linux 设计的。大多数 Linux 性能监控工具都依赖于虚拟 proc 文件系统。为了衡量性能,我们还必须使用合适的基准测试工具。

本章将概述一些 Linux 性能监控工具,并讨论一些常用的命令。此外,我们还将介绍一些实用的基准测试工具。

我们讨论的大多数监控工具都随企业 Linux 发行版一起提供。

2.1 简介

企业 Linux 发行版附带许多监控工具。有些工具可以集中处理多个指标，并提供格式良好的输出，方便用户理解系统活动。有些工具则专门针对某些性能指标（例如磁盘 I/O），提供详细信息。

熟悉这些工具有助于增强您对系统状况的理解，并帮助您找到性能问题的可能原因。

2.2 工具功能概述

表2-1列出了本章介绍的监控工具的功能。

表2-1 Linux性能监控工具

工具	最有用的工具功能
顶部	流程活动
虚拟机状态	系统活动、硬件和系统信息
正常运行时间,w	平均系统负载
ps,pstree	显示进程
自由的	内存使用情况
iostat	平均 CPU 负载、磁盘活动
萨里	收集并报告系统活动
mpstat	多处理器使用情况
纽马司他	NUMA 相关统计数据
pmap	进程内存使用情况
网络状态	网络统计
IP流量	实时网络统计
tcpdump,以太	详细的网络流量分析
纳米	收集并报告系统活动
斯特拉斯	系统调用
Proc 文件系统	各种内核统计数据
KDE 系统卫士	实时系统报告和图表
Gnome 系统监视器	实时系统报告和图形

表2-2列出了本章介绍的基准测试工具的功能。

表 2-2 基准测试工具

工具	最有用的工具功能
林茨	操作系统功能的微基准测试
碘酮	文件系统基准测试

工具	最有用的工具功能
netperf	网络性能基准

2.3 监测工具

在本节中,我们将讨论监控工具。大多数工具都包含在企业 Linux 发行版中。您应该熟悉这些工具。

2.3.1 顶部

top命令显示实际的进程活动。默认情况下,它会显示服务器上正在运行的 CPU 占用率最高的任务,并每五秒更新一次列表。
您可以按 PID (按数字排序)、年龄 (最新进程优先)、时间 (累计时间)以及驻留内存使用量和时间 (进程自启动以来占用 CPU 的时间) 对进程进行排序。

示例 2-1 top 命令的输出示例

```
顶部 - 02:06:59 启动 4 天,17:14,2 个用户,平均负载:0.00.0.00.0.00
任务:总计 62 个,其中 1 个正在运行,61 个正在休眠,0 个已停止,0 个僵尸
CPU:0.2% us.0.3% sy.0.0% ni.97.8% id.1.7% wa.0.0% hi.0.0% si
内存:总计 515144k,已用 317624k,可用 197520k,交换:总计 1048120k,               66068k 缓冲区
                  已使用 12k,可用 1048108k,已缓存 179632k
```

PID 用户	PR NI VIRT RES SHR S %CPU %MEM TIME+ 命令
13737 根 238	17 0 1760 896 1540 R 0.7 0.2 0:00.05 顶部
根 1 根 2 根	5 -10 0 0 S 0.3 0.0 0:01.56 reiserfs/0
3 根 4 根	0 16 0 588 240 444 S 0.0 0.0 0:05.70 初始化
5 根 6 根	RT 0 0 34 19 0 0 0 S 0.0 0.0 0:00.00 迁移/0
	RT 0 0 34 19 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
	5 -10 0 5 -10 0 0 0 S 0.0 0.0 0:00.00 迁移/1
	5 -10 0 5 -10 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/1
	15 0 0 5 -10 0 0 S 0.0 0.0 0:00.02 事件/0
7 根	0 16 0 0 15 0 0 0 S 0.0 0.0 0:00.00 事件/1
8 根 9 根	0 13 -10 0 13 0 0 S 0.0 0.0 0:00.09 kblockd/0
10 根 13	-10 0 0 0 S 0.0 0.0 0:00.01 kblockd/1
根 14 根 16	0 0 0 秒 0.0 0.0 0:00.00 kirqd
根 17 根 18	0 0 0 S 0.0 0.0 0:00.02 khelperd/0
根	0 0 0 秒 0.0 0.0 0:00.45 pdflush
	0 0 0 秒 0.0 0.0 0:00.61 kswapd0
	0 0 0 S 0.0 0.0 0:00.00 aio/0
	0 0 0 秒 0.0 0.0 0:00.00 aio/1

您可以使用renice进一步修改进程,为每个进程赋予新的优先级。如果某个进程挂起或占用过多 CPU,您可以终止该进程 (kill命令)。

输出中的列是:

PID	过程识别。
用户	拥有 (或许是启动)该进程的用户的名称。
优先等级	进程的优先级。(有关详细信息,请参阅第 5 页上的 1.1.4 “进程优先级和 nice 级别”)

友善性

友好度（进程是否尝试通过给定的数字调整优先级来表现友好。详情见下文。）

尺寸	进程使用的内存量（代码+数据+堆栈），以千字节为单位。
RSS	已使用的物理 RAM 数量（以千字节为单位）。
分享	与其他进程共享的内存量（以千字节为单位）。
统计	进程状态:S=休眠,R=运行,T=停止或跟踪,D=可中断休眠,Z=僵尸。进程状态将在第 6 页 1.1.7 节“进程状态”中讨论。
%中央处理器	CPU 使用率份额（自上次屏幕更新以来）。
%内存	物理内存的份额。
时间	该进程使用的总 CPU 时间（自启动以来）。
COMMAND	用于启动任务的命令行（包括参数）。
顶级实用程序支持几个有用的热键，包括：	
吨	关闭或打开显示摘要信息。
米	关闭或打开显示内存信息。
-+	按各种系统资源的最大消耗者排序显示。有助于快速识别系统中性能要求高的任务。
f	进入 top 的交互式配置屏幕。有助于设置 top 用于特定任务。
哦	使您能够以交互方式选择顶部内的排序。
r	发出 renice 命令。
千	发出终止命令。

2.3.2 vmstat

vmstat 提供有关进程、内存、分页、块 I/O、陷阱和 CPU 活动的信息。vmstat 命令可以显示平均数据或实际样本。通过为 vmstat 提供采样频率和采样持续时间来启用采样模式。

注意：在采样模式下，请考虑实际数据采集过程中可能出现的峰值。将采样频率降低可以避免此类隐藏的峰值。

示例 2-2 vmstat 的输出示例

```
[root@lnxsu4 ~]# vmstat 2
进程 ----- 内存 ----- 交换 ----- io ----- 系统 ----- cpu -----
  rb swpd 免费 buff 缓存 si so bo in cs us sy id wa      双
  0 1 0 1742264 112116 1999864 1 3 0 0 99 0          0 0        4 3
  0 1 0 1742072 112208 1999772 0 2536 1258 1146 0 1 75 24 0
  0 1 0 1741880 112260 1999720 0 2668 1235 1002 0 1 75 24 0
  0 1 0 1741560 112308 1999932 0 2930 1240 1015 0 1 75 24 0
  1 1 0 1741304 112344 2000416 0 2980 1238 925 0 1 75 24 0
  0 1 0 1741176 112384 2000636 0 2968 1233 929 0 1 75 24 0
  0 1 0 1741304 112420 2000600 0 3024 1247 925 0 1 75 24 0
```

注意：vmstat 报告的第一行数据显示自上次重启以来的平均值，因此应该将其消除。

输出中的列如下：

进程 (procs)	r:等待运行时的进程数 b:处于不可中断睡眠状态的进程数
记忆	swpd:已使用的虚拟内存量 (KB) free:空闲内存量 (KB) buff:用作缓冲区的内存量 (KB) cache:用作缓存的内存量 (KB)
交换	si:从磁盘交换的内存量 (KBps) so:交换到磁盘的内存量 (KBps)
输入输出	bi:发送到块设备的块数 (块/秒) bo:从块设备接收的块数 (块/秒)
系统	in:每秒中断次数,包括时钟 cs:每秒上下文切换次数
CPU (占总 CPU 时间的百分比)	us:运行非内核代码所花费的时间 (用户时间,包括 nice 时间)。 sy:运行内核代码所花费的时间 (系统时间)。 id:空闲时间。在 Linux 2.5.41 之前,此时间包含 I/O 等待时间。 wa:等待 IO 的时间。在 Linux 2.5.41 之前的版本中,它显示为零。

vmstat命令支持大量命令行参数,这些参数在vmstat 的手册页中有完整说明。一些比较有用的标志包括：

-m	显示内核的内存利用率 (slabs)
-a	提供有关活动和非活动内存页面的信息
-n	仅显示一个标题行,这在以采样模式运行vmstat并将输出通过管道传输到文件时非常有用。(例如,root#vmstat -n 2 10 会以两秒的采样率生成 10 次 vmstat。)

当与-p {partition} 标志一起使用时, vmstat还提供 I/O 统计数据。

2.3.3 正常运行时间

uptime命令可用于查看服务器运行了多长时间、有多少用户登录,以及快速概览服务器的平均负载。

(请参阅第 34 页上的 1.6.1 “处理器指标”)。显示过去 1 分钟、5 分钟和 15 分钟间隔的系统负载平均值。

负载的最佳值为 1,这意味着每个进程都可以立即访问 CPU,并且不会浪费任何 CPU 周期。典型负载因系统而异。

对于单处理器工作站,1 或 2 可能是可以接受的,而在多处理器服务器上您可能会看到 8 到 10 的值。

您可以使用uptime来查明服务器或网络的问题。例如,如果某个网络应用程序运行不佳,请运行uptime来查看系统负载是否过高。如果负载不高,则问题更有可能出在网络上,而不是服务器上。

提示:您可以使用w代替uptime。w还提供有关当前谁登录到机器以及用户正在做什么的信息。

例 2-3 正常运行时间的示例输出

凌晨 1:57 启动 4 天 17:05,2 个用户,平均负载:0.00,0.00,0.00

2.3.4 ps 和 pstree

ps和pstree命令是系统分析中最基本的命令之一。ps有三种不同类型的命令选项:UNIX风格、BSD风格和GNU风格。这里我们来看一下UNIX风格的选项。

ps命令提供现有进程的列表。top命令显示进程信息,但ps命令会提供更详细的信息。列出的进程数量取决于所使用的选项。一个简单的ps -A命令可以列出所有进程及其各自的进程ID(PID),这对于进一步调查至关重要。使用pmap或renice等工具时需要PID号。

在运行Java™应用程序的系统上,ps -A命令的输出可能会很容易地填满显示屏,以至于很难获得所有正在运行的进程的完整图像。

在这种情况下,pstree命令可能会派上用场,因为它可以以树形结构显示正在运行的进程,并合并生成的子进程(例如Java线程)。pstree命令可以帮助识别原始进程。此外,ps还有另一个变体,pgrep。

它可能也很有用。

例 2-4 ps 输出示例

```
[root@bc1srv7 ~]# ps -A
 PID TTY      时间命令
 1 ? 00:00:00 初始化
 2 ?      00:00:00 迁移/0
 3?第    00:00:00 ksoftirqd/0
 2347章第 00:00:00 sshd
 2435章    00:00:00 发送邮件
 27397 ?
 27402 分/0 27434 00:00:00 猛击
 分/0      下午 00:00:00
```

我们将查看一些有用的选项以获取详细信息。

-e	所有进程。与-A相同
-l	显示长格式
-F	额外完整模式
-H	森林
-L	显示线程,可能包含LWP和NLWP列
-m	显示进程后的线程

以下是使用以下命令的进程详细输出的示例:

ps-eFL

例 2-5 详细输出示例

```
[root@lnxsu3 ~]# ps -eFL
文件系统用户ID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN RSS PSR STIME TTY 1 0 552 0 Mar08 ? 2 0 0          时间命令
4 S根           0      Mar08 ? 3 0 1 16a008457 -1 -40 -- 1 94          00:00:01 初始化 [3]
1 S根           1      1      19 -          0 迁移 0      0          00:00:36 [迁移/0]
1 S根           2 3      1      ksofti          0          0          00:00:00 [ksoftirqd/0]
```

1 S根	4	1	405	1 -40 -- 1 94 19	0 迁移 0	0	1 Mar08 ?	00:00:27 [迁移/1]
1 S根	5	1	060	-1 -40 -- 1 94	ksofti 0 迁	0	1 Mar08 ?	00:00:00 [ksoftirqd/1]
1 S根	6	1	708	19 -1 -40 -- 1	移 0 ksofti	0	2 Mar08 ?	00:00:00 [迁移/2]
1 S根		1	090	94 19 -1 65 -10	0 迁移 0	0	2 Mar08 ?	00:00:00 [ksoftirqd/2]
1 S根	7	1	100	-1 65 -10 -1 65	ksofti 0 工	0	3 Mar08 ?	00:00:00 [迁移/3]
1 S根	8	1	110	-10 -1 65 -10 -	人 0 工人 0	0	3 Mar08 ?	00:00:00 [ksoftirqd/3]
1 S根	9	1	120		工人 0 工人	0	0 Mar08 ?	00:00:00 [事件/0]
1 S根	10	1	130			0	1 Mar08 ?	00:00:00 [事件/1]
1 S根	11	1				0	2 Mar08 ?	00:00:00 [事件/2]
1 S根	12 13	1				3 Mar08 ?		00:00:00 [事件/3]
5 S 根 3493 1 3493 0				1 76 0 -1889 -1 78 0 -374 -	4504 1 Mar08?		00:07:40 暂停	
4 S 根 3502 1 3502 0				1 78 0 -445 -1 78 0 -815 -	408 1 Mar08 tty1		00:00:00 /sbin/mingetty tty1	
4 S 根 3503 1 3503 0				1 78 0 -373 -1 78 0 -569 -	412 1 Mar08 tty2		00:00:00 /sbin/mingetty tty2	
4 S 根 3504 1 3504 0				1 78 0 -585 -1 76 0 -718 -	412 2 Mar08 tty3		00:00:00 /sbin/mingetty tty3	
4 S 根 3505 1 3505 0				1 75 0 -1443 -1 75 0 -5843	412 1 Mar08 tty4		00:00:00 /sbin/mingetty tty4	
4 S 根 3506 1 3506 0				-	412 3 Mar08 tty5		00:00:00 /sbin/mingetty tty5	
4 S 根 3507 1 3507 0					412 0 Mar08 tty6		00:00:00 /sbin/mingetty tty6	
0 S takech 3509 1 3509 0					1080 0 Mar08?		00:00:00 /usr/libexec/gam_server	
0 S takech 4057 1 4057 0					1860 0 Mar08?		00:00:01 xscreensaver-nosplash	
4 S 根 4239 1 4239 0 --sm-client-id=default1					9180 1 Mar08?		00:00:01 /usr/bin/metacity	
0 S takech 4238 1 4238 0 --sm-client-id=default1				1 76 0 -3414 -	5212 2 Mar08?		00:00:00 /usr/bin/metacity	
4 S 根 4246 1 4246 0 --sm-client-id default2				1 76 0 -5967 -	12112 2 Mar08?		00:00:00 gnome 面板	
0 S takech 4247 1 4247 0 --sm-client-id default2				1 77 0 -5515 -	11068 0 Mar08?		00:00:00 gnome 面板	
0 S takech 4249 1 4249 0 --无默认窗口 --sm-client-id default3				9 76 0 -10598 -	17520 1 Mar08?		00:00:01 鹦鹉螺	
1 S takech -- 4249 1 4282 0 no-default-window --sm-client-id default3				9 75 0 -10598 -	17520 0 Mar08?		00:00:00 鹦鹉螺	
1 S takech -- 4249 1 4311 0 no-default-window --sm-client-id default3				9 75 0 -10598 322565 17520 0 Mar08?			00:00:00 鹦鹉螺	
1 S takech -- 4249 1 4312 0 no-default-window --sm-client-id default3				9 75 0 -10598 322565 17520 0 Mar08?			00:00:00 鹦鹉螺	

输出中的列是：

F 进程标志
 秒 进程状态 :S=休眠,R=运行,T=停止或跟踪,D=可中断休眠,Z=僵尸。进程状态将在第 6 页
 1.1.7 节 “进程状态”中进一步讨论。

唯一标识符 拥有 (或许是启动)该进程的用户的名称。
 PID 进程 ID 号
 进程标识符 父进程ID号
 轻量级 所报告的轻量级进程或线程 (LWP) 的 LWP ID。
 碳 处理器利用率百分比的整数值。(CPU 使用率)
 NLWP进程中的 lwp (线程) 数量。 (别名 thcount)。
 优先等级 进程的优先级。(有关详细信息,请参阅第 5 页上的 1.1.4 “进程优先级和 nice 级别” 。)

友好度 (进程是否通过给定的数字调整优先级来尝试变得友好;详情见下文)。

ADDR进程地址空间 (不显示)

深圳 进程使用的内存量 (代码+数据+堆栈) ,以千字节为单位。

WCHAN进程在其中休眠的内核函数的名称,如果进程正在运行,则为“-”,如果进程是多线程的并且ps未显示线程,则为“*”。

RSS 驻留集大小,任务已使用的非交换物理内存(以千字节为单位)。

脉冲重复序列 该进程当前被分配到的处理器。

STIME命令开始的时间。

终端电话 终端

时间 该进程使用的总CPU时间(自启动以来)。

命令 用于启动任务的命令行(包括参数)。

线程信息

您可以使用ps -L选项查看线程信息。

示例 2-6 使用 ps -L 查看线程信息

[root@edam ~]# ps -eLF grep -E LWP /usr/sbin/httpd					
唯一识别符	PID	PPID	LWP	C NLWP	时间命令
根	4504	0	4504	4507 0	1 4313 8600 2 08:33 ? 1 4313 4236 00:00:00 /usr/sbin/httpd
apache	4508	4504	4508 0	4509 4504	1 08:33 ? 1 4313 4228 1 08:33 ? 1 00:00:00 /usr/sbin/httpd
apache	4509	0	4510	4504 4510 0	4313 4228 0 08:33 ? 1 4313 4228 3 00:00:00 /usr/sbin/httpd
apache					08:33 ? 00:00:00 /usr/sbin/httpd
apache					00:00:00 /usr/sbin/httpd

[root@edam ~]# ps -eLF grep -E LWP /usr/sbin/httpd					
唯一识别符	PID	PPID	LWP	C NLWP	时间命令
根	4632	0	1 3640	7772 2 08:44?	4635 4632 4635 0 27 72795 5352 3 00:00:00 /usr/sbin/httpd.worker
apache	08:44?	4635	4632 4638 0 27	72795 5352 1 08:44?	4635 4632 4639 00:00:00 /usr/sbin/httpd.worker
apache	0 27	72795	5352 3 08:44?	4635 4632 4640 0 27	72795 5352 3 08:44? 00:00:00 /usr/sbin/httpd.worker
apache					00:00:00 /usr/sbin/httpd.worker
apache					00:00:00 /usr/sbin/httpd.worker

2.3.5 免费

/bin/free命令显示系统中可用和已用内存(包括交换空间)的总量信息。它还包含内核使用的缓冲区和缓存的信息。

示例 2-7 free 命令的输出示例

	总计	二手	免费	共享	缓冲区	缓存
记忆:	1291980 -/	998940	293040	0	89356	772016
+ 缓冲区/缓存:		137568	1154412			
交换:2040244		0	2040244			

使用free时,请记住Linux内存架构和虚拟内存管理器的工作方式。空闲内存的用处是有限的,单纯的交换空间利用率统计并不能反映内存瓶颈。

第 47 页的图 2-1 描述了free命令输出所显示的基本概念。

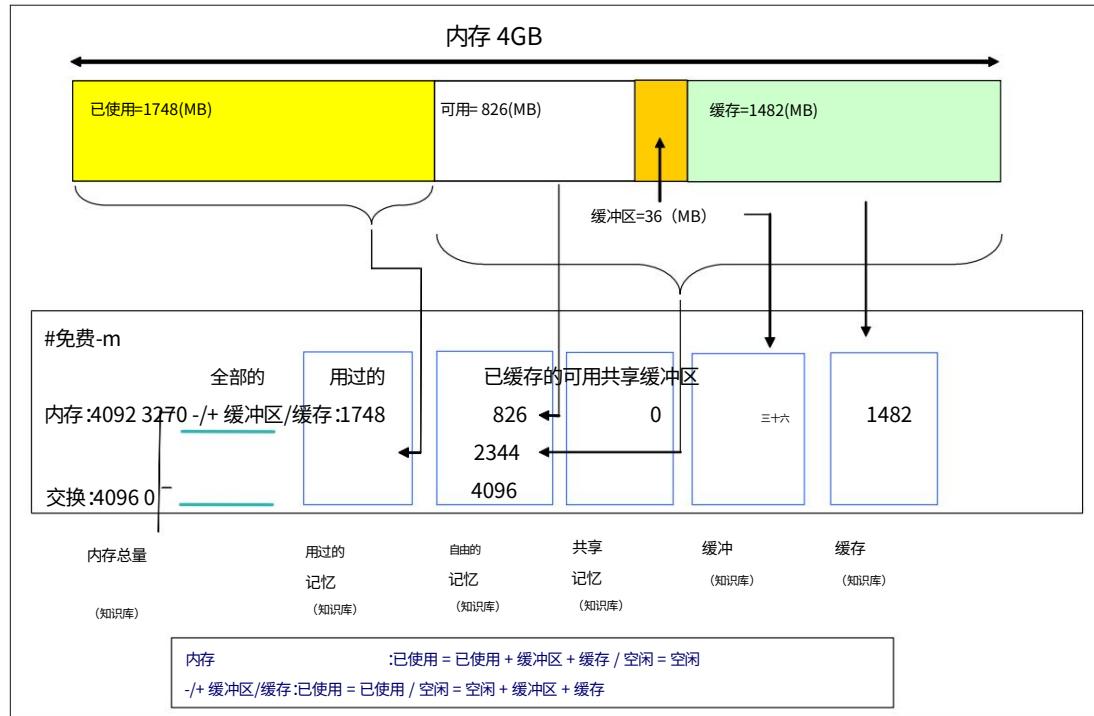


图2-1 free命令输出

free命令的有用参数包括：

- b、-k、-m、-g以字节、千字节、兆字节和千兆字节为单位显示值
- l 区分低端内存和高端内存（请参阅第 10 页上的 1.2 “Linux 内存架构”。）
- c <计数> 显示空闲输出<count>次

区域中使用的内存

使用-l选项，您可以查看每个内存区域使用了多少内存。

示例 2-8 和示例 2-9 分别展示了 32 位和 64 位系统的free -l输出示例。请注意，64 位系统不再使用高端内存。

示例 2-8 32 位版本内核上的 free 命令的示例输出

```
[root@edam ~]# 免费 -l
          全部的    二手    免费    共享 0    缓冲区    缓存
内存:4154484        2381500  1772984          0      108256  1974344
低:877828           199436   678392          0      10300   42060
最高:3276656 -/+ 缓
存:                           2182064  1094592          0      10300   42060
交换:4194296          298900  3855584          0      10300   42060
          0          4194296          0      10300   42060
```

示例 2-9 64 位版本内核上的 free 命令的示例输出

```
[root@lnxsu4 ~]# 免费 -l
          总计    使用    免费    共享 0    缓冲区    缓存
记忆:      4037420      138508  3898912          0      10300   42060
低的:      4037420 0
高: -/+
+ 缓冲区/缓存:          138508 0 8614851272
```

交换:	2031608	332	2031276
-----	---------	-----	---------

您还可以使用 /proc/buddyinfo 文件确定每个区域中可用的内存块数量。每列数字表示可用页面的数量。在示例 2-10 中,ZONE_DMA 中有 5 个大小为 2^{2*PAGE_SIZE} 的块可用,ZONE_DMA32 中有 16 个大小为 2^{3*PAGE_SIZE} 的块可用。记住伙伴系统是如何分配页面的（请参阅第 13 页的“伙伴系统”）。这些信息可以显示内存碎片化的程度,并让您了解可以安全分配的页面数量。

示例 2-10 64 位系统的伙伴系统信息

```
[root@lnxsu5 ~]# cat /proc/buddyinfo
节点 0,区域      直接市场 1
节点 0,区域          DMA32      5       4       6       1       1       0       2       0       2
节点 0,区域正常        0       56     3 14     2      16      7      3      1      7     41     42    670
节点 0,区域正常        0       6       3       2       1       0       1       0       0       1       0
```

2.3.6 iostat

iostat命令显示自系统启动以来的平均 CPU 时间（类似于uptime）。它还会创建一份包含服务器磁盘子系统活动的报告,该报告分为两部分:CPU 利用率和设备（磁盘）利用率。要使用iostat执行详细的 I/O 瓶颈分析和性能调优,请参阅第 84 页上的 3.4.1 “查找磁盘瓶颈”。iostat 实用程序是 sysstat 软件包的一部分。

示例 2-11 iostat 的示例输出

Linux 2.4.21-9.0.3.EL (x232) 2004年5月11日

平均 CPU:%用户%nice%sys%空闲
0.03 0.00 0.02 99.95

设备:	每秒块读取/秒块写入/秒块读取块写入
dev2-0	0.00 0.00 0.04 203 2880
dev8-0	0.45 2.18 2.21 166464 168268
dev8-1	0.00 0.00 0.00 16 0
dev8-2	0.00 0.00 0.00 8 0
dev8-3	0.00 0.00 0.00 344 0

CPU 利用率报告包含四个部分：

%用户 显示在用户级别（应用程序）执行时占用的 CPU 利用率百分比。

%好的 显示在用户级执行 nice 优先级的程序时占用的 CPU 利用率百分比。（优先级和 nice 级别的说明请参见第 67 页 2.3.7 节“nice,renice”）。

%系统 显示在系统级（内核）执行时占用的 CPU 利用率百分比。

%闲置的 显示 CPU 空闲时间的百分比。

设备利用率报告包含以下部分：

设备 块设备的名称。

每秒交易量

每秒向设备传输的次数（每秒的 I/O 请求数）。

多个单个 I/O 请求可以组合在一个传输请求中，因为传输请求可以有不同的大小。

Blk_read/s, Blk_wrtn/s

每秒读取和写入的块数表示以秒为单位从设备读取或写入的数据量。块的大小也可能不同。典型的大小为 1024、2048 和 4096 字节，具体取决于分区大小。例如，可以使用以下命令查找 /dev/sda1 的块大小：

```
dumpe2fs -h /dev/sda1 | grep -F "块大小"
```

这将产生类似以下内容的输出：

```
dumpe2fs 1.34 (2003年7月25日)
区块大小:1024
```

Blk_read、Blk_wrtn

指示自启动以来读取和写入的块总数。

iostat 可以使用许多选项。从性能角度来看，最有用的选项是 -x 选项。它显示扩展统计信息。以下是示例输出。

示例 2-12 iostat -x 扩展统计信息显示

```
[root@lnxsu4 ~]# iostat -d -x sdb 1
Linux 2.6.9-42.ELsmp (lnxsu4.itso.ral.ibm.com) 2007 年 3 月 18 日
```

设备:sdb	rrqm/s wrqm/sr/sw/s rsec/s wsec/s 0.46 0.00 0.15 0.00 0.02 0.00	rkB/s 0.23	wkB/s 0.00	avgrq-sz 29.02	avgqu-sz 0.00	等待 svctm 2.60	%util 1.05 0.00
--------	--	---------------	---------------	-------------------	------------------	------------------	--------------------

rrqm/s, wrqm/s

每秒向设备发出的合并读/写请求数。多个单个 I/O 请求可以合并到一个传输请求中，因为传输请求的大小可能不同。

速度/秒, 宽度/秒

每秒向设备发出的读/写请求数。

rsec/s, wsec/s 每秒从设备读取/写入的扇区数。

rkB/s, wkB/s 每秒从设备读取/写入的千字节数。

平均数

向设备发出的请求的平均大小。此值以扇区为单位显示。

avgqu-sz

向设备发出的请求的平均队列长度。

等待

显示在系统级（内核）执行时使用的 CPU 利用率百分比。

svctm

向设备发出的 I/O 请求的平均服务时间（以毫秒为单位）。

%实用程序

向设备发出 I/O 请求的 CPU 时间百分比（设备的带宽利用率）。当此值接近 100% 时，表示设备已饱和。

计算平均 I/O 大小可能有助于根据访问模式定制磁盘子系统。以下示例是使用带有 -d 和 -x 标志的 iostat 的输出，以便仅显示有关感兴趣的磁盘子系统的信息：

示例 2-13 使用 iostat -x -d 分析平均 I/O 大小

设备:	rrqm/s wrqm/sr/sw/s rsec/s wsec/s 0.00 0.00 0.00 2502.97 0.00 达斯直流 24601.98	rkB/秒 0.00 12300.99 wkB/s avgrrq-sz 9.83 142.93 57.08 0.40 100.00	avgqu-sz 等待 svctm %util
-----	--	--	-------------------------

示例 2-13 中的 iostat 输出显示,设备 dasdc 每秒必须写入 12300.99 KB 的数据,如 kB_wrtn/s 标题下所示。该数据量以 2502.97 次 I/O 发送到磁盘子系统,如上例中 w/s 下所示。

平均 I/O 大小或平均请求大小显示在 avgrrq-sz 下,在我们的示例中为 9.83 个 512 字节块。对于异步写入,平均 I/O 大小通常为某个奇数。然而,大多数应用程序执行的读写 I/O 大小都是 4 KB 的倍数(例如 4 KB、8 KB、16 KB、32 KB 等等)。在上面的示例中,应用程序只发出 4 KB 的随机写入请求,但 iostat 显示的平均请求大小为 4.915 KB。这种差异是由 Linux 文件系统造成的,即使我们执行的是随机写入,Linux 文件系统也发现一些可以合并的 I/O,以便更高效地刷新到磁盘子系统。

注意:当使用文件系统的默认异步模式时,iostat 中显示的平均请求大小才是正确的。即使应用程序执行的写入请求大小不同,Linux 的 I/O 层也很可能进行合并,从而改变平均 I/O 大小。

2.3.7 沙特里亚尔

sar 命令用于收集、报告和保存系统活动信息。sar

该命令由三个应用程序组成: sar(用于显示数据)以及 sa1 和 sa2(用于收集和存储数据)。sar 工具提供丰富的选项,因此请务必查看其手册页。sar 实用程序是 sysstat 软件包的一部分。

使用 sa1 和 sa2,可以配置系统以获取信息并记录下来以供日后分析。

提示:我们建议您在大多数(如果不是全部)系统上运行 sar。这样,当出现性能问题时,您将能够以极低的开销获得非常详细的信息,并且无需额外付费。

为此,请在 /etc/crontab 文件中添加以下行(示例 2-14)。请记住,在系统上安装 sar 后,会自动设置一个默认的 cron 任务,每天运行 sar。

示例 2-14 使用 cron 启动自动日志报告的示例

```
# 工作日上午 8 点至下午 7 点每 10 分钟报告一次活动。
*/10 8-18 * * 1-5 /usr/lib/sa/sa1 600 &
# 工作日晚上 7 点至早上 8 点每小时报告一次活动。
0 19-7 * * 1-5 /usr/lib/sa/sa1 &
# 周六和周日每小时报告一次活动情况。
0 * * * 0,6 /usr/lib/sa/sa1 &
# 每日摘要于 19:05 准备
5 19 * * * /usr/lib/sa/sa2 -A &
```

sar 工具的原始数据存储在 /var/log/sa/ 下,其中的各个文件代表相应月份的日期。要检查结果,请选择该月份的星期几和所需的性能数据。例如,要显示从 21 号开始的网络计数器,请使用命令 sar -n DEV -f sa21 并将其通过管道传输到 less,如第 51 页的示例 2-15 所示。

示例 2-15 使用 sar 显示系统统计信息

```
[root@linux sa]# sar -n DEV -f sa21 |较少的
Linux 2.6.9-5.ELsmp (linux.itso.ral.ibm.com)          2005年4月21日

上午 12:00:01      IFACE rxpck/s txpck/s rxbyt/s txbyt/s rxcmp/s txcmp/s rxmcst/s
上午 12:10:01      低       0.00     0.00     0.00     0.00     0.00     0.00     0.00
上午 12:10:01      eth0     1.80     0.00   247.89     0.00     0.00     0.00     0.00
上午 12:10:01      eth1     0.00     0.00     0.00     0.00     0.00     0.00     0.00
```

您还可以使用sar从命令行运行近乎实时的报告（示例 2-16）。

示例 2-16 临时 CPU 监控

```
[root@x232 root]# sar -u 3 10
Linux 2.4.21-9.0.3.EL (x232)          2004年5月22日

下午 2:10:40      CPU    用户百 %nice %系统 0.00 %闲置的
下午 2:10:43      全     分比   0.00 0.00 0.00 0.00 100.00
下午 2:10:46      部     0.00   0.00 0.00 18.57 0.00 99.67
下午 2:10:49      全     0.33   28.57 0.00 100.00 100.00
下午 2:10:52      部     0.00   0.00 0.00 0.00 100.00 74.29
下午 2:10:55      全     7.14   0.00 50.00 0.00 0.00
下午 2:10:58      部     71.43  100.00 0.00 3.33 0.00
下午 2:11:01      全     0.00   0.00
下午 02:11:04      部     0.00   0.00
下午 2:11:07      全     0.00   0.00
下午 2:11:10      部     50.00   0.00
平均的:          全部 全部 全部 全部 1.62 95.06
```

从收集的数据中,您可以看到 CPU 利用率 (%user,%nice,%system,%idle)、内存分页、网络 I/O 和传输统计、进程创建活动、块设备活动以及每秒中断次数的详细概述。

2.3.8 mpstat

mpstat命令用于报告多处理器服务器上每个可用 CPU 的活动情况。此外,还会报告所有 CPU 的全局平均活动情况。mpstat 实用程序是 sysstat 软件包的一部分。

mpstat实用程序允许您显示每个系统或每个处理器的整体 CPU 统计信息。mpstat 还可以在采样模式下创建统计信息,类似于 vmstat命令,但需要指定采样频率和采样次数。示例 2-17 显示了使用mpstat -P ALL 命令创建的示例输出,用于显示每个处理器的平均 CPU 利用率。

示例 2-17 多处理器系统上的 mpstat 命令的输出

```
[root@linux ~]# mpstat -P ALL
Linux 2.6.9-5.ELsmp (linux.itso.ral.ibm.com)          2005年4月22日

下午 03:19:21 CPU %user %nice %system %iowait %irq %soft %idle 下午 03:19:21 全部 0.34 0.02 下午 03:19:21 0.33 0.04 下午      中断/秒
03:19:21 1 0.36 0.01      0.03     0.00     0.06     0.08 99.47 1124.22
                           0.03     0.00     0.03     0.15 99.43 612.12
                           0.03     0.00     0.10     0.01 99.51 512.09
```

要以一秒的间隔显示多处理器服务器的所有处理器的三个统计信息条目,请使用以下命令:

mpstat -P 全部 1 2

示例 2-18 双向机器上的 mpstat 命令输出

下午 03:31:51 CPU %user %nice %system %iowait %irq %soft %idle 下午 03:31:52 全部 0.00 0.00 下午 03:31:52 0 0.00 0.00 下午 中断/秒					
03:31:52 1 0.00 0.00	0.00	0.00	0.00	0.00	100.00 1018.81
	0.00	0.00	0.00	0.00	100.00 991.09
	0.00	0.00	0.00	0.00	99.01 27.72
<hr/>					
平均的:	CPU %用户%nice%系统%iowait%irq%soft%idle全部0.00 0 0.00 1 0.00				中断/秒
平均的:	0.00	0.00	0.00	0.00	100.00 1031.89
平均的:	0.00	0.00	0.00	0.00	100.00 795.68
平均的:	0.00	0.00	0.00	0.00	99.67 236.54

要获取mpstat命令的完整语法,请发出:

mpstat -?

2.3.9 numastat

随着 IBM System x 3950 等非统一内存架构 (NUMA) 系统的出现,NUMA 架构已成为企业数据中心的主流。然而,NUMA 系统也给性能调优带来了新的挑战。在 NUMA 系统出现之前,诸如内存局部性之类的问题并不引人关注。幸运的是,企业 Linux 发行版提供了一个用于监控 NUMA 架构行为的工具.numastat

命令提供有关本地内存与远程内存使用率以及所有节点整体内存配置的信息。应调查本地内存分配失败 (如 numa_miss 列所示) 和远程内存 (较慢的内存) 分配 (如 numa_foreign 列所示)。过多的远程内存分配会增加系统延迟,并可能降低整体性能。将进程绑定到具有本地 RAM 中内存映射的节点很可能会提高性能。

示例 2-19 numastat 命令的示例输出

[root@linux ~]# numastat

	节点1	节点0
numa_hit	76557759	92126519
numa_miss	30772308	30827638
numa_foreign	30827638	30772308
interleave_hit	106507	103832
local_node	76502227	92086995
other_node	30827840	30867162

2.3.10 pmap

pmap命令报告一个或多个进程正在使用的内存量。您可以使用此工具确定服务器上哪些进程正在分配内存,以及

此内存量是否是造成内存瓶颈的原因。有关详细信息,请使用pmap -d选项。

pmap -d <进程号>

示例 2-20 init 进程正在使用的进程内存信息

地址	千字节模式偏移映射	设备
0000000000400000	36 rx-- 00000000000000000000 0fd:00000 初始化	
0000000000508000	8 读写--- 00000000000000000000 0fd:00000 初始化	
0000000000050a000	132 rwx-- 00000000000000000000 000:00000 [匿名]	
00000002a95556000	4 rw--- 00000002a95556000 000:00000 [匿名]	
00000002a95574000	8 rw--- 00000002a95574000 000:00000 [匿名]	
00000030c3000000	84 rx-- 00000000000000000000 0fd:00000 ld-2.3.4.so	
00000030c3114000	8 读写--- 00000000000014000 0fd:00000 ld-2.3.4.so	
00000030c3200000	1196 rx-- 00000000000000000000 0fd:00000 libc-2.3.4.so	
00000030c332b000	1024 ----- 000000000012b000 0fd:00000 libc-2.3.4.so	
00000030c342b000	8 r--- 000000000012b000 0fd:00000 libc-2.3.4.so	
00000030c342d000	12 rw--- 000000000012d000 0fd:00000 libc-2.3.4.so	
00000030c3430000	16 rw--- 00000030c3430000 000:00000 [匿名]	
00000030c3700000	56 rx-- 00000000000000000000 0fd:00000 libsepolicy.so.1	
00000030c370e000	1020 ----- 0000000000000000e000 0fd:00000 libsepolicy.so.1	
00000030c380d000	4 读写--- 000000000000d000 0fd:00000 libsepolicy.so.1	
00000030c380e000	32 rw--- 00000030c380e000 000:00000 [匿名]	
00000030c4500000	56 rx-- 00000000000000000000 0fd:00000 libselinux.so.1	
00000030c450e000	1024 ----- 0000000000000000e000 0fd:00000 libselinux.so.1	
00000030c460e000	4 读写--- 000000000000e000 0fd:00000 libselinux.so.1	
00000030c460f000	4 rw--- 00000030c460f000 000:00000 [匿名]	
0000007fbfffc000	16 rw--- 0000007fbfffc000 000:00000 [堆栈]	
ffffffffffff600000 映射:	8192 ----- 0000000000000000 000:00000 [匿名]	
12944K	可写/私有:248K	共享:0K

一些最重要的信息位于显示屏底部。该行显示：

映射： 映射到进程中使用的文件的内存总量

可写/私有：此过程占用的私有地址空间量

共享： 该进程与其他进程共享的地址空间量

您还可以查看存储信息的地址空间。在 32 位和 64 位系统上执行pmap命令时,您会发现一个有趣的区别。要查看pmap命令的完整语法,请执行以下命令：

pmap -?

2.3.11 netstat

netstat是最流行的工具之一。如果您从事网络工作,您应该对它很熟悉。它显示许多与网络相关的信息,例如套接字使用情况、路由、接口、协议、网络统计信息等等。以下是一些基本选项：

-a	显示所有套接字信息
-r	显示路由信息
-i	显示网络接口统计信息
-s	显示网络协议统计信息

还有许多其他有用的选项。请查阅手册页。以下示例显示了套接字信息的示例输出。

示例 2-21 使用 netstat 显示套接字信息

```
[root@lnxsu5 ~]# netstat -natuw
活动的互联网连接（服务器和已建立的连接）
Proto Recv-Q Send-Q 本地地址 外部地址 状态
tcp        0 0 0.0.0.0:111 0.0.0.* 监听
tcp 0 0 127.0.0.1:25 0.0.0.* 监听
tcp 0 0 127.0.0.1:2207 0.0.0.* 监听
tcp 0 0 127.0.0.1:36285 127.0.0.1:12865 TIME_WAIT
tcp 0 0 10.0.0.5:37322 10.0.0.4:33932 TIME_WAIT
tcp 0 1 10.0.0.5:55351 10.0.0.4:33932 SYN_SENT
tcp 0 1 10.0.0.5:55350 10.0.0.4:33932 LAST_ACK
tcp 0 0 10.0.0.5:64093 10.0.0.4:33932 TIME_WAIT
tcp 0 0 10.0.0.5:35122 10.0.0.4:12865 已建立
tcp 0 0 10.0.0.5:17318 10.0.0.4:33932 TIME_WAIT
tcp 0 0 ::22 ::* 倾听
tcp 0 2056 ::ffff:192.168.0.254:22 ::ffff:192.168.0.1:3020 已建立
udp 0 0 0.0.0.0:111 udp 0 0 0.0.0:631 udp 0
0 ::5353                                         0.0.0.*:
                                                ::*:
```

套接字信息

原始	套接字使用的协议 (tcp、udp、raw)。
接收-Q	连接到此套接字的用户程序未复制的字节数。
发送-Q	远程主机未确认的字节数。
本地地址	套接字本地端的地址和端口号。除非指定 --numeric (-n) 选项,套接字地址解析为其规范主机名 (FQDN),并且端口号转换为相应的服务名称。
外部地址	套接字远端的地址和端口号。
状态	套接字的状态。由于原始模式下没有状态,并且 UDP 中通常不使用状态,因此此列可以留空。 有关可能的状态,请参见第 32 页的图 1-28 和手册页。

2.3.12 iptraf

iptraf实时监控 TCP/IP 流量并生成实时报告。它显示每个会话、接口和协议的 TCP/IP 流量统计信息。iptraf实用程序由iptraf软件包提供。

iptraf向我们提供如下报告：

- IP流量监控 :通过TCP连接进行网络流量统计
- 通用接口统计 :按网络接口统计IP流量
- 详细接口统计 :按协议统计网络流量
- 统计细分 :按 TCP/UDP 端口和数据包大小进行的网络流量统计
- LAN站点监控 :按二层地址统计网络流量

以下是iptraf生成的一些报告。

图 2-2 iptraf 按协议输出的 TCP/IP 统计信息

图 2-3 iptraf 按数据包大小输出的 TCP/IP 流量统计信息

2.3.13 tcpdump / ethereal

`tcpdump`和`letworker`用于捕获和分析网络流量。这两个工具都使用`libpcap`库来捕获数据包。它们监控处于混杂模式的网络适配器上的所有流量，并捕获适配器接收到的所有帧。要捕获所有数据包，应以超级用户权限执行这些命令，以使接口处于混杂模式。

您可以使用这些工具深入研究与网络相关的问题。您可以发现 TCP/IP 重传、窗口大小缩放、名称解析问题、网络配置错误等等。但请记住，这些工具只能监控网络适配器接收的帧，而不能监控整个网络流量。

tcpdump

tcpdump 是一款简单但功能强大的实用程序。它具有基本的协议分析功能，可让您大致了解网络状况。tcpdump 支持多种选项和灵活的表达式来过滤要捕获的帧（捕获过滤器）。我们将在下面进行介绍。

选项：

- i <接口> 网络接口
- e 打印链接级标题
- s <快照长度> 从每个数据包中捕获 <snaplen> 字节
- n 避免 DNS 查找
- w <文件> 写入文件
- r <文件> 从文件读取
- v, -vv, -vvv Vervose 输出

捕获过滤器的表达式：

关键词：

主机 dst、src、端口、src 端口、dst 端口、tcp、udp、icmp、网络、dst 网络、src 网络等

可以使用以下方式组合原语：

- 否定（“!”或“不”）
- 连接（“&&”或“and”）
- 交替（“||”或“或”）

一些有用的表达式的示例：

DNS 查询数据包

tcpdump -i eth0 udp 端口 53

FTP 控制和 FTP 数据会话至 192.168.1.10

tcpdump -i eth0 dst 192.168.1.10 和 (端口 ftp 或 ftp-data)

与 192.168.2.253 的 HTTP 会话

tcpdump -ni eth0 dst 192.168.2.253 和 tcp 和端口 80

与子网 192.168.2.0/24 的 Telnet 会话

tcpdump -ni eth0 dst 网络 192.168.2.0/24 和 tcp 以及端口 22

源和目标不在子网 192.168.1.0/24 中，且 TCP SYN 或 TCP FIN 标志处于打开状态（TCP 建立或终止）的数据包
`tcpdump -tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net 192.168.1.0/24`

示例 2-22 tcpdump 输出示例

```
21:11:49.555340 10.1.1.1.2542 > 66.218.71.102.http:S 2657782764:2657782764 (0)win 65535 <mss 1460,nop,nop,sackOK> (DF)21:11:49.671811
66.218.71.102.http > 10.1.1.1.2542:S 2174620199:2174620199 (0)ack 2657782765 win 65535 <mss 1380> 21:11:51.211869 10.1.1.18.2543>
216.239.57.99.http: S 2658253720:2658253720(0) win 65535 <mss 1460,nop,nop,sackOK> (DF) 21:11:51.332371 216.239.57.99.http > 10.1.1.1.2543: S
3685788750:3685788750(0) ack 2658253721 win 8190 <mss 1380> 21:11:56.972822 10.1.1.1.2545 > 129.42.18.99.http: S 2659714798:2659714798(0)
win 65535 <mss 1460,nop,nop,sackOK> (DF) 21:11:57.133615 129.42.18.99.http > 10.1.1.1.2545:S 2767811014:2767811014(0) ack 2659714799 win
65535 <mss 1348> 21:11:57.656919 10.1.1.1.2546 > 129.42.18.99.http:S 2659939433:2659939433(0) win 65535 <mss 1460,nop,nop,sackOK>
(DF)21:11:57.818058 129.42.18.99.http > 9.116.198.48.2546:S 1261124983:1261124983 (0)ack 2659939434 win 65535 <mss 1348>
```

请参阅手册页以了解更多详细信息。

ethereal 的功

能与tcpdump类似,但更加复杂,并具有高级协议分析和报告功能。它还具有 GUI 界面和使用ethereal命令的命令行界面,该命令是 ethereal 软件包的一部分。

与tcpdump类似,它可以使用捕获过滤器,并且还支持显示过滤器。它可以用来缩小帧范围。以下是一些有用的表达式示例:

```
ip.version == 6 且 ip.len > 1450 ip.addr ==
129.111.0.0/16 ip.dst 等于
www.example.com 且 ip.src == 192.168.1.1 而不是 ip.addr 等于 192.168.4.1
```

TCP/UDP

```
tcp.port eq 22
tcp.port == 80 且 ip.src == 192.168.2.1 tcp.dstport == 80
且 (tcp.flags.syn == 1 或 tcp.flags.fin == 1) tcp.srcport == 80 且 (tcp.flags.syn == 1 且
tcp.flags.ack == 1) tcp.dstport == 80 且 tcp.flags == 0x12 tcp.options.mss_val == 1460 且
tcp.option.sack == 1
```

应用

```
http.request.method == "POST"
smb.path包含\\SERVER\\SHARE
```

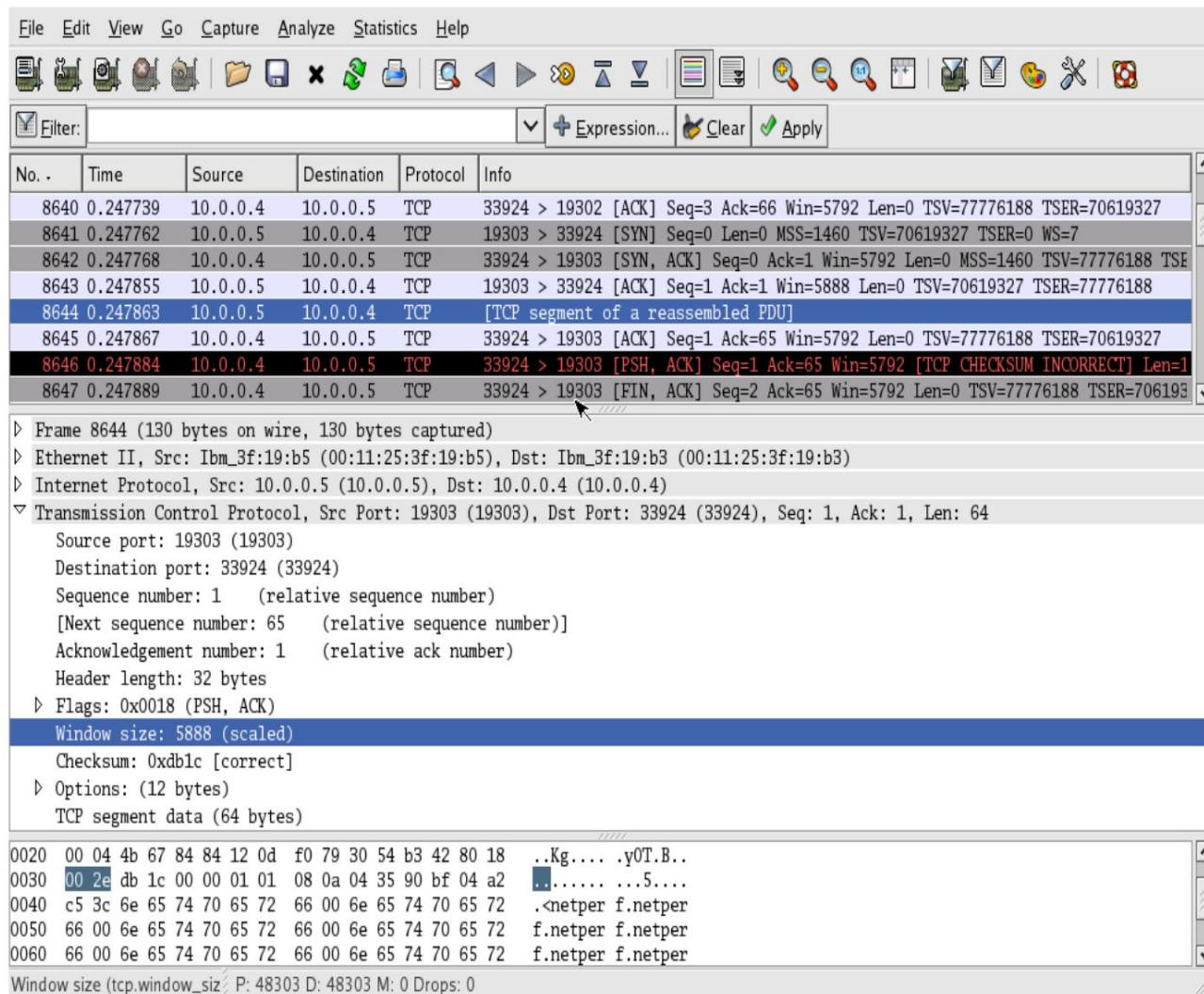


图 2-4 EtherNet/IP 图形用户界面

2.3.14 nmon

nmon 是Nigel's Monitor 的缩写,是由 Nigel Griffiths 开发的一款用于监控 Linux 系统性能的流行工具。由于 nmon 整合了多个子系统的性能信息,因此可以将其作为性能监控的单一来源。nmon 可以执行的任务包括处理器利用率、内存利用率、运行队列信息、磁盘 I/O 统计信息、网络 I/O 统计信息、分页活动以及进程指标。

要运行 nmon,只需启动该工具,然后输入一键命令即可选择所需的子系统。例如,要获取 CPU、内存和磁盘统计信息,请启动nmon并输入cmd。

nmon的一个非常有用的特性是能够将性能统计数据保存为逗号分隔值 (CSV) 文件,以便日后分析。nmon 的 CSV 输出可以导入电子表格应用程序,生成图形报告。为此, nmon应使用-f标志启动 (详情请参阅 nmon -h) 。例如,使用第 59 页示例 2-23 中的命令,可以实现运行 nmon 一小时,每 30 秒捕获一次数据快照。

示例 2-23 使用 nmon 记录性能数据

```
# nmon -f -s 30 -c 120
```

上述命令的输出将存储在当前目录中名为 <hostname>_date_time.nmon 的文本文件中。

有关 nmon 的更多信息,我们建议您访问

<http://www-941.haw.ibm.com/collaboration/wiki/display/WikiPtype/nmon>

要下载 nmon,请访问

<http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmonanalyser>

2.3.15 strace

strace命令会拦截并记录进程调用以及进程接收的信号。它是一个非常有用的诊断、指导和调试工具。系统管理员发现它在解决程序问题方面非常有用。

要跟踪某个进程,请指定要监控的进程 ID (PID):

strace -p <进程号>

示例 2-24 显示了 strace 输出的示例。

示例 2-24 strace 监控 httpd 进程的输出

```
[root@x232 html]# strace -p 815
进程 815 已连接 - 中断退出
semop(360449, 0xb73146b8, 1) = 0
poll([{fd=4, events=POLLIN}, {fd=3, events=POLLIN, revents=POLLIN}], 2, -1) = 1
接受 (3,{sa_family=AF_INET,sin_port=htons (52534),sin_addr=inet_addr ("192.168.1.1") },[16])=13
semop(360449, 0xb73146be, 1) = 0
getsockname(13,{sa_family=AF_INET,sin_port=htons(80),sin_addr=inet_addr(" 192.168.1.2 " )},[16])=0
fcntl64(13,F_GETFL)=0x2 (标志O_RDWR)
fcntl64(13, F_SETFL, O_RDWR|O_NONBLOCK)=0
读取 (13,0x8259bc8,8000)轮询 = -1 EAGAIN (资源暂时不可用)
 ([{fd=13,events=POLLIN,revents=POLLIN}],1,300000)=1
读取 (13, "GET /index.html HTTP/1.0\r\nUser-Agent: ...",8000)= 91
获取日期时间 ({1084564126,750439},NULL)=0
stat64( /var/www/html/index.html ,{st_mode=S_IFREG|0644,st_size=152,...})=0
打开 ( "/var/www/html/index.html" ,O_RDONLY)=14
mmap2(NULL,152,PROT_READ,MAP_SHARED,14,0)=0xb7052000
writev(13,[{"HTTP/1.1 200 OK\r\n日期: 星期五, 14 M ... ,264},{<html>\n<title>\n 152}], 2)=416           RedPaper" ...
munmap (0xb7052000, 152) = 0
socket (PF_UNIX,SOCK_STREAM,0) = 15
connect (15,{sa_family=AF_UNIX,path= /var/run/.nscd_socket },110)=-1 ENOENT (没有此文件或目录)
关闭(15) = 0
```

注意:当strace命令针对某个进程运行时,PID 的性能会大幅降低,因此只能在数据收集时运行。

这里还有另一个有趣的用途。这个命令报告每次系统调用执行一个命令在内核中消耗了多少时间。

strace -c <命令>

示例 2-25 strace 计数系统时间的输出

% 时间	秒数 usecs/调用		调用	错误
25.12	0.026714	12	2203	getdents64
25.09	0.026689	8	3302	lstat64
17.20	0.018296	8	2199	更改目录
9.05	0.009623	9	1109	打开
8.06	0.008577	8	1108	关闭
7.50	0.007979		1108	fstat64
7.36	0.007829		1100	fcntl64
0.19	0.000205	77		执行
0.13	0.000143	205	1	读
0.08	0.000084	24	6	old_mmap
0.05	0.000048	11		mmap2
0.04	0.000040	10		蒙玛普
0.03	0.000035	13		写
0.02	0.000024	35		1 次访问
0.02	0.000020	12		mprotect
0.02	0.000019			布克
0.01	0.000014			调用目录
0.01	0.000009			时间
0.01	0.000007			名字
0.01	0.000007	1067977	85312232111	设置线程区域
100.00	0.106362		12165	共 1 个

要获取strace命令的完整语法,请发出:

strace -?

2.3.16 Proc文件系统

proc 文件系统并非真正的文件系统,但它却极其有用。它并非用于存储数据,而是为正在运行的内核提供接口。proc 文件系统使管理员能够实时监控和修改内核。第 61 页的图 2-5 展示了一个 proc 文件系统示例。大多数 Linux 性能测量工具都依赖于 /proc 提供的信息。

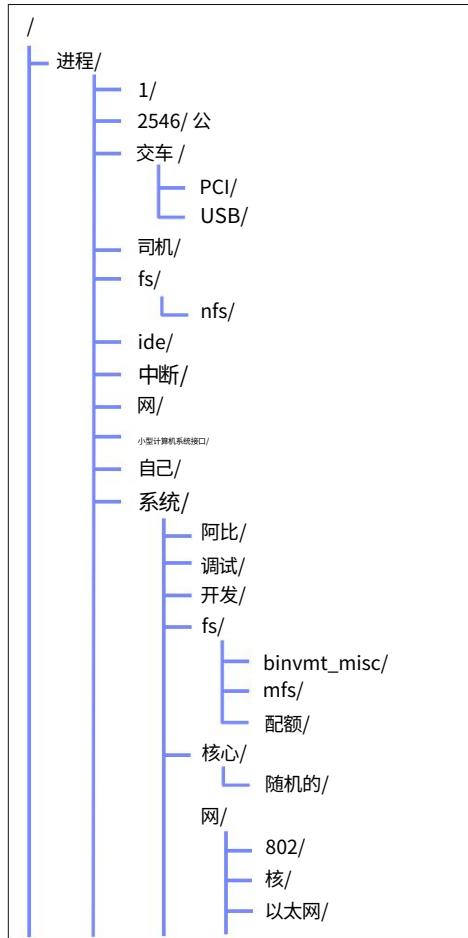


图 2-5 /proc 文件系统示例

观察 proc 文件系统,我们可以区分出几个用于不同用途的子目录,但由于 proc 目录中的大部分信息不易阅读,因此建议您使用 vmstat 等工具以更易读的方式显示各种统计信息。请记住,proc 文件系统的布局和包含的信息在不同的系统架构中会有所不同。

/proc 目录中的文件

proc 根目录中的各种文件涉及几个相关的系统统计信息。

在这里您可以找到 Linux 工具 (例如 vmstat 和 cpuinfo) 获取的信息作为其输出的来源。

数字 1 到 X

各个子目录用数字表示,它们代表正在运行的进程或其各自的进程 ID (PID)。目录结构始终以 PID 1 开头,代表 init 进程,依次向上,直到相应系统上运行的 PID 数量。

每个编号的子目录存储与进程相关的统计信息。例如,进程映射的虚拟内存就是此类数据。

交流电源接口

ACPI 是指大多数现代台式机和笔记本电脑系统支持的高级配置和电源接口。由于 ACPI 主要是一项 PC 技术,因此在服务器系统上经常被禁用。有关 ACPI 的更多信息,请参阅:

<http://www.apci.info>

公共汽车

该子目录包含有关总线子系统（例如 PCI 总线或相应系统的 USB 接口）的信息。

中断请求

irq 子目录包含系统中中断的信息。该目录下的每个子目录都指向一个中断，并且可能指向一个连接的设备，例如网卡。在 irq 子目录中，你可以更改给定中断的 CPU 亲和性（本书后面会介绍此功能）。

网

net 子目录包含大量有关网络接口的原始统计信息，例如接收到的多播数据包或每个接口的路由。

小型计算机系统接口

此子目录包含相应系统的 SCSI 子系统信息，例如连接的设备或驱动程序版本。子目录 ips 指的是大多数 IBM System x 服务器上的 IBM ServeRAID 控制器。

系统

在 sys 子目录中，您可以找到可调的内核参数，例如虚拟内存管理器或网络堆栈的行为。我们将在第 104 页的 4.3 节“更改内核参数”中介绍 /proc/sys 中的各种选项和可调值。

终端

tty 子目录包含有关系统各个虚拟终端以及它们所连接的物理设备的信息。

2.3.17 KDE 系统防护

KDE System Guard (KSysguard) 是 KDE 的任务管理器和性能监视器。它采用客户端/服务器架构，可以监控本地和远程主机。

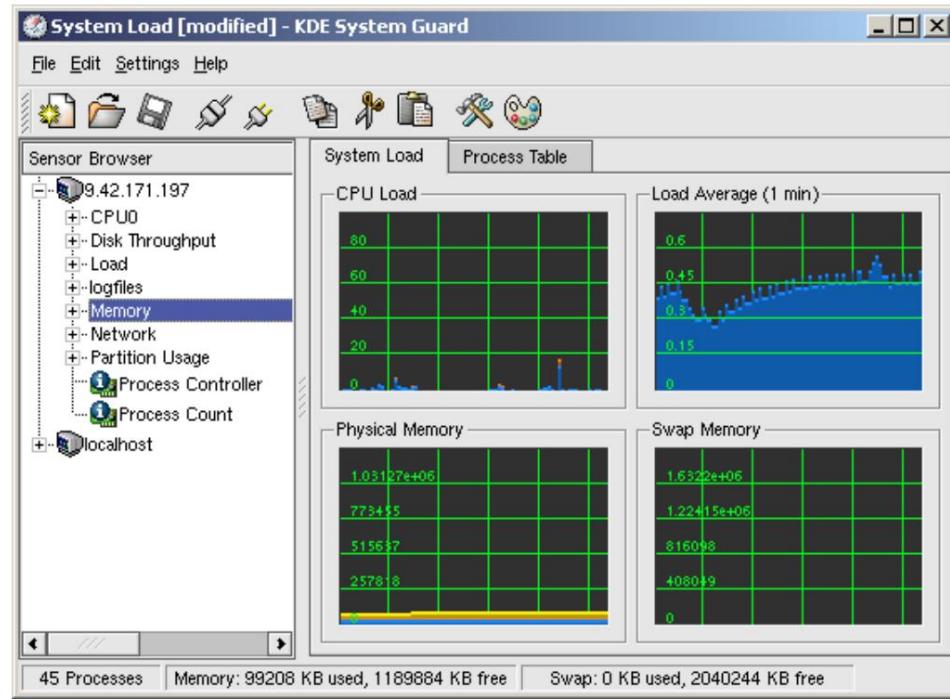


图 2-6 默认 KDE System Guard 窗口

图形前端（图 2-6）使用传感器检索其显示的信息。传感器可以返回简单值，也可以返回更复杂的信息（例如表格）。每种类型的信息都提供一个或多个显示。这些显示以工作表的形式组织，可以彼此独立地保存和加载。

KSysguard 主窗口由菜单栏、可选工具栏和状态栏、传感器浏览器以及工作区组成。首次启动时，您会看到默认设置：您的本地计算机在传感器浏览器中显示为 localhost，工作区中显示两个选项卡。

每个传感器监测一个特定的系统值。所有显示的传感器都可以拖放到工作区中。有三个选项：

- 您可以在实际工作区中删除和替换传感器。
- 您可以编辑工作表属性并增加行数和列数。
- 您可以创建一个新的工作表并放置满足您需求的新传感器。

工作区

第 64 页的图 2-7 中的工作区显示了两个选项卡：

系统负载，首次启动 KSysguard 时的默认视图
进程表

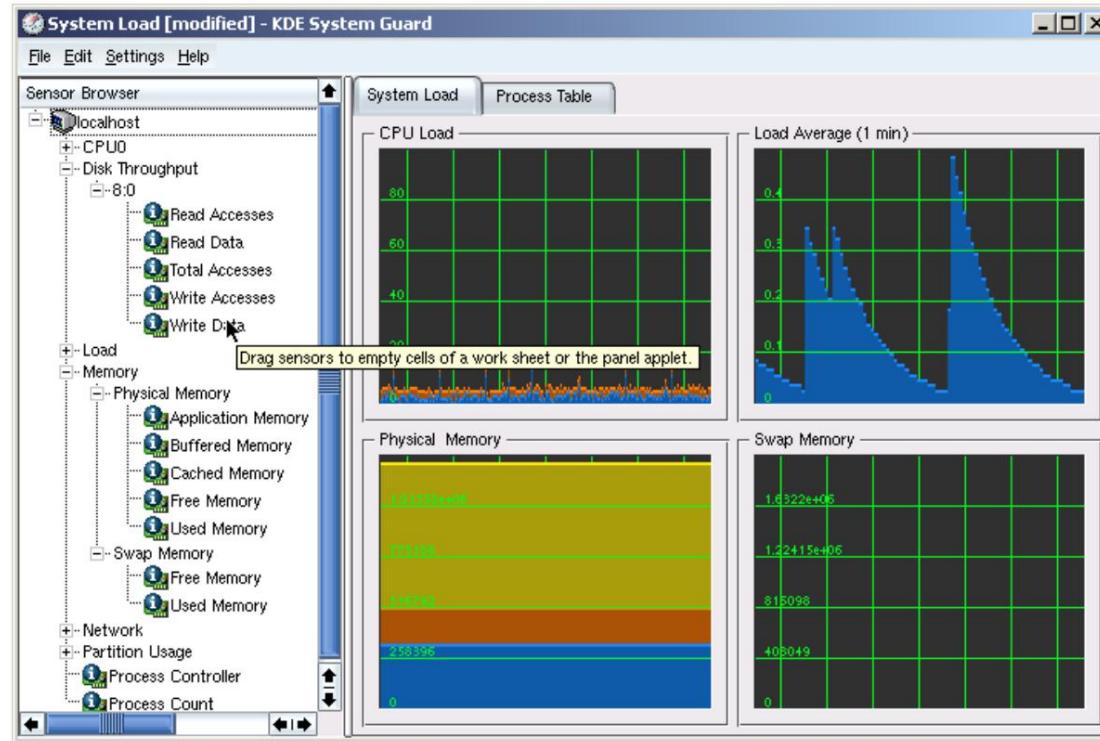


图 2-7 KDE System Guard 传感器浏览器

系统负载

系统负载工作表显示四个传感器窗口:CPU 负载、平均负载（1 分钟）、物理内存和交换内存。一个窗口中可以显示多个传感器。要查看窗口中正在监控哪些传感器,请将鼠标悬停在图表上,然后会显示描述性文字。您也可以右键单击图表,然后点击“属性”,然后点击

传感器选项卡 (图 2-8)。这也显示了图表上每种颜色所代表的含义。

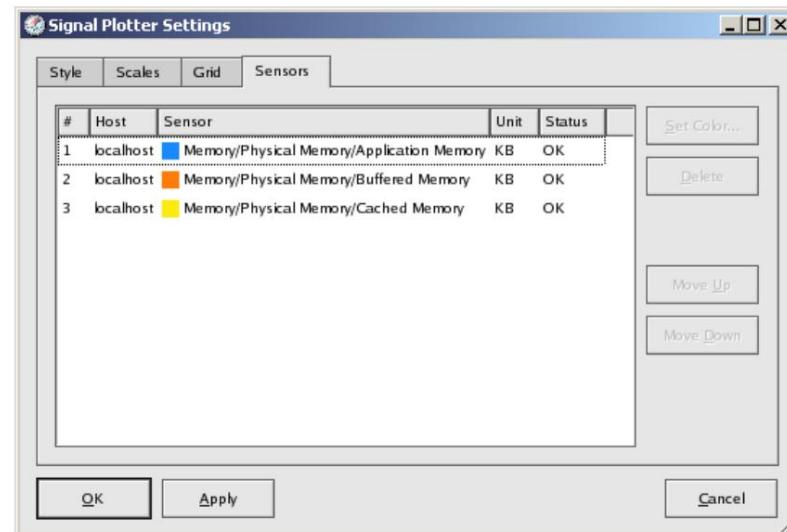


图 2-8 传感器信息、物理内存信号绘图仪

进程表

点击“进程表”选项卡,显示服务器上所有正在运行的进程的信息(图 2-9)。默认情况下,该表按系统 CPU 利用率排序,但可以通过点击其他标题进行更改。

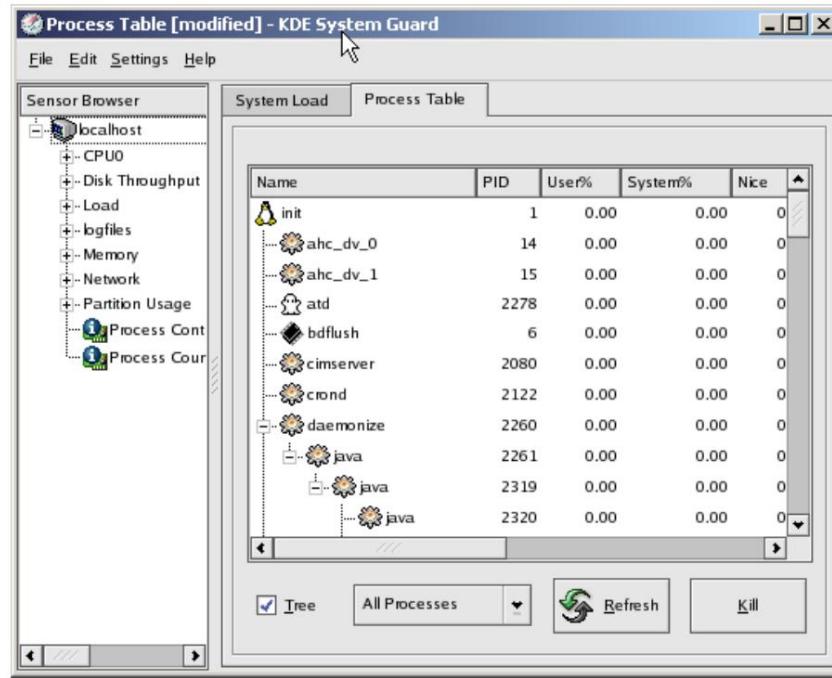


图 2-9 进程表视图

配置工作表

对于您的环境或您希望监控的特定区域,您可能需要使用不同的传感器进行监控。最好的方法是创建自定义工作表。在本节中,我们将指导您完成创建第 67 页图 2-12 所示工作表所需的步骤:

1. 点击文件→新建,创建一个空白工作表,打开图 2-10 所示的窗口。

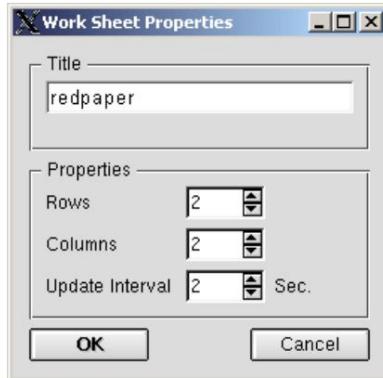


图 2-10 新工作表的属性

2. 输入标题以及行数和列数;这将决定监视窗口的最大数量,在本例中为四个。信息完成后,单击“确定”创建空白工作表,如第 66 页的图 2-11 所示。

注意:可以定义的最快更新间隔是两秒。

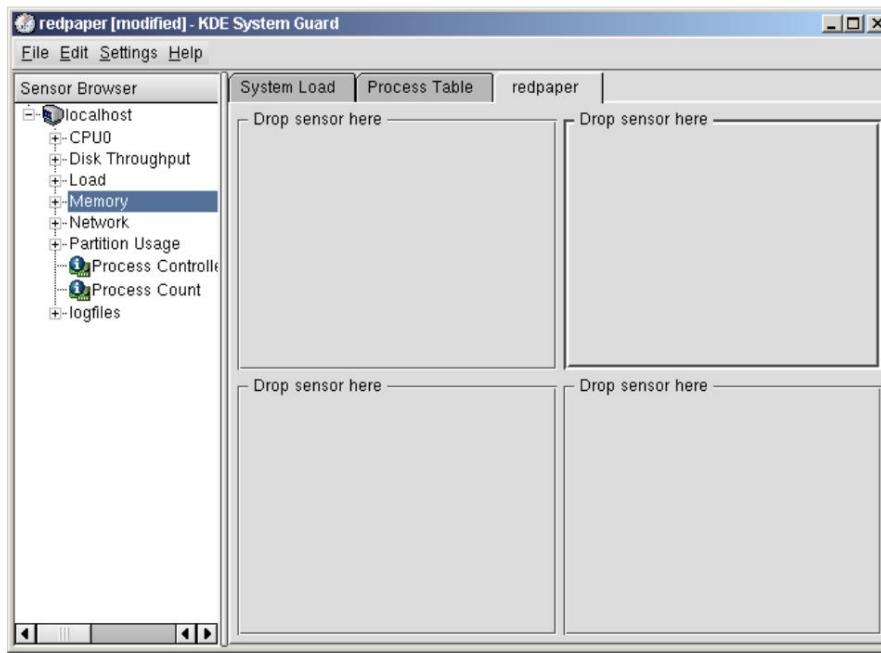


图2-11 空工作表

3. 将窗口左侧的传感器拖到

在右侧的框中,选择所需的显示类型。显示类型包括:

- **信号绘图仪**:显示一个或多个传感器随时间变化的样本。如果多个传感器的数值会以不同的颜色分层显示。如果显示屏足够大,则会显示网格以显示绘制样本的范围。

默认情况下,自动量程模式处于启用状态,因此最小值和最大值将自动设置。如果您需要固定的最小值和最大值,可以停用自动量程模式,并在“属性”对话框窗口(右键单击图表即可访问)中的“比例”选项卡中设置值。

- **万用表**:以数字仪表形式显示传感器值。在“属性”对话框中,您可以指定下限和上限。如果超出范围,则显示颜色为警报颜色。

- **条形图**:以跳动的条形显示传感器值。在“属性”对话框中,您可以指定范围的最小值和最大值以及下限和上限。如果超出范围,则显示警报颜色。

- **传感器记录器**:它不显示任何值,但将它们与附加日期和时间信息一起记录在文件中。

对于每个传感器,您必须定义一个目标日志文件、传感器记录的时间间隔以及是否启用警报。

4. 单击文件→保存,将更改保存到工作表。

注意:当您保存工作表时,它将保存在用户的主目录中,这可能会阻止其他管理员使用您的自定义工作表。

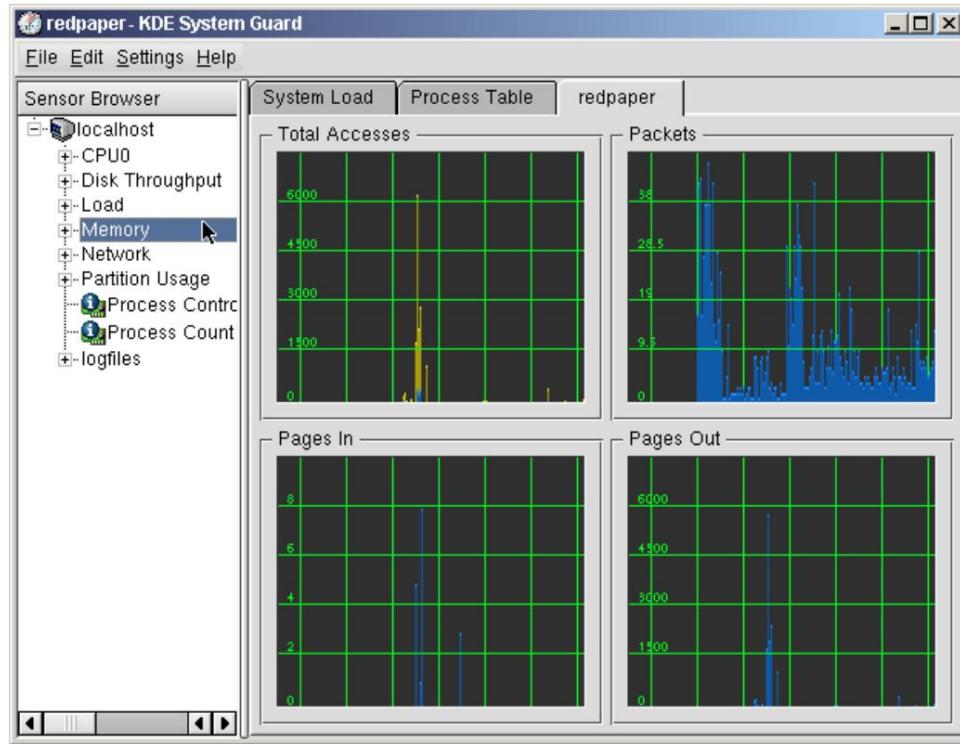


图 2-12 示例工作表

了解有关 KDE System Guard 的更多信息,请访问: <http://docs.kde.org/>

2.3.18 Gnome 系统监视器

虽然不如 KDE System Guard 强大,但 Gnome 桌面环境提供了一个图形性能分析工具。Gnome 系统监视器可以以图表形式显示与性能相关的系统资源,以便直观地了解可能出现的峰值和瓶颈。请注意,所有统计信息都是实时生成的。长期性能分析应使用其他工具进行。

2.3.19 容量管理器

容量管理器 (Capacity Manager) 是 IBM Director 系统管理套件 (适用于 IBM Systems) 的附加组件,包含在适用于 IBM System x 系统的 ServerPlus Pack 中。容量管理器提供了跨多个系统和平台进行长期性能测量的可能性。容量管理器支持容量规划,可为您提供未来所需系统容量的估算。使用容量管理器,您可以将报告导出为 HTML、XML 和 GIF 文件,这些文件可自动存储在内联网 Web 服务器上。IBM Director 可在不同的操作系统平台上使用,这使得在异构环境中收集和分析数据变得更加容易。《IBM System x 服务器性能调优》(SG24-5287) 中详细介绍了容量管理器。

要使用容量管理器,必须在需要使用其高级功能的系统上安装相应的 RPM 包。安装 RPM 包后,在 IBM Director Console 中选择“容量管理器” → “Monitor Activator”。



图2-13 IBM Director Console中的任务列表

将 Monitor Activator 图标拖放到已安装 Capacity Manager 软件包的单个系统或一组系统上。此时将打开一个窗口（图 2-14），您可以在其中选择要随时间监控的各个子系统。Linux 版 Capacity Manager 尚不支持所有可用的性能计数器功能。系统统计信息仅限于一些基本的性能参数子集。

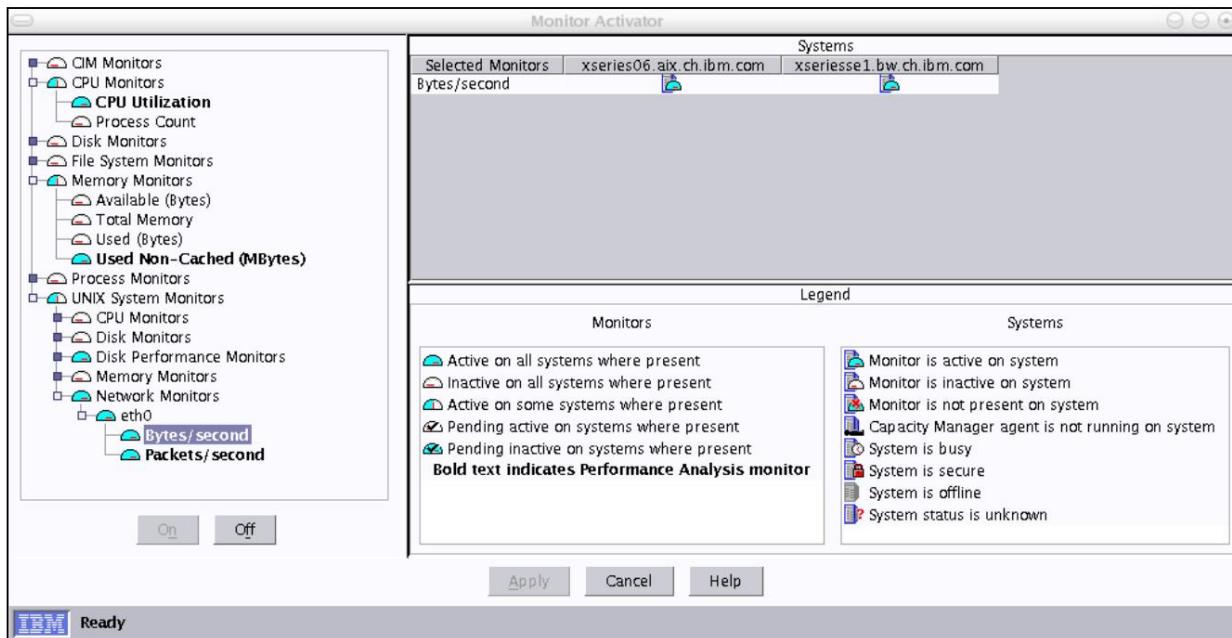


图2-14 激活性能监视器多个系统

Monitor Activator 窗口右侧显示各个系统及其当前状态，左侧显示不同的可用性能监视器。要添加新监视器，请选择该监视器并点击“开启”。更改将在 Monitor Activator 窗口关闭后立即生效。完成此步骤后，IBM Director 会开始收集所需的性能指标，并将其存储在不同系统上的临时位置。

要创建收集到的数据的报告，请选择“容量管理器”→“报告生成器”（参见图 2-13），并将其拖到您想要查看性能统计数据的单个系统或一组系统上。IBM Director 会询问您是立即生成报告还是安排稍后执行（参见第 69 页的图 2-15）。

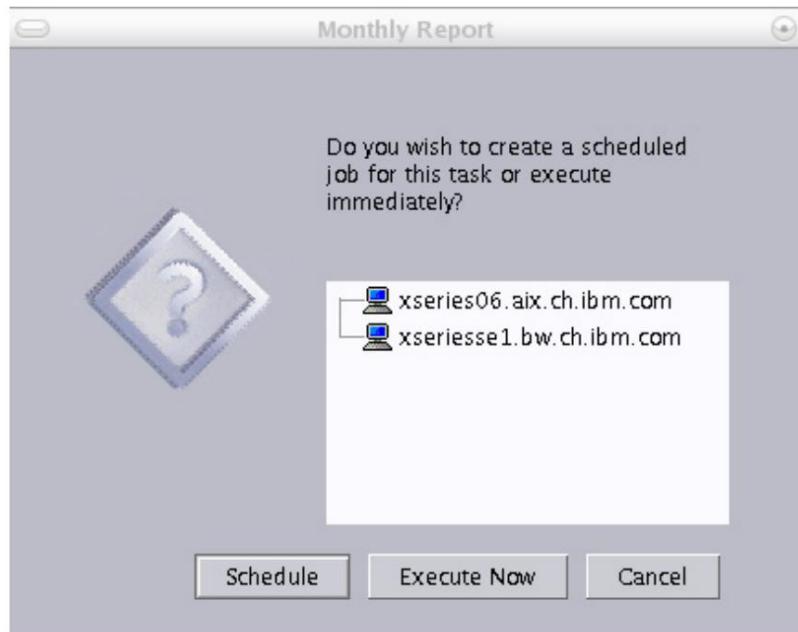


图2-15 调度报表

在生产环境中,让容量管理器定期生成报告是一个好主意。根据我们的经验,在周末非工作时间执行的每周报告可能非常有价值。系统会根据您的选择生成立即执行或计划执行的报告。报告完成后,会立即存储在中央 IBM Director 管理服务器上,您可以使用“报告查看器”任务进行查看。第 70 页的图 2-16 显示了容量管理器月度报告的示例输出。

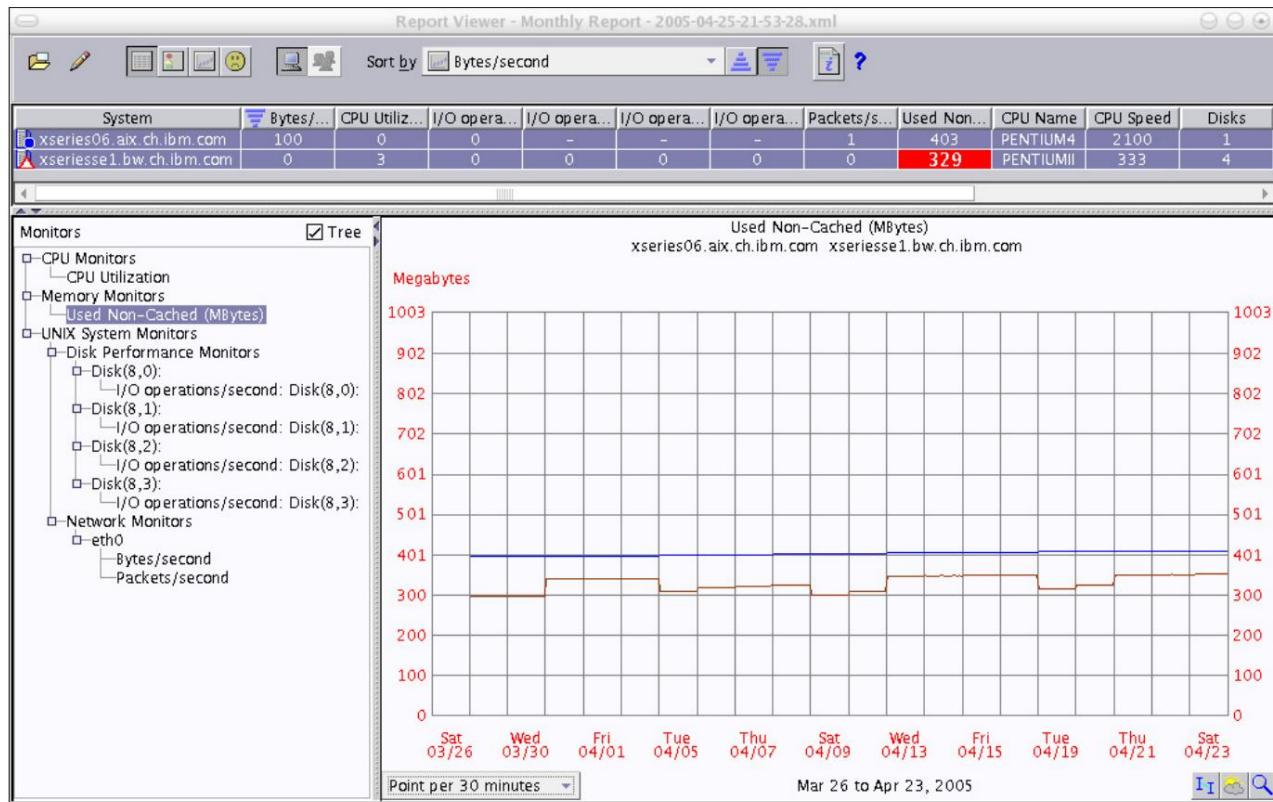


图 2-16 容量管理器报告示例

报告查看器窗口可让您选择收集的不同性能计数器，并将这些数据与单个系统或选定的系统关联。

容量管理器获取的数据可以导出为 HTML 或 XML 文件，以便在内联网 Web 服务器上显示或用于将来的分析。

2.4 基准测试工具

本节我们将讨论主要的基准测试工具。要衡量性能，使用优秀的基准测试工具是明智之举。市面上有很多优秀的工具。其中一些工具具备以下全部或部分功能：

- 负载生成
- 监控性能
- 监控系统利用率
- 报告

基准测试只不过是针对特定工作负载的模型，该模型可能接近或不接近系统实际运行的工作负载。即使某个系统的 Linpack 得分很高，它也可能不是理想的文件服务器。您应该始终记住，基准测试无法模拟最终用户有时难以预测的反应。基准测试无法告诉您用户访问数据或备份启动后文件服务器的行为。

一般来说，对任何系统执行基准测试时应遵循以下规则：

使用服务器工作负载的基准：服务器系统具有独特的特性，这使得它们与典型的台式电脑有很大不同，尽管 IBM System x

该平台共享许多适用于台式计算机的技术。服务器基准测试会生成多个线程,以利用系统的对称多处理 (SMP) 功能并模拟真正的多用户环境。PC 启动一个 Web 浏览器的速度可能比高端服务器快,但服务器启动一千个 Web 浏览器的速度却比 PC 快。

模拟预期的工作负载:所有基准测试都有不同的配置选项,您可以使用这些选项来根据系统未来可能运行的工作负载定制基准测试。如果应用程序必须依赖低磁盘延迟,那么再好的 CPU 性能也没什么用。

隔离基准测试系统:如果要使用基准测试系统,则务必尽可能将其与任何其他负载隔离。在开放会话中运行顶级

命令可以极大地影响基准测试的结果。

平均结果:即使您尝试隔离基准测试系统,也可能存在一些未知因素,这些因素会在基准测试时影响系统性能。建议至少运行任何基准测试三次,并计算结果的平均值,以确保一次性事件不会影响您的整个分析。

在以下部分中,我们根据这些标准选择了一些工具:

适用于 Linux:Linux 是基准测试的目标。

适用于所有硬件平台:由于 IBM 提供三种不同的硬件平台 (假设 IBM System p 和 IBM System i™ 的硬件技术均基于 IBM POWER™ 架构),因此选择一个可以在所有架构上无需进行大量移植工作即可使用的基准非常重要。

开源:Linux 在多个平台上运行,因此如果源代码不可用,二进制文件可能不可用。

文档齐全:进行基准测试时,您必须了解该工具。文档可以帮助您熟悉这些工具。在决定使用某些工具之前,了解其概念、设计和细节也有助于评估该工具是否适合您的需求。

积极维护:旧的废弃工具可能不符合最新的规范和技术。它可能会得出错误的结果。

广泛使用:您可以找到大量有关广泛使用的工具的信息。

易于使用:您需要一个易于使用的工具。

报表能力:拥有报表能力将大大减少性能分析的工作量。

2.4.1 LMbench

LMbench 是一套微基准测试工具,可用于分析不同的操作系统设置,例如启用 SELinux 的系统与未启用 SELinux 的系统。LMbench 中包含的基准测试可以测量各种操作系统例程,例如上下文切换、本地通信、内存带宽和文件操作。LMbench 的使用非常简单,只需了解三个重要命令:

制作结果:第一次运行 LMbench 时,它会提示一些系统配置的详细信息以及它应该执行的测试。

make rerun:在初始配置和第一次基准测试运行之后,使用 make rerun 命令只是使用 make results 运行期间提供的配置重复基准测试。

`make see`:至少运行三次后,可以使用 `make see` 命令查看结果。结果将显示出来,并可复制到电子表格应用程序中,以便进一步分析或以图形方式呈现数据。

LMbench 基准测试可以在<http://sourceforge.net/projects/lmbench/>找到

2.4.2 IO区域

IOzone 是一个文件系统基准测试,可用于模拟各种不同的磁盘访问模式。由于 IOzone 的配置选项非常详细,因此可以精确模拟目标工作负载配置文件。IOzone 会使用可变的块大小写入一个或多个大小可变的文件。

虽然 IOzone 提供了非常舒适的自动基准测试模式,但定义文件大小、I/O 大小和访问模式等工作负载特征通常更有效。

如果要评估文件系统是否适合数据库工作负载,那么让 IOzone 对大文件以较大的块大小创建随机访问模式是合理的,而不是对小块大小的大文件进行流式传输。IOzone 的一些最重要的选项包括:

- b <输出.xls> 告诉 IOzone 将结果存储在与 Microsoft® Excel® 兼容的电子表格中。
- C 显示每个子进程的输出(可用于检查所有子进程是否真的同时运行)。
- f <文件名> 可用于告诉 IOzone 将数据写入何处。
- i <测试数量>此选项用于指定要运行的测试。首次写入测试文件时,必须指定 -i 0。
 - 有用的测试是 -i 1 用于流读取, -i 2 用于随机读取和随机写入访问,以及 -i 8 用于混合随机的工作负载使用权。
- h 显示屏幕帮助。
- r 告诉 IOzone 应该使用什么记录或 I/O 大小进行测试。
 - 记录大小应尽可能接近目标工作负载将使用的记录大小。
- k <异步 I/O 数量> 使用内核 2.6 的异步 I/O 特性,该特性常被 IBM DB2® 等数据库使用。
- m 如果目标应用程序使用多个内部缓冲区,则可以使用 -m 标志模拟此行为。
- s <大小 (KB)> 指定基准测试的文件大小。对于异步文件系统(大多数文件系统的默认挂载选项),IOzone 应使用至少两倍于系统内存的文件大小,以便准确衡量磁盘性能。文件大小也可以使用 m 或 g 分别以 MB 或 GB 为单位指定,具体单位请直接在文件大小后加上。
- +你 是一个实验性的开关,可用于测量测试期间的处理器利用率。

注意:任何使用适合系统内存且存储在异步文件系统上的文件的基准测试都将测量内存吞吐量,而不是磁盘子系统的性能。因此,您应该使用同步选项挂载目标文件系统,或者使用大约两倍于系统内存大小的文件。

使用 IOzone 测量安装在 /perf 上的给定磁盘子系统对 10 GB 大小的文件以 32 KB I/O 大小的随机读取性能（这些特性模拟了一个简单的数据库）如下所示：

示例 2-26 IOzone 命令行示例

```
./iozone -b 结果.xls -R -i 0 -i 2 -f /perf/iozone.file -r 32 -s 10g
```

最后，可以将得到的结果导入到您选择的电子表格应用程序中，然后将其转换为图表。使用数据的图形输出可能更容易分析大量数据并识别趋势。示例 2-26 的示例输出可能类似于图 2-17 中显示的图形。

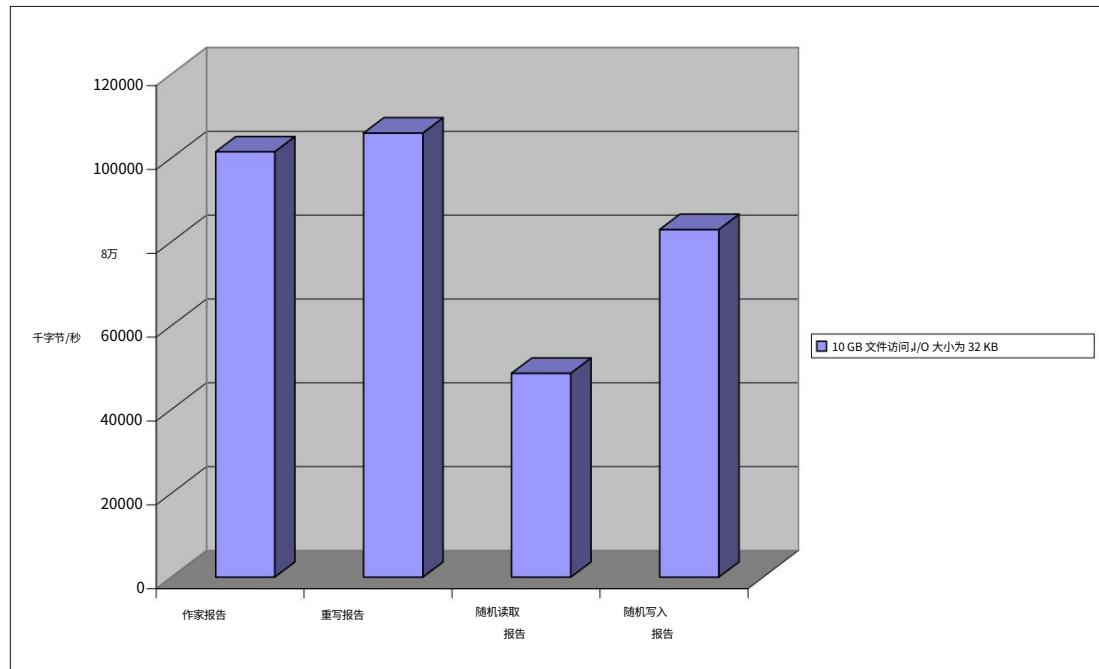


图 2-17 根据示例 2-26 的样本结果生成的图形

如果 IOzone 所用文件大小与系统内存或缓存大小相符，它还可以用于获取一些有关缓存和内存吞吐量的数据。需要注意的是，由于文件系统开销，IOzone 只会报告系统带宽的 70% 到 80%。

IOzone 基准测试可以在<http://www.iozone.org/>找到

2.4.3 netperf

netperf 是一款专注于 TCP/IP 网络性能的性能基准测试工具。它支持 UNIX 域套接字和 SCTP 基准测试。

netperf 基于客户端-服务器模型设计。netserver 在目标系统上运行，netperf 在客户端上运行。netperf 控制 netserver，将配置数据传递给 netserver，生成网络流量，并通过与实际基准测试流量连接分离的控制连接从 netserver 获取结果。在基准测试期间，控制连接上不会发生任何通信，因此不会对结果产生任何影响。netperf 基准测试工具还具有报告功能，包括 CPU 利用率报告。截至撰写本文时，当前稳定版本为 2.4.3。

netperf可以生成多种类型的流量。这些流量基本上分为两类:批量数据传输流量和请求/响应类型流量。需要注意的是,netperf每次只使用一个套接字。netperf的下一个版本(netperf4)将完全支持并发会话的基准测试。届时,我们可以按照下文所述执行多会话基准测试。

批量数据传输

批量数据传输是网络基准测试中最常用的测量指标。批量数据传输以每秒传输的数据量来衡量。它模拟大型文件传输,例如多媒体流和FTP数据传输。

请求/响应类型

这模拟了请求/响应类型的流量,以每秒交换的事务数量来衡量。请求/响应流量类型是在线事务应用程序的典型特征,例如Web服务器、数据库服务器、邮件服务器、文件服务器(用于处理中小型文件)和目录服务器。在实际环境中,除了数据交换之外,还应执行会话的建立和终止。为了模拟这种情况,引入了TCP_CRR类型。

并发会话

netperf在当前稳定版本中并不真正支持并发多会话基准测试,但我们可以发出多个netperf实例来执行一些基准测试,如下所示:

对于 seq 1 10 中的 i, 执行 netperf -t TCP_CRR -H target.example.com -i 10 -P 0 &; 完成

我们将研究一些有用且有趣的选项。

全局选项:

-a	更改远程系统上的发送和接收缓冲区对齐
-b	流测试中的数据包突发
-H <远程主机>	远程主机
-t <测试名称>	测试流量类型
TCP_STREAM	批量数据传输基准
TCP_MAERTS	与 TCP_STREAM 类似,只是流的方向相反。
TCP_SENDFILE	与 TCP_STREAM 类似,但使用 sendfile() 而不是 send()。它会导致零拷贝操作。
UDP_STREAM	与 TCP_STREAM 相同,但使用 UDP。
TCP_RR	请求/响应类型流量基准
TCP_CC	TCP 连接/关闭基准测试。没有交换请求和响应数据包。
TCP_CRR	执行连接/请求/响应/关闭操作。它与禁用 HTTP keepalive 的 HTTP 1.0/1.1 会话非常相似。
UDP_RR	与 TCP_RR 相同,但使用 UDP。
-l <测试长度>	基准测试的测试长度。如果设置为正值,netperf 将在 testlen 秒内执行基准测试。如果设置为负值,则基准测试将持续执行至交换 testlen 字节的数据(对于批量数据传输基准测试)或 testlen 事务(对于请求/响应类型)。
-c	本地 CPU 利用率报告

-C

远程 CPU 利用率报告

注意:某些平台上的 CPU 利用率报告可能不准确。在执行基准测试之前,请确保报告准确无误。

-I <配置级别><间隔>

此选项用于保持结果的置信度。置信度应为 99 或 95 (百分比),间隔 (百分比)可设置。为了使结果保持一定的置信度,netperf 会重复执行相同的基准测试多次。例如, -I 99,5 表示 100 次中有 99 次结果与实际结果的误差在 5% 的区间 (+- 2.5%) 内。

-i <最大值><最小值>

测试迭代的最大和最小次数。此选项限制迭代次数。-i 10,3 表示 netperf 执行相同的基准测试至少 3 次,最多 10 次。如果迭代次数超过最大值,结果将不符合 -I 选项指定的置信度,并且会在结果中显示警告。

-s <字节数>, -S <字节数>

更改本地和远程系统的发送和接收缓冲区大小。这将影响通告窗口大小和有效窗口大小。

TCP_STREAM、TCP_MAERTS、TCP_SENDFILE、UDP_STREAM 的选项

-m <字节>, -M <字节>

分别指定传递给 send()、recv() 函数调用的缓冲区的大小,并控制每次调用发送和接收的大小。

TCP_RR、TCP_CC、TCP_CRR、UDP_RR 的选项:

-r <字节>, -R <字节>

分别指定请求和响应的大小。例如, -r 128,8129 表示 netperf 向网络服务器发送 128 字节的数据包,并将 8129 字节的数据包返回给 netperf。

以下是 netperf 针对 TCP_CRR 类型基准测试的示例输出。

例 2-27 TCP_CRR 基准测试结果示例

使用以下命令行进行测试:

```
/usr/local/bin/netperf -l 60 -H plnxsu4 -t TCP_CRR -c 100 -C 100 -i 3 -I 95,5 -v 1 -- -r 64,1 -s 0 -S 512
```

从 0.0.0.0 (0.0.0.0) 端口 0 AF_INET 到 plnxsu4 (10.0.0.4) 端口 0 AF_INET 的 TCP 连接/请求/响应测试

本地/远程

套接字大小	请求响应已用交易CPU发送接收大小	速率	字节	字节	字节	秒	CPU S.dem	S.dem
-------	-------------------	----	----	----	----	---	-----------	-------

秒%

本地 远程

% 美国/交易 美国/交易

16384	87380	64	2048	1024	1	60.00	3830.65	25.27	10.16	131.928	53.039
-------	-------	----	------	------	---	-------	---------	-------	-------	---------	--------

进行基准测试时,建议使用 netperf 自带的示例测试脚本。通过修改脚本中的一些变量,您可以根据需要执行基准测试。脚本位于 netperf 软件包的 doc/examples/ 目录中。

更多详细信息请参见<http://www.netperf.org/>

2.4.4 其他有用的工具

这里还有一些其他有用的基准测试工具。请记住,您必须了解基准测试工具的特性,以便选择符合您需求的工具。

表 2-3 其他基准测试工具

工具	最有用的工具功能
邦妮	磁盘 I/O 和文件系统基准测试 http://www.textuality.com/bonnie/
邦妮++	磁盘 I/O 和文件系统基准测试 http://www.coker.com.au/bonnie++/
NetBench	文件服务器基准测试。它在 Windows 上运行。
dbench	文件系统基准测试。通常用于文件服务器基准测试。 http://freshmeat.net/projects/dbench/
碘计	磁盘 I/O 和网络基准测试 http://www.iometer.org/
ttcp	简单的网络基准测试
网络传输控制协议	简单的网络基准测试
iperf	网络基准 http://dast.nlanr.net/projects/Iperf/
ab (Apache Bench)	简单的 Web 服务器基准测试。它自带 Apache HTTP 服务器。 http://httpd.apache.org/
WebStone	Web 服务器基准测试 http://www.mindcraft.com/webstone/
Apache JMeter	主要用于Web服务器性能基准测试。它还支持其他协议,例如SMTP、LDAP、JDBC™等,并具有良好的报告功能。 http://jakarta.apache.org/jmeter/
fsstone、smtpstone	邮件服务器基准测试。它们随 Postfix 一起提供。 http://www.postfix.org/
nhfsstone	网络文件系统基准测试。随 nfs-utils 软件包提供。
目录标记	LDAP 基准 http://www.mindcraft.com/directorymark/



3

第 3 章分析性能瓶颈

本章介绍如何查找可能影响服务器的性能问题。我们概述了一系列步骤，引导您找到具体的解决方案，以便将服务器恢复到可接受的性能水平。

本章涵盖的主题包括：

- 3.1, “识别瓶颈” (第 78 页)
- 3.2, “CPU 瓶颈” 第 81 页
- 3.3, “内存瓶颈” 第 82 页
- 3.4, “磁盘瓶颈” 第 84 页
- 3.5, “网络瓶颈” 第 87 页

3.1 识别瓶颈

以下步骤作为我们的快速调整策略：

1. 了解您的系统。
2. 备份系统。
3. 监控并分析系统的性能。
4. 缩小瓶颈范围并找出其原因。
5. 每次尝试一个改变来解决瓶颈问题。
6. 返回步骤 3,直到对系统性能满意为止。

提示:您应该记录每个步骤,特别是您所做的更改及其对性能的影响。

3.1.1 收集信息

最有可能的是,您唯一能获得的第一手信息就是诸如“服务器有问题”之类的陈述。使用探索性问题来澄清和记录问题至关重要。以下是您应该提出的问题列表,以帮助您更好地了解系统。

您能给我提供所讨论的服务器的完整描述吗?

- 模型
- 年龄
- 配置
- 外围设备
- 操作系统版本和更新级别

你能告诉我具体问题是什么吗?

- 有什么症状?
- 描述任何错误消息。

有些人可能难以回答这个问题,但客户提供的任何额外信息都可能有助于你找到问题所在。例如,客户可能会说“我把大文件复制到服务器时速度真的很慢。”这可能表明存在网络问题或磁盘子系统问题。

谁遇到了这个问题?

是某个人、某个特定群体,还是整个组织都遇到了这个问题?这有助于确定问题是否存在于网络的某个特定部分,是否与应用程序相关等等。如果只有一个用户遇到问题,那么问题可能出在该用户的电脑上(或用户的想象)。

客户端对服务器的感知通常是一个关键因素。从这个角度来看,性能问题可能并非与服务器直接相关:服务器和客户端之间的网络路径很容易成为问题的根源。这条路径包括网络设备以及其他服务器(例如域控制器)提供的服务。

问题可以重现吗?

所有可重现的问题都可以解决。如果您对系统有足够的了解,您应该能够缩小问题范围,找到其根源,并决定应该采取哪些措施。

问题可以重现这一事实可以让您更好地观察和理解它。

记录重现问题所需的操作顺序：

- 重现该问题的步骤是什么？

了解这些步骤或许能帮助你在相同条件下，在另一台机器上重现同样的问题。如果成功，你就有机会在测试环境中使用这台机器，从而避免生产服务器崩溃的风险。

- 这是一个间歇性问题吗？

如果问题是间歇性的，首先要做的是收集信息，并找到将问题移至可重现类别的途径。这里的目标是创建一个场景，让问题能够按照指令发生。

- 它是否发生在一天中的特定时间或一周中的特定日子？

这或许能帮你确定问题的原因。问题可能发生在大家上班或午休回来的时候。试着改变发生时间（也就是减少或增加发生频率），这样问题就能重现。

- 这不寻常吗？

如果问题属于不可复现类别，您可能会认为这是特殊情况造成的，并将其归类为已修复。在现实生活中，这种情况很有可能再次发生。

解决难以重现的问题的一个好方法是对服务器执行常规维护：重新启动，或使机器更新驱动程序和补丁。

问题是什么时候开始的？是逐渐出现的，还是突然出现的？

如果性能问题是逐渐出现的，那么很可能是尺寸问题；如果它是一夜之间出现的，那么问题可能是由服务器或外围设备的更改引起的。

服务器是否进行了任何更改（小更改或大更改）或者客户端使用服务器的方式是否有任何变化？

客户是否对服务器或外围设备进行了某些更改，从而导致了问题？是否所有网络更改的日志？

需求可能会根据业务变化而变化，这可能会影响对服务器和网络系统的需求。

是否涉及其他服务器或硬件组件？

有可用的日志吗？

这个问题的优先级是什么？什么时候需要修复？

这个问题需要在接下来的几分钟或几天内解决吗？你可能还有时间去解决；或者现在可能已经到了不得不紧急行动的时候了。

- 问题有多严重？

- 该问题的相关成本是多少？

3.1.2 分析服务器性能

重要提示:在采取任何故障排除措施之前,请备份所有数据和配置信息,以防止部分或全部丢失。

此时,您应该开始监控服务器。最简单的方法是从要分析的服务器运行监控工具。(有关更多信息,请参阅第 2 章“监控和基准测试工具”(第 39 页)。)

应在服务器运行高峰时段(例如,上午 9:00 至下午 5:00)创建服务器性能日志;日志的创建时间取决于服务器提供的服务以及用户。创建日志时,应包含以下对象(如有):

- 处理器
- 系统
- 服务器工作队列
- 记忆
- 页面文件
- 物理磁盘
- 重定向器
- 网络接口

在开始之前,请记住,系统地进行性能调整非常重要。

我们推荐的流程如下,您可以将其用于服务器性能调整过程:

1. 了解影响服务器性能的因素。
2. 测量当前性能以创建性能基线,与未来的测量进行比较并识别系统瓶颈。
3. 使用监控工具识别性能瓶颈。按照下一节的说明,您应该能够将瓶颈缩小到子系统级别。
4. 根据需求执行一些操作来提高服务器性能,从而处理造成瓶颈的组件。

注意:重要的是要明白,当服务器中的其他组件有足够的“功率”来维持较高的性能水平时,通过升级存在瓶颈的组件可以获得最大的收益。

5. 衡量新的绩效。这有助于您比较升级前后的绩效
调整步骤。

当尝试修复性能问题时,请记住以下几点:

应使用适当的优化级别来编译应用程序以减少路径长度。

在升级或修改任何内容之前进行测量,以便了解更改是否产生了效果。(也就是说,进行基线测量。)

检查涉及重新配置现有硬件的选项,而不仅仅是涉及添加新硬件的选项。

3.2 CPU瓶颈

对于主要充当应用程序或数据库服务器的服务器来说,CPU 是一项关键资源,并且经常会成为性能瓶颈的根源。需要注意的是,高 CPU 利用率并不总是意味着 CPU 正忙于工作;它可能正在等待其他子系统。在进行正确的分析时,务必把系统作为一个整体来看待,并考虑所有子系统,因为子系统内部可能会产生级联效应。

注意:人们普遍误以为 CPU 是服务器最重要的部件。事实并非总是如此,服务器通常 CPU 配置过高,而磁盘、内存和网络子系统配置不足。只有真正占用大量 CPU 资源的特定应用程序才能充分利用当今的高端处理器。

3.2.1 查找 CPU 瓶颈

确定 CPU 瓶颈可以通过多种方式实现。正如第 2 章“监控和基准测试工具”(第 39 页)中所述,Linux 提供了多种工具来帮助确定瓶颈。问题在于使用哪些工具。

其中一个工具是`uptime`。通过分析 `uptime` 的输出,我们可以大致了解过去 15 分钟内系统发生的情况。有关此工具的更详细说明,请参阅第 43 页上的 2.3.3 “`uptime`”。

示例 3-1 CPU 受限系统的正常运行时间输出

18:03:16 启动 1 天,2:46,6 个用户,平均负载:182.53,92.02,37.95

使用 KDE System Guard 和 CPU 传感器可以查看当前 CPU 工作负载。

提示:请注意不要同时运行过多工具,以免加剧 CPU 问题。您可能会发现,同时使用大量不同的监控工具可能会导致 CPU 负载过高。

使用`top`命令,您可以查看 CPU 利用率以及哪些进程是问题的主要贡献者(第 41 页的示例 2-1)。如果您已设置`sar`,则会收集大量信息,其中一些是一段时间内的 CPU 利用率。分析这些信息可能很困难,因此请使用`isag`命令,它可以使用`sar`的输出绘制图表。或者,您可能希望通过脚本解析这些信息,并使用电子表格将其绘制成图表,以查看 CPU 利用率的趋势。您也可以在命令行中使用`sar`,方法是输入`sar -u`或

`sar -U`处理器号。为了更广泛地了解系统以及 CPU 子系统以外的其他资源的当前利用率,一个不错的工具是`vmstat`(有关更多信息,请参见第 42 页上的 2.3.2 “`vmstat`”)

3.2.2 SMP

基于 SMP 的系统可能会出现一系列难以检测的有趣问题。在 SMP 环境中,存在“CPU 亲和性”的概念,这意味着需要将进程绑定到 CPU。

之所以有用,主要原因是 CPU 缓存优化,它通过将同一个进程保持在一个 CPU 上而不是在处理器之间移动来实现。当进程在 CPU 之间移动时,新 CPU 的缓存必须刷新。因此,在处理器之间移动的进程会导致多次缓存刷新,这意味着

单个进程需要更长时间才能完成。这种情况很难检测到,因为在监控时,CPU负载看起来非常平衡,不一定在任何一个CPU上达到峰值。亲和性在基于NUMA的系统(例如IBM System x 3950)中也很有用,在这些系统中,保持内存、缓存和CPU访问彼此本地化非常重要。

3.2.3 性能调优选项

第一步是确保系统性能问题是由CPU引起的,而不是其他子系统。如果处理器是服务器的瓶颈,那么可以采取一些措施来提升性能。这些措施包括:

使用ps -ef确保后台没有运行不必要的程序。如果发现此类程序,请将其停止,并使用cron将其安排在非高峰时段运行。

使用top识别非关键、CPU密集型进程,并使用renice修改其优先级。

在基于SMP的机器中,尝试使用任务集将进程绑定到CPU,以确保进程不会在处理器之间跳跃,从而导致缓存刷新。

根据正在运行的应用程序,纵向扩展(更大的CPU)可能比横向扩展(更多CPU)更好。这取决于您的应用程序是否设计为能够有效利用更多处理器。例如,单线程应用程序使用更快的CPU时扩展性会更好,而使用更多CPU时则不会。

常规选项包括确保您使用最新的驱动程序和固件,因为这可能会影响它们对CPU的负载。

3.3 内存瓶颈

在Linux系统上,许多程序会同时运行。这些程序支持多个用户,并且某些进程的使用率高于其他进程。其中一些程序会使用一部分内存,而其余程序则处于“休眠”状态。当应用程序访问缓存时,性能会提升,因为内存访问会检索数据,从而无需访问速度较慢的磁盘。

操作系统使用一种算法来控制哪些程序将使用物理内存,哪些程序将被调出。这对用户程序来说是透明的。页面空间是由操作系统在磁盘分区上创建的文件,用于存储当前未使用的用户程序。通常,页面大小为4KB或8KB。在Linux中,页面大小由include/asm-<architecture>/param.h内核头文件中的变量EXEC_PAGESIZE定义。将进程调出到磁盘的过程称为页面调出(pageout)。

3.3.1 查找内存瓶颈

首先列出服务器上正在运行的应用程序,然后确定每个应用程序运行所需的物理内存和交换空间。第83页的图3-1显示了KDE System Guard监控内存使用情况。

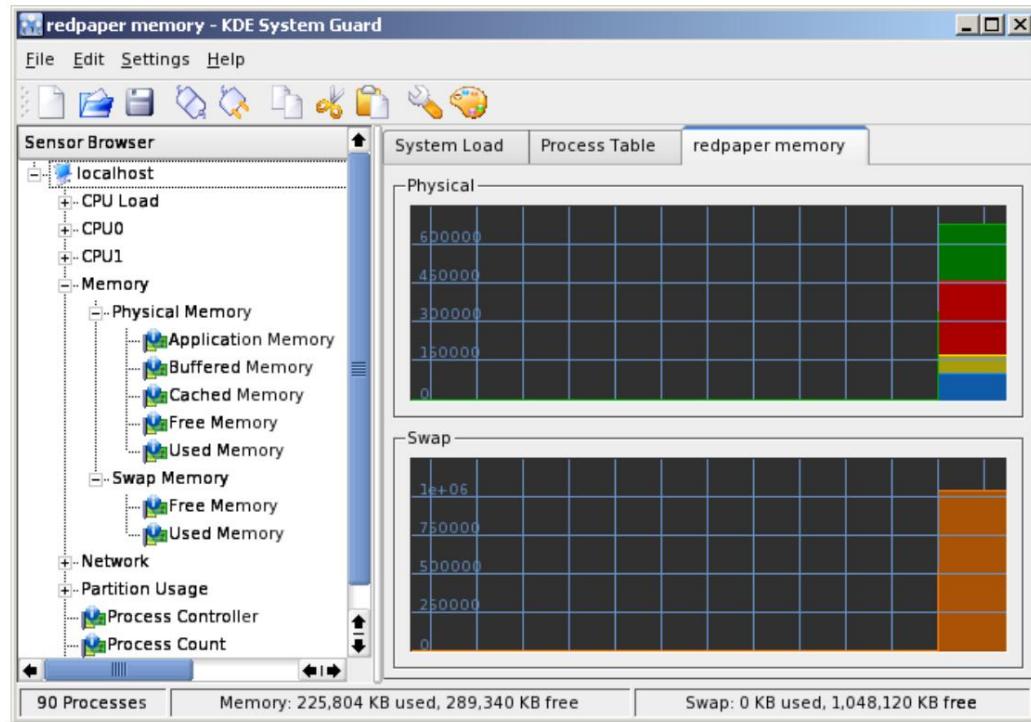


图 3-1 KDE System Guard 内存监控

表 3-1 中的指标也可以帮助您定义内存问题。

表3-1 内存分析指标

内存指示器	分析
可用内存	这指示可用的物理内存量。如果启动应用程序后,此值显著下降,则可能存在内存泄漏。请检查导致泄漏的应用程序并进行必要的调整。使用free -l -t -o 命令了解更多信息。
页面错误	页面错误有两种类型:软页面错误 (页面在内存中找到)和硬页面错误 (页面在内存中找不到,必须从磁盘获取)。访问磁盘会显著降低应用程序的速度.sar -B命令可以提供用于分析页面错误的有用信息,特别是 pgpgin/s 和 pgpgout/s 列。
文件系统缓存	这是文件系统缓存使用的公共内存空间。使用free -l -t -o 命令以获取更多信息。
进程的私有内存	这表示服务器上运行的每个进程使用的内存。您可以使用pmap 命令查看分配给特定进程的内存量。

分页和交换指示器

在 Linux 中,与所有基于 UNIX 的操作系统一样,分页和交换之间存在差异。分页将单个页面移动到磁盘上的交换空间;交换操作规模更大,它会一次性将进程的整个地址空间移动到交换空间。

交换可能有两个原因:

进程进入睡眠模式。这通常是因为进程依赖于交互操作,而编辑器、shell 和数据输入应用程序大部分时间都在等待用户输入。在此期间,它们处于非活动状态。

进程行为异常。当可用内存页面数量低于指定的最小值时，分页机制可能会造成严重的性能问题，因为分页机制无法处理物理内存页面请求，因此会调用交换机制来释放更多页面。这会显著增加磁盘 I/O 操作，并迅速降低服务器性能。

如果您的服务器总是将数据分页到磁盘（页面调出率很高），请考虑添加更多内存。
但是，对于页面输出率较低的系统，它可能不会影响性能。

3.3.2 性能调优选项

如果您认为存在内存瓶颈，请考虑执行以下一项或多项操作：

- 使用 bigpages、hugetlb、共享内存调整交换空间。
- 增加或减少页面大小。
- 改进活动和非活动内存的处理。
- 调整页出率。
- 限制服务器上每个用户使用的资源。
- 停止不需要的服务，如第 97 页上的“守护进程”中所述。
- 添加内存。

3.4 磁盘瓶颈

磁盘子系统通常是服务器性能最重要的方面，也是最常见的瓶颈。然而，问题也可能被其他因素所掩盖，例如内存不足。当应用程序仅仅因为等待 I/O 任务完成而浪费 CPU 周期时，就会被视为 I/O 受限。

最常见的磁盘瓶颈是磁盘数量过少。大多数磁盘配置基于容量需求，而非性能。最经济的解决方案是购买尽可能少的容量最大的磁盘。然而，这会将更多的用户数据放在每个磁盘上，导致物理磁盘的 I/O 速率更高，从而导致磁盘瓶颈

发生。

第二个最常见的问题是同一阵列上有太多逻辑磁盘。这会增加寻道时间并显著降低性能。

磁盘子系统在第 112 页上的 4.6 “调整磁盘子系统”中讨论。

3.4.1 查找磁盘瓶颈

出现以下症状的服务器可能存在磁盘瓶颈（或隐藏的内存问题）：

磁盘速度慢将导致：

- 内存缓冲区已填满写入数据（或等待读取数据），这将延迟所有请求，因为空闲的内存缓冲区不可用于写入请求（或响应正在等待磁盘队列中的读取数据）。
- 内存不足，例如没有足够的内存缓冲区来处理网络请求，
将导致同步磁盘 I/O。

磁盘利用率、控制器利用率或两者通常会非常高。

大多数 LAN 传输仅在磁盘 I/O 完成后才会发生，从而导致响应时间非常长且网络利用率低。

磁盘 I/O 可能需要相对较长的时间，并且磁盘队列将变满，因此 CPU 将处于空闲状态或利用率较低，因为它们需要等待很长时间才能处理下一个请求。

磁盘子系统可能是最难正确配置的子系统。

除了查看原始磁盘接口速度和磁盘容量外，了解工作负载也很重要。磁盘访问是随机的还是顺序的？是大 I/O 还是小 I/O？回答这些问题可以提供必要的信息，以确保磁盘子系统得到充分调优。

磁盘制造商倾向于展示其驱动器技术吞吐量的上限。

然而，花时间了解工作负载的吞吐量将有助于您了解对底层磁盘子系统的真正期望。

表 3-2 练习显示不同驱动器速度下 8 KB I/O 的真实吞吐量

磁盘速度	延迟	寻道时间	总随机接入时间 ^a	每个磁盘每秒的 I/O	8 KB I/O 的吞吐量
15,000 转/分 2.0 毫秒		3.8 毫秒	6.8 毫秒	147	1.15 MBps
10,000 转/分 3.0 毫秒		4.9 毫秒	8.9 毫秒	112	900 KBps
7200 转/分	4.2 毫秒	9 毫秒	13.2 毫秒	75	600 KBps

a. 假设命令处理 + 数据传输 < 1 毫秒，总随机

访问时间 = 延迟 + 寻道时间 + 1 毫秒

b. 计算为 1/总随机接入时间

随机读/写工作负载通常需要多个磁盘才能扩展。SCSI 或光纤通道的总线带宽则不太受关注。具有随机访问工作负载的大型数据库将受益于更多磁盘。大型 SMP 服务器拥有更多磁盘将获得更好的扩展性。考虑到平均商业工作负载的 I/O 占 70% 的读取和 30% 的写入，RAID-10 的性能将比 RAID-5 提高 50% 到 60%。

顺序工作负载往往会给磁盘子系统的总线带宽带来压力。当需要最大吞吐量时，请特别注意 SCSI 总线和光纤通道控制器的数量。假设阵列中的驱动器数量相同，RAID-10、RAID-0 和 RAID-5 的流式读写吞吐量都差不多。

有两种方法可以进行磁盘瓶颈分析：实时监控和跟踪。

问题发生时必须进行实时监控。当系统工作负载动态变化且问题不可重复时，这可能不太实际。

然而，如果问题是可重复的，则此方法很灵活，因为随着问题变得清晰，可以添加对象和计数器。

跟踪是指收集一段时间内的性能数据以诊断问题。这是一种执行远程性能分析的好方法。但它也有一些缺点，例如，当性能问题不可重复时，可能需要分析大型文件；跟踪中可能没有包含所有关键对象和参数，因此需要等到下次问题发生时才能获取更多数据。

vmstat 命令

在 Linux 系统上跟踪磁盘使用情况的一种方法是使用 vmstat 工具。vmstat 中与 I/O 相关的两个重要列是 bi 和 bo 字段。这两个字段用于监控数据块在磁盘子系统中的进出情况。设定基准是识别随时间变化的关键。

示例 3-2 vmstat 输出

```
[root@x232 root]# vmstat 2
rb swpd 空闲 buff 缓存 si so 0 9004 47196 1141672 0 9672 47224          双      博内 0 0      CS US SY ID WA
2 1 0      1140924 0 9276 47224 1141308 0 9160      0      950 149 74 87 13 0 0
2 0 2      47224 1141424 0 9272 47224 1141280      0      0 12 42392 189 65 88 10 0 1
0 2 0      0 9180 47228 1141360 0 9200 47228      0      0 448 0 144 28 0 0 0 100
2 0 2      1141340 0 9756 47228 1140784 0 9448      0      0 448 1764 149 66 0 1 0 99
1 0 1      47228 1141092 0 9740 47228 1140832      0      0 448 60 155 46 0 1 0 99
0 0 2      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0      0      0 6208 10730 425 413 0 3 0 97
0 2        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0      0      0 11200 6 631 737 0 6 0 94
0          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0      0      0 12224 3632 684 763 0 11 0 89
0          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0      0      0 5824 25328 403 373 0 3 0 97
0          0 640 0 159                                31 0 0 0 100
```

iostat 命令

当打开、读写并反复关闭过多文件时,可能会出现性能问题。当寻道时间(移动到存储数据的准确磁道所需的时间)开始增加时,这个问题会变得尤为明显。使用iostat工具,您可以实时监控I/O设备的负载。不同的选项可以让您更深入地收集必要的数据。

示例 3-3 显示了设备 /dev/sdb1 上的潜在 I/O 瓶颈。此输出显示平均等待时间(await)约为 2.7 秒,服务时间(svctm)为 270 毫秒。

示例 3-3 使用 iostat 2 -x /dev/sdb1 显示的 I/O 瓶颈示例

```
[root@x232 root]# iostat 2 -x /dev/sdb1

平均 CPU:%用户%nice%sys%空闲
11.50 0.00 2.00 86.50

设备:rrqm/s wrqm/sr/sw/s rsec/s wsec/s avgqu-sz 等待 svctm %util          rkB/s      wkB/s 平均-sz
/dev/sdb1 441.00 3030.00 7.00 30.50 3584.00 24480.00 1792.00 12240.00 748.37 101.70 2717.33 266.67 100.00

平均 CPU:%用户%nice%sys%空闲
10.50 0.00 1.00 88.50

设备:rrqm/s wrqm/sr/sw/s rsec/s wsec/s avgqu-sz 等待 svctm %util          rkB/s      wkB/s 平均-sz
/dev/sdb1 441.00 3030.00 7.00 30.00 3584.00 24480.00 1792.00 12240.00 758.49 101.65 2739.19 270.27 100.00

平均 CPU:%用户%nice%sys%空闲
10.95 0.00 1.00 88.06

设备:rrqm/s wrqm/sr/sw/s rsec/s wsec/s avgqu-sz 等待 svctm %util          rkB/s      wkB/s 平均-sz
/dev/sdb1 438.81 3165.67 6.97 30.35 3566.17 25576.12 1783.08 12788.06 781.01 101.69 2728.00 268.00 100.00
```

有关这些字段的更详细说明,请参阅 iostat(1) 的手册页。

按照第 115 页 4.6.2 节 “I/O 电梯调优与选择”中所述,对电梯算法所做的更改将体现在 avgqrq-sz (平均请求大小)和 avgqu-sz (平均队列长度)中。通过调整电梯设置降低延迟后,avgqrq-sz 也会随之减少。您还可以监控 rrqm/s 和 wrqm/s,以了解它们对磁盘可管理的合并读写次数的影响。

3.4.2 性能调优选项

在确认磁盘子系统是系统瓶颈之后,可以采取多种解决方案。
这些解决方案包括:

如果工作负载是顺序性的,并且对控制器带宽造成压力,解决方案是添加更快的磁盘控制器。但是,如果工作负载的随机性更强,那么瓶颈很可能在于磁盘驱动器,而添加更多驱动器可以提高性能。

在 RAID 环境中添加更多磁盘驱动器。这会将数据分散到多个物理磁盘上,从而提高读写性能。这将增加每秒的 I/O 数量。此外,请使用硬件 RAID,而不是 Linux 提供的软件实现。如果使用硬件 RAID,则 RAID 级别对操作系统是隐藏的。

考虑使用具有条带化的 Linux 逻辑卷,而不是没有条带化的大型单个磁盘或逻辑卷。

将处理卸载到网络中的另一个系统 (用户、应用程序或服务)。

添加更多内存。添加内存会增加系统内存磁盘缓存,从而有效缩短磁盘响应时间。

3.5 网络瓶颈

网络子系统的性能问题可能导致许多问题,例如内核崩溃。为了分析这些异常以检测网络瓶颈,每个 Linux 发行版都包含流量分析器。

3.5.1 查找网络瓶颈

我们推荐 KDE System Guard,因为它拥有图形界面且易于使用。该工具包含在发行版 CD 中,第 62 页 “2.3.17 KDE System Guard”章节对此进行了详细介绍。第 88 页的图 3-2 展示了它的运行情况。

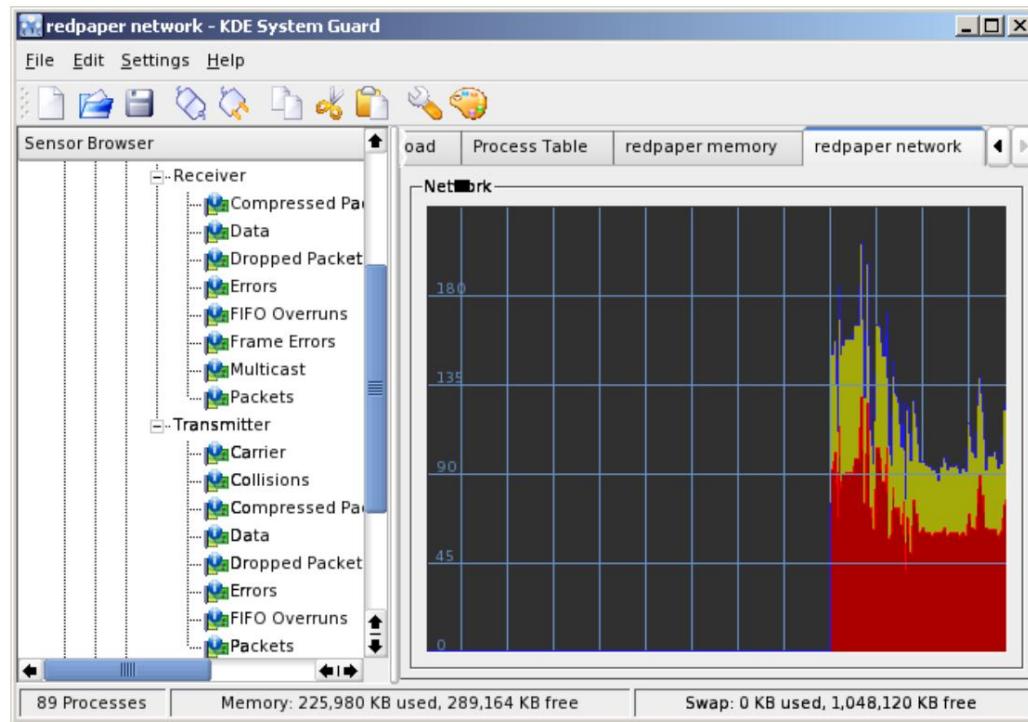


图3-2 KDE System Guard网络监控

需要注意的是,导致这些性能问题的原因可能有很多,而且有时问题会同时发生,这使得查明根源变得更加困难。表 3-3 中的指标可以帮助您确定网络问题。

表3-3 网络分析指标

网络指示灯	分析
收到的数据包 已发送数据包	显示进出指定网络接口的数据包数量。检查内部和外部接口。
碰撞数据包	当同一域中存在多个系统时,会发生冲突。使用集线器可能会导致许多冲突。
丢弃的数据包	数据包可能因多种原因被丢弃,但最终会影响性能。例如,如果服务器网络接口配置为以 100 Mbps 全双工运行,但网络交换机配置为以 10 Mbps 运行,则路由器可能具有丢弃这些数据包的 ACL 过滤器。例如: <code>iptables -t 过滤器 -A FORWARD -p all -i eth2 -o eth1 -s 172.18.0.0/24 -j DROP</code>
错误	如果通信线路(例如电话线)质量不佳,就会发生错误。在这种情况下,必须重新发送损坏的数据包,从而降低网络吞吐量。
适配器故障	网络速度变慢通常是由于网络适配器故障造成的。当这类硬件发生故障时,它可能会开始在网络上广播垃圾数据包。

3.5.2 性能调优选项

以下步骤说明了解决与网络瓶颈相关的问题应该采取哪些措施：

确保网卡配置与路由器和交换机配置相匹配（例如，帧大小）。

修改子网的组织方式。

使用更快的网卡。

调整适当的 IPv4 TCP 内核参数。（请参阅第 4 章“调整操作系统”（第 91 页）。）一些与安全相关的参数也可以提高性能，如该章所述。

如果可能，请更换网卡并重新检查性能。

如果可能的话，添加网卡并将它们绑定在一起以形成适配器组。



第 4 章调整操作系统

Linux 发行版和 Linux 内核提供了各种参数和设置,方便 Linux 管理员调整系统以最大限度地提高性能。正如本文前面所述,没有神奇的调优旋钮可以提升任何应用程序的系统性能。下一章讨论的设置将提升特定硬件配置和应用程序布局的性能。提升 Web 服务器性能的设置可能会对数据库系统的性能产生负面影响。

本章介绍调整基于内核 2.6 的 Linux 发行版的步骤。

由于当前基于 2.6 内核的发行版版本号从 2.6.9 到 2.6.19 (截至本文撰写时)不等,因此某些调选项可能仅适用于特定内核版本。本文旨在描述能够最大程度提升性能的参数,并帮助您了解 Linux 中使用的一些基本技术,包括:

Linux 内存管理

系统清理

磁盘子系统调整

使用 sysctl 进行内核调整

网络优化

本章包含以下部分:

- 4.1, “调整原则” 第 92 页
- 4.2, “安装注意事项” (第 92 页)
- 4.3, “更改内核参数” 第 104 页
- 4.4, “调整处理器子系统” (第 107 页)
- 4.5, “调整 vm 子系统” (第 109 页)
- 4.6, “调整磁盘子系统” (第 112 页)
- 4.7, “调整网络子系统” (第 124 页)

4.1 调优原则

任何系统调优都应遵循一些简单的原则,其中最重要的是如下所述的变更管理。通常,系统调优的第一步应该是分析和评估当前的系统配置。确保系统性能符合硬件制造商的规定,并且所有设备都以最佳模式运行,这将为后续的调优奠定坚实的基础。此外,在执行任何特定的调优任务之前,为实现最佳性能而设计的系统应该尽量减少运行不必要的任务和子系统。最后,在进行特定的系统调优时,应该注意,调优通常会针对特定的工作负载定制系统。因此,系统在预期的负载特性下会表现更好,但在不同的工作负载模式下可能会表现更差。例如,对系统进行低延迟调优,这在大多数情况下会对吞吐量产生不利影响。

4.1.1 变更管理

虽然变更管理与性能调优并非严格相关,但它可能是成功进行性能调优的最重要因素。以下考虑因素可能对您来说习以为常,但为了提醒您,我们强调以下几点:

- 在调整任何 Linux 系统之前,实施适当的变更管理流程。
- 切勿在生产系统上开始调整设置。
- 在调整过程中切勿更改多个变量。
- 重新测试据称可以提高性能的参数;有时统计数据会发挥作用。

记录成功的参数,并与社区分享,无论您认为它们多么微不足道。在生产环境中获得的任何结果都能极大地提升 Linux 性能。

4.2 安装注意事项

理想情况下,服务器系统针对特定性能目标的调优应该从设计和安装阶段开始。根据工作负载模式定制系统的正确安装,将在后期的调优阶段节省大量时间。

4.2.1 安装

理想情况下,任何系统的调优都应该从非常早期的阶段开始。理想情况下,系统应该根据应用程序的需求和预期的工作负载进行量身定制。我们理解,大多数情况下,管理员需要调整已安装的系统以应对瓶颈问题,但我们也想强调在操作系统初始安装期间可用的调优选项。

在开始安装 Linux 之前应该解决几个问题,包括:

- 处理器技术的选择
- 磁盘技术的选择
- 应用

然而,这些问题超出了本文的范围。

理想情况下,在开始安装之前应该回答以下问题:

我需要什么风格和版本的 Linux?

收集完业务和应用程序需求后,决定使用哪个版本的 Linux。企业通常会签订合同,允许使用特定的 Linux 发行版。在这种情况下,财务和合同利益可能会决定可以使用的 Linux 版本。但是,如果您可以完全自由地选择 Linux 发行版,则需要考虑以下问题:

- 支持企业版 Linux 还是定制发行版?

在某些科学环境中,运行不受支持的 Linux 版本(例如 Fedora)是可以接受的。对于企业工作负载,我们强烈建议使用完全受支持的发行版,例如 Red Hat Enterprise Linux 或 Novell SUSE Enterprise Linux。

- 企业发行版是什么版本?

大多数企业级 Linux 发行版都有不同的版本,其内核版本、支持的软件包或功能,以及最重要的硬件支持级别各不相同。在进行任何安装之前,请仔细检查支持的硬件配置,以免丢失任何硬件功能。

选择正确的内核

企业级 Linux 发行版提供了多个内核包,如表 4-1 所示。出于性能方面的考虑,请务必为您的系统选择最合适的核心。

但是在大多数情况下,安装程序会选择正确的内核。请记住,确切的内核包名称因发行版而异。

表 4-1 可用的内核类型

内核类型	描述
标准	单处理器机器。
对称多处理器	内核支持 SMP 和超线程机器。一些软件包还包含对 NUMA 的支持。具体支持情况可能会有所不同,具体取决于内存大小、CPU 数量等。
Xen	包括在 Xen 虚拟机中运行的 Linux 内核版本。

注意:大多数最新内核都具有称为 SMP 替代方案的功能,该功能可在启动时进行自我优化。有关详细信息,请参阅发行版的发行说明。

选择什么分区布局?

磁盘子系统的分区布局通常由应用程序需求、系统管理考虑和个人偏好决定,而非性能。大多数情况下都会提供分区布局。我们唯一的建议是,如果可能,请使用交换分区。与交换文件相比,交换分区具有性能优势,因为它没有文件系统的开销。交换分区很简单,可以根据需要通过额外的交换分区甚至交换文件进行扩展。

使用什么文件系统?

不同的文件系统在数据完整性和性能方面具有不同的特点。

某些文件系统可能不受相应的 Linux 发行版或要使用的应用程序支持。对于大多数服务器安装,安装程序建议的默认文件系统即可提供足够的性能。如果您对最小延迟或最大吞吐量有特定要求,我们建议您根据这些要求选择相应的文件系统。有关详细的选择标准,请参阅第 112 页上的 4.6 “调整磁盘子系统”。

套餐选择:最少还是全部?

在 Linux 安装过程中,管理员面临着“最小化安装”或“全部安装”的选择。有些人更喜欢“全部安装”,因为这样很少需要安装软件包来解决依赖关系。

请考虑以下几点:虽然与性能无关,但需要指出的是,“全包”或“近乎全包”的安装会给系统带来安全威胁。生产系统上可用的开发工具可能会导致严重的安全威胁。安装的软件包越少,浪费的磁盘空间就越少,而且拥有更多可用空间的磁盘的性能会优于拥有少量可用空间的磁盘。如果需要,诸如 Red Hat Packet Manager、rpm 或 yum 之类的智能软件安装程序会自动解析依赖项。因此,我们建议您考虑选择最少的软件包,仅包含成功实施应用程序所需的软件包。

Netfilter 配置

您需要确定是否需要配置 Netfilter 防火墙。Netfilter 防火墙通常用于保护系统免受恶意攻击。但是,过多复杂的防火墙规则可能会降低高数据流量环境下的性能。我们将在第 132 页的 4.7.6 节“Netfilter 对性能的影响”中介绍 Netfilter 防火墙。

SELinux

在某些 Linux 发行版(例如 Red Hat Enterprise Linux 4.0)中,安装程序会允许您选择安装 SELinux。SELinux 会显著降低性能,因此您应该仔细评估特定系统是否真的需要 SELinux 提供的额外安全性。有关更多信息,请参阅第 102 页上的 4.2.4 “SELinux”。

运行级别选择

安装过程中的最后一个选项是选择系统默认的运行级别。除非您特别需要在运行级别 5(图形用户模式)下运行系统,否则我们强烈建议所有服务器系统都使用运行级别 3。

通常情况下,数据中心的系统不需要 GUI,而且运行级别 5 带来的开销相当大。如果安装程序未提供运行级别选择,我们建议您在初始系统配置后手动选择运行级别 3。

4.2.2 查看当前配置

正如引言中所述,为任何系统调优尝试打下坚实的基础至关重要。坚实的基础意味着确保所有子系统都按设计运行,并且没有异常。例如,千兆网卡和存在网络性能瓶颈的服务器就属于此类异常。如果网卡自动协商为 100 MBit/半双工,那么调整 Linux 内核的 TCP/IP 实现可能就没什么用。

消息

dmesg 的主要目的是显示内核消息。如果出现硬件问题或将模块加载到内核时出现问题, dmesg 可以提供有用的信息。

此外,使用 dmesg,您可以确定服务器上安装了哪些硬件。Linux 每次启动时都会检查硬件并记录相关信息。您可以使用命令 /bin/dmesg 查看这些日志。

示例 4-1 dmesg 的部分输出

Linux 版本 2.6.18-8.el5 (brewbuilder@ls20-bc1-14.build.redhat.com) (gcc 版本 4.1.1 20070105 (Red Hat 4.1.

1-52)) #1 SMP 星期五 1 月 26 日 14:15:14 EST 2007 命令行:ro
root=/dev/VolGroup00/LogVol00 rhgb quiet

未找到 NUMA 配置
伪造一个位于 0000000000000000-0000000140000000 的节点
Bootmem 设置节点 0 0000000000000000-0000000140000000
在节点 0 上,总页数:1029288
DMA 区域:2726 页,LIFO 批次:0
DMA32 区域:768002 页,LIFO 批次:31
正常区域:258560 页,后进先出批次:31

内核命令行:ro root=/dev/VolGroup00/LogVol00 rhgb quiet
初始化 CPU#0

内存:4042196k/5242880k 可用 (2397k 内核代码,151492k 保留,1222k 数据,196k 初始化)

使用计时器特定程序校准延迟.. 7203.13 BogoMIPS (lpj=3601568)
安全框架 v1.0.0 初始化 SELinux:正在初始化。

SELinux:以宽容模式启动

CPU:跟踪缓存:12K uops,L1 D 缓存:16K CPU:L2 缓存:1024K 在空闲线程中
使用 mwait。

CPU:物理处理器 ID:0 CPU:处理器核心 ID:0
CPU0:已启用热监控 (TM2)

SMP 替代方案:切换到 UP 代码 ACPI:核心修订版 20060707
使用本地 APIC 定时器中断,结果 12500514
检测到 12.500 MHz APIC 定时器。

SMP 替代方案:切换到 SMP 代码

sizeof(vma)=176 字节
sizeof(page)=56 字节
sizeof(inode)=560 字节
sizeof(dentry)=216 字节
sizeof(ext3inode)=760 字节
sizeof(buffer_head)=96 字节
sizeof(skbuff)=240 字节

io 调度器 noop 已注册 io 调度器 anticipatory
已注册 io 调度器 deadline 已注册 io 调度器 cfq 已注册
(默认)

SCSI 设备 sda:143372288 个 512 字节 hdwr 扇区 (73407 MB)sda:假设已启用写入 sda:假设
驱动器缓存:直写

eth0:Tigon3 [partno(BCM95721) rev 4101 PHY(5750)] (PCI Express) 10/100/1000BaseT 以太网 00:11:25:3f:19:b4 eth0:RXcsums[1] LinkChgREG[0]
Mlirq[0] ASF[1] Split[0]
WireSpeed[1] TSOcap[1] eth0:dma_rwctrl[76180000] dma_mask[64 位]

dm-0 上的 EXT3 FS,内部日志

kjournald 正在启动。提交间隔 5 秒
sda1 上的 EXT3 FS,内部日志
EXT3-fs:以有序数据模式挂载的文件系统。

ulimit

此命令内置于 bash shell 中,用于控制 shell 可用的资源以及在允许此类控制的系统上由其启动的进程。

使用 -a 选项列出我们可以设置的所有参数:

```
ulimit -a
```

示例 4-2 ulimit 的输出

```
[root@x232 html]# ulimit -a
核心文件大小 (块, -c)0
数据段大小 (千字节, -d)无限制
文件大小 (块, -f)无限制
最大锁定内存 (kbytes, -l)4
最大内存大小 (kbytes, -m)无限制
打开文件 (-n)1024
管道大小 (512 字节, -p)8
堆栈大小 (kbytes, -s)10240
CPU 时间 (秒, -t)无限制
最大用户进程虚拟内存 (kbytes, -v)无限制
(-u)7168
```

-H 和 -S 选项指定可以为给定资源设置的硬限制和软限制。如果超出软限制,系统管理员将收到警告。硬限制是用户收到“文件句柄不足”错误消息之前可以达到的最大值。

例如,您可以为文件句柄和打开文件的数量设置硬限制 (-n) :

```
ulimit -Hn 4096
```

对于文件句柄数和打开文件数的软限制,使用:

```
ulimit -Sn 1024
```

要查看硬值和软值,请使用新值发出命令:

```
ulimit -Hn
ulimit -Sn
```

例如,此命令可用于动态限制 Oracle® 用户。要在启动时设置,请在 /etc/security/limits.conf 中输入以下行:

```
软文件 4096
硬盘 nofile 10240
```

此外,请确保默认 pam 配置文件 (Red Hat Enterprise Linux 为 /etc/pam.d/system-auth,SUSE Linux Enterprise Server 为 /etc/pam.d/common-session)具有以下条目:

```
会话需要 pam_limits.so
```

需要此条目,以便系统可以强制执行这些限制。

要获取ulimit命令的完整语法,请发出:

```
ulimit -?
```

4.2.3 尽量减少资源使用

为追求最高性能而设计的系统必须最大限度地减少资源浪费。赛车不会提供与普通乘用车相同的便利设施,但为了尽可能快地行驶而配备杯架和舒适的座椅,其实是一种资源浪费。服务器系统也是如此。运行占用大量内存的 GUI 和大量不必要的守护进程会降低整体性能。本节介绍如何优化系统资源消耗。

守护进程

Linux 发行版默认安装后,可能会启用一些可能不必要的服务和守护进程。禁用不需要的守护进程可以减少系统的整体内存占用,减少正在运行的进程和上下文切换次数,更重要的是,可以降低遭受各种安全威胁的风险。禁用不需要的守护进程还可以缩短服务器的启动时间。

默认情况下,在大多数系统上,可以安全地停止和禁用已启动的多个守护进程。表 4-2 列出了各种 Linux 安装中启动的守护进程。如果适用,您应该考虑在您的环境中禁用这些守护进程。请注意,该表列出了几个商用 Linux 发行版的守护进程。实际运行的守护进程数量可能因您的具体 Linux 安装而异。有关这些守护进程的更详细说明,请参阅第 99 页图 4-1 中显示的“系统配置服务”或第 100 页图 4-2 中显示的 YaST GUI。

表 4-2 默认安装中启动的可调守护进程

守护进程描述	
apmd	高级电源管理守护进程。apmd 通常不会在服务器上使用。
arpTables_jf	arpTables 网络过滤器的用户空间程序。除非您打算使用 arpTables,否则可以安全地禁用此守护进程。
自动文件系统	按需自动挂载文件系统(例如,自动挂载 CD-ROM)。在服务器系统上,很少需要自动挂载文件系统。
cpuspeed 守护进程	用于动态调整 CPU 的频率。在服务器环境中,这建议关闭守护进程。
杯子	通用 UNIX 打印系统。如果您计划在服务器上提供打印服务,请不要禁用此服务。
加仑/分钟	文本控制台的鼠标服务器。如果您希望本地文本控制台支持鼠标,请不要禁用。
霍普金斯	HP OfficeJet 支持。如果您计划使用 HP OfficeJet 打印机,请勿禁用服务器。
中断平衡	在多个处理器之间平衡中断。如果您计划静态平衡 IRQ,则可以在单 CPU 系统上安全地禁用此守护进程。
ISDN 业务数字网	ISDN 调制解调器支持。如果您计划使用 ISDN 调制解调器,请勿禁用服务器。
葛	检测并配置新硬件。如果硬件发生变更,则应手动运行。

守护进程描述	
净流表	用于支持导出 NFS 共享。如果您计划为服务器提供 NFS 共享,请不要禁用该选项。
nfslock	用于 NFS 文件锁定。如果您计划提供 NFS 共享,请勿禁用你的服务器。
PCMCIA	服务器上的 PCMCIA 支持。服务器系统很少依赖 PCMCIA 适配器,因此在大多数情况下禁用此守护进程是安全的。
端口映射	为 RPC 服务 (例如 NIS 和 NFS) 动态分配端口。如果系统不提供基于 RPC 的服务,则无需此守护进程。
rawdevices	提供对原始设备绑定的支持。如果您不打算使用原始设备,可以放心地将其关闭。
rpc*	主要用于 NFS 和 Samba 的各种远程过程调用守护进程。如果系统不提供基于 RPC 的服务,则不需要此守护进程。
发送邮件	邮件传输代理。如果您计划为相应系统提供邮件服务,请不要禁用此守护进程。
智能	自我监控和报告技术守护进程,用于监视 SMART 兼容设备是否存在错误。除非您使用基于 IDE/SATA 技术的磁盘子系统,否则无需使用 SMART 监控。
xfs	X Windows 的字体服务器。如果您要在运行级别 5 下运行,请不要禁用此守护进程。

注意:关闭 xfs 守护进程会导致 X 无法在服务器上启动。仅当服务器不打算启动到 GUI 界面时才需要关闭此功能。只需在执行 startx 命令之前启动 xfs 守护进程,即可使 X 正常启动。

在 Novell SUSE 和 Red Hat Enterprise Linux 系统上, /sbin/chkconfig 命令为管理员提供了一个易于使用的界面,用于更改各种守护进程的启动选项。使用 chkconfig 时首先要运行的命令之一是检查所有正在运行的守护进程:

```
/sbin/chkconfig --list | grep on
```

如果您不希望守护进程在下次计算机启动时启动,请以 root 身份执行以下命令之一。它们的结果相同,区别在于第二个命令会在所有运行级别上禁用守护进程,而 --level 标志可用于指定精确的运行级别:

```
/sbin/chkconfig --levels 2345 发送邮件关闭 /sbin/
chkconfig 发送邮件关闭
```

提示:无需浪费宝贵的时间等待重启完成,只需将运行级别分别更改为 1 并改回 3 或 5 即可。

还有另一个有用的系统命令 /sbin/service, 它使管理员能够立即更改任何已注册服务的状态。首先, 管理员应该始终选择通过发出以下命令来检查服务 (在我们的示例中为 sendmail) 的当前状态:

```
/sbin/service sendmail status
```

要立即停止示例中的 sendmail 守护程序, 请使用以下命令:

/sbin/service sendmail 停止

service 命令非常有用,因为它可以让你立即验证是否需要守护进程。除非你更改系统运行级别或重启系统,否则通过 chkconfig 执行的更改将不会生效。但是,通过 service 命令禁用的守护进程在重启后会重新启用。如果你的 Linux 发行版不支持service命令,你可以通过 init.d 目录启动或停止守护进程。

例如,检查 CUPS 守护进程的状态可以按如下方式执行:

/etc/init.d/cups状态

同样,有一些基于 GUI 的程序可用于修改要启动的守护进程,如图 4-1 所示。要运行 Red Hat Enterprise Linux 的服务配置 GUI,请点击“主菜单” → “系统设置” → “服务器设置” → “服务”,或输入以下命令:

/usr/bin/redhat-config-services

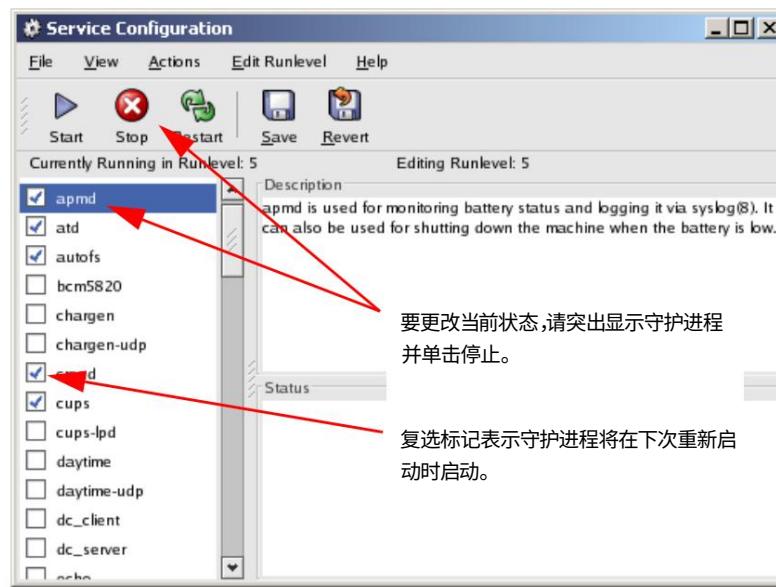


图4-1 Red Hat服务配置界面

Novell SUSE 系统通过 YaST 实用程序提供相同的功能。在 YaST 中,服务配置位于“系统” → “系统服务 (运行级别)”。进入服务配置后,我们建议您使用专家模式,以便准确设置相应守护进程的状态。在运行级别 3 中运行 YaST 的界面如图 4-2 (第 100 页)所示。

Press F1 for Help

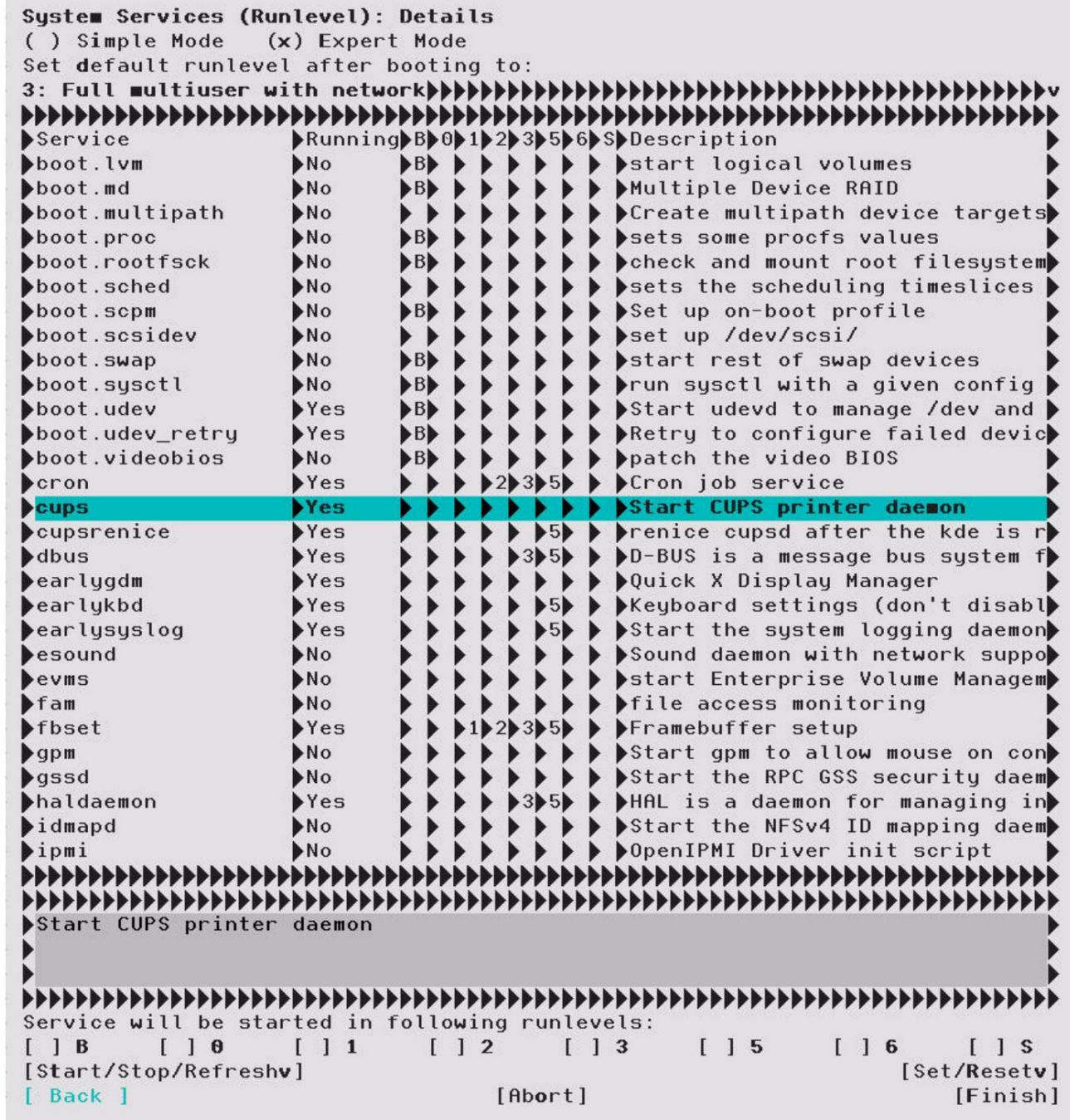


图 4-2 YaST 中的系统服务面板

在图 4-2 所示的 YaST 面板中,您可以根据运行级别启用或禁用各种服务。但是,这需要使用图 4-2 顶部所示的专家模式。

更改运行级别

尽可能不要在 Linux 服务器上运行图形用户界面 (GUI)。通常情况下,Linux 服务器不需要 GUI,大多数 Linux 管理员都会很乐意地向你保证。

所有管理任务都可以通过命令行、重定向 X 显示器或 Web 浏览器界面高效完成。如果您更喜欢图形界面,可以使用一些实用的 Web 工具,例如 webmin、Linuxconf 和 SWAT。

提示:即使在服务器本地禁用了 GUI,您仍然可以远程连接并使用 GUI。为此,请在 ssh 命令中使用 -X 参数。

如果必须使用 GUI,请根据需要启动和停止它,而不是一直运行它。大多数情况下,服务器应该运行在运行级别 3,这样机器启动时就不会启动 X 服务器。如果要重新启动 X 服务器,请在命令提示符下使用 startx 命令。

1. 使用 runlevel 命令确定机器正在运行哪个运行级别。

这将打印前一个运行级别和当前运行级别。例如,N 5 表示没有前一个运行级别 (N),当前运行级别为 5。

2. 要在运行级别之间切换,请使用 init 命令。例如,要切换到运行级别 3,请输入 init 3 命令。

Linux 中使用的运行级别是:

- | | |
|---|---|
| 0 | 停止 (不要将 initdefault 设置为该项,否则服务器将在完成启动过程后立即关闭。) |
| 1 | 单用户模式 |
| 2 | 多用户,无 NFS (如果没有网络,则与 3 相同) |
| 3 | 完全多用户模式 |
| 4 | 未使用 |
| 5 | X11 |
| 6 | 重新启动 (不要将 initdefault 设置为该项,否则服务器计算机将在启动时不断重新启动。) |

要设置机器启动时的初始运行级别,请修改 /etc/inittab 文件,如下所示

第 102 页的图 4-3 中有以下行:

id:3:初始化默认值:

```

... (未显示行)

# 默认运行级别在这里定义
id:3:初始化默认值:                                ← 要启动 Linux 而不启动 GUI,请将运行级别设置为 3。
# 如果未在紧急 (-b)模式下启动,则首先执行的脚本
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc 负责运行级别处理
#
# 运行级别 0 表示系统停止 (请勿将其用于 initdefault! )
# 运行级别 1 是单用户模式
# 运行级别 2 是没有远程网络 (例如 NFS) 的本地多用户
# 运行级别 3 是带网络的完整多用户
# 运行级别 4 未使用
# 运行级别 5 是具有网络和 xdm 的完整多用户
# 运行级别 6 是系统重启 (不要将其用于 initdefault! )
#



... (未显示行)

# getty-programs 用于正常运行级别
# <id>:<运行级别>:<操作>:<进程>
# “id” 字段必须与最后一个相同
设备的#个字符 ( “tty”之后 ) 。
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:重生:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6
#
#S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102
... (未显示行)

```

要仅提供两个本地虚拟终端,请注释掉 3、4、5 和 6 的 mingetty 条目。

图 4-3 修改后的 /etc/inittab (仅显示文件的一部分)

限制本地终端

默认情况下,本地会生成多个虚拟控制台。虚拟终端占用的内存量可以忽略不计;尽管如此,我们仍会尽力充分利用系统。

只需减少正在运行的进程数量,故障排除和进程分析就会变得简单,这就是将本地终端限制为两个的原因。

为此,请注释掉要禁用的每个 mingetty ttyx 行。例如,在图 4-3 中,我们将控制台限制为两个。这样,当某个命令终止了您正在本地工作的 shell 时,就可以使用备用本地终端。

4.2.4 SELinux

Red Hat Enterprise Linux 4 引入了一种新的安全模型,即安全增强型 Linux (SELinux),这是迈向更高安全性的重要一步。SELinux 引入了

强制策略模型克服了 Linux 采用的标准自主访问模型的局限性。SELinux 在用户和进程级别强制执行安全机制；因此，任何给定进程的安全漏洞都只会影响分配给该进程的资源，而不会影响整个系统。SELinux 的工作原理类似于虚拟机。例如，如果恶意攻击者在 Apache 中使用 root 权限漏洞，则只有分配给 Apache 守护进程的资源可能会受到影响。

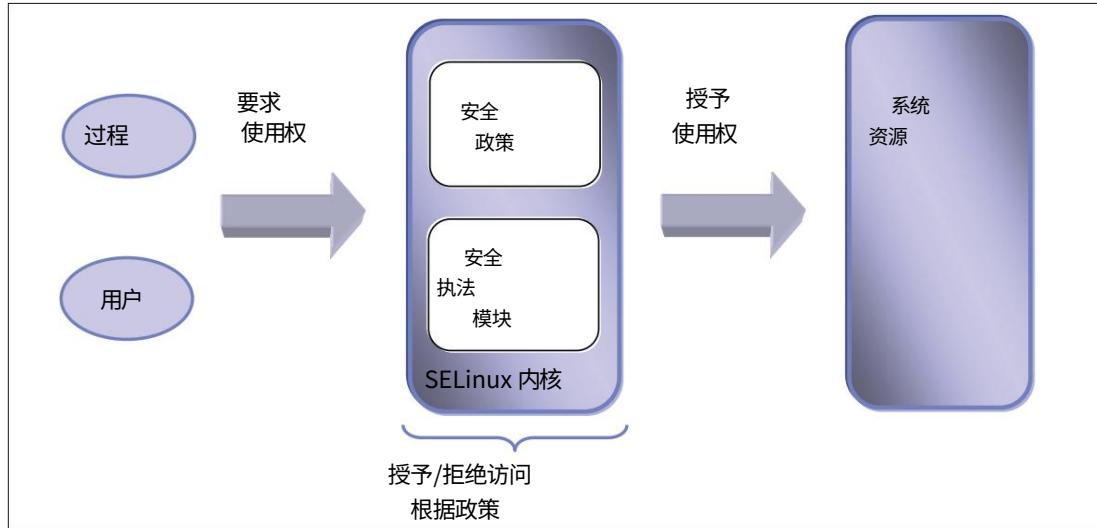


图 4-4 SELinux 示意图

然而，强制执行这种基于策略的安全模型是有代价的。用户或进程对系统资源（例如 I/O 设备）的每次访问都必须受到 SELinux 的控制。

检查权限的过程可能会造成高达 10% 的开销。SELinux 对于任何边缘服务器（例如防火墙或 Web 服务器）都至关重要，但后端数据库服务器增加的安全级别可能无法弥补性能损失。

通常，禁用 SELinux 最简单的方法是一开始就不安装它。但系统通常使用默认参数安装，没有意识到 SELinux 会影响性能。要在安装后禁用 SELinux，请在 GRUB 引导加载程序中包含正在运行的内核的行后附加条目 `selinux=0`（示例 4-3）。

示例 4-3 禁用 SELinux 的示例 grub.conf 文件

```

默认值=0
splashimage=(hd0,0)/grub/splash.xpm.gz
隐藏菜单
标题 Red Hat Enterprise Linux AS (2.6.9-5.ELsmp)
    根 (hd0,0)
    内核/vmlinuz-2.6.9-5.ELsmp ro root=LABEL=/ selinux=0
    initrd /initrd-2.6.9-5.ELsmp.img

```

禁用 SELinux 的另一种方法是通过存储在 `/etc/selinux/config` 下的 SELinux 配置文件。从该文件中禁用 SELinux 的步骤如示例 4-4 所示。

示例 4-4 通过配置文件禁用 SELinux

```

# 该文件控制系统上的 SELinux 的状态。
# SELINUX= 可以采用以下三个值之一：
#       强制执行 强制执行 SELinux 安全策略。
#       宽容 - SELinux 打印警告而不是强制执行。
#       已禁用   SELinux 已完全禁用。

```

```
SELINUX=已禁用
# SELINUXTYPE= 正在使用的策略类型。可能的值包括：
# targeted - 仅目标网络守护进程受到保护。
# strict - 完整的 SELinux 保护。
SELINUXTYPE=目标
```

如果您决定在基于 Linux 的服务器上使用 SELinux，则可以调整其设置以更好地适应您的环境。在正在运行的系统上，检查缓存的 Linux 安全模块 (LSM) 权限的工作集是否超过了默认的访问向量缓存 (AVC) 大小（512 个条目）。

检查 `/selinux/avc/hash_stats` 以了解最长链的长度。任何超过 10 的值都表明可能存在瓶颈。

提示：要检查访问向量缓存的使用情况统计信息，您也可以使用 `avcstat` 实用程序。

如果系统在访问向量缓存中遇到瓶颈（例如，在负载过重的防火墙上），请尝试将 `/selinux/avc/cache_threshold` 调整为稍高的值，然后重新检查哈希统计信息。

4.2.5 编译内核

创建和编译自己的内核对提升系统性能的影响远不如人们通常认为的那样。大多数 Linux 发行版附带的现代内核都是模块化的。它们只加载需要用到的部分。重新编译内核可以减小内核大小并降低其整体行为（例如，实时行为）。更改源代码中的某些参数也可能会提升系统性能。然而，大多数企业 Linux 发行版提供的支持订阅并不涵盖非标准内核。此外，如果使用非标准内核，企业 Linux 发行版提供的广泛 ISV 应用程序和 IBM 硬件认证也将失效。

话虽如此，定制内核确实可以提升性能，但这并不能弥补在企业环境中运行不受支持的内核所带来的挑战。虽然这对于商业工作负载来说确实如此，但如果您的领域是高性能计算等科学工作负载，那么定制内核或许更适合您。

重新编译内核时，请勿尝试使用 `-C09` 之类的特殊编译器标志。Linux 内核的源代码已手动调整以匹配 GNU C 编译器。使用特殊编译器标志轻则可能会降低内核性能，重则可能会破坏代码。

请记住，除非您真正知道自己在做什么，否则错误的内核参数实际上可能会降低系统性能。

4.3 更改内核参数

虽然不建议大多数用户修改并重新编译内核源代码，但 Linux 内核提供了另一种调整内核参数的方法。`proc` 文件系统提供了一个与正在运行的内核接口，可用于监视内核运行情况以及动态更改内核设置。

要查看当前内核配置,请在 /proc/sys 中选择一个内核参数目录,并对相应文件使用cat命令。在示例 4-5 中,我们解析了系统当前的内存过量使用策略。输出 0 表示系统在授予应用程序内存分配请求之前始终会检查可用内存。要更改此默认行为,我们可以使用echo命令并为其提供新值,在本例中为 1 (1 表示内核将授予所有内存分配,而不检查分配是否可以满足)。

示例 4-5 通过 proc 文件系统更改内核参数

```
[root@linux vm]# cat overcommit_memory
0
[root@linux vm]# echo 1 > overcommit_memory
```

虽然使用cat和echo更改内核参数的演示方法速度很快,并且可以在任何具有 proc 文件系统的系统上使用,但它有两个明显的缺点。

- echo 命令不会对参数执行任何一致性检查。
- 系统重新启动后,对内核的所有更改都将丢失。

为了解决这个问题,一个名为sysctl的实用程序可以帮助管理员更改内核参数。

提示:默认情况下,内核包含必要的模块,允许您使用sysctl进行更改,而无需重启。但是,如果您选择删除此支持(在操作系统安装期间),则必须重启 Linux 才能使更改生效。

此外,Red Hat Enterprise Linux 和 Novell SUSE Enterprise Linux 提供了图形化的方式来修改这些 sysctl 参数。图 4-5 展示了其中一个用户界面。

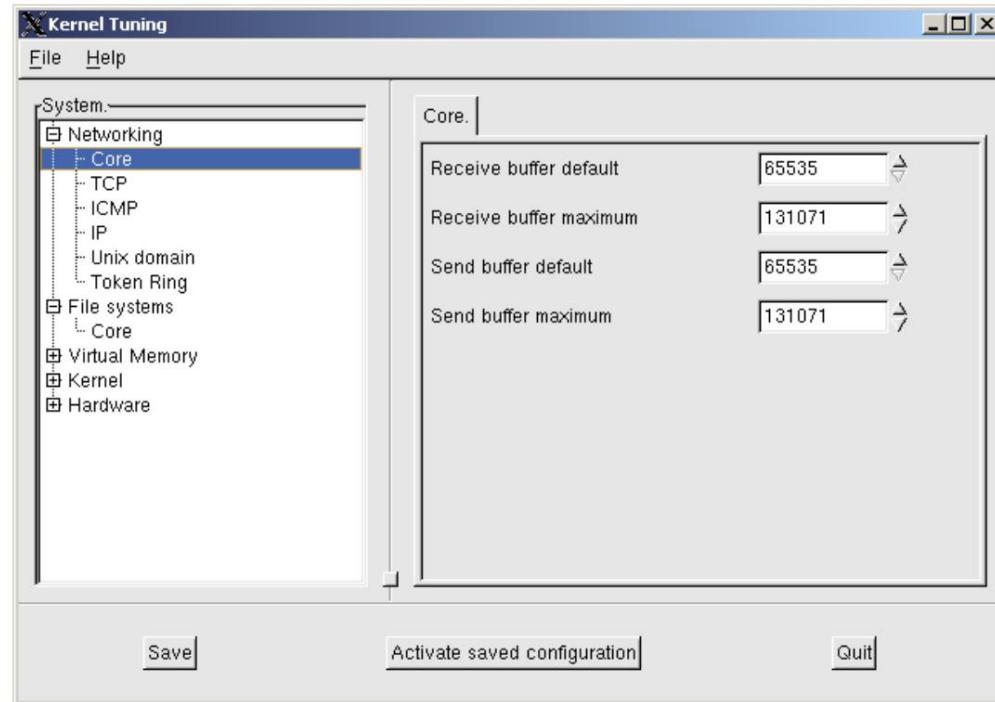


图4-5 Red Hat内核调优

对于基于 Novell SUSE 的系统,YaST 和更具体地说 powertweak 是更改任何内核参数的首选工具。

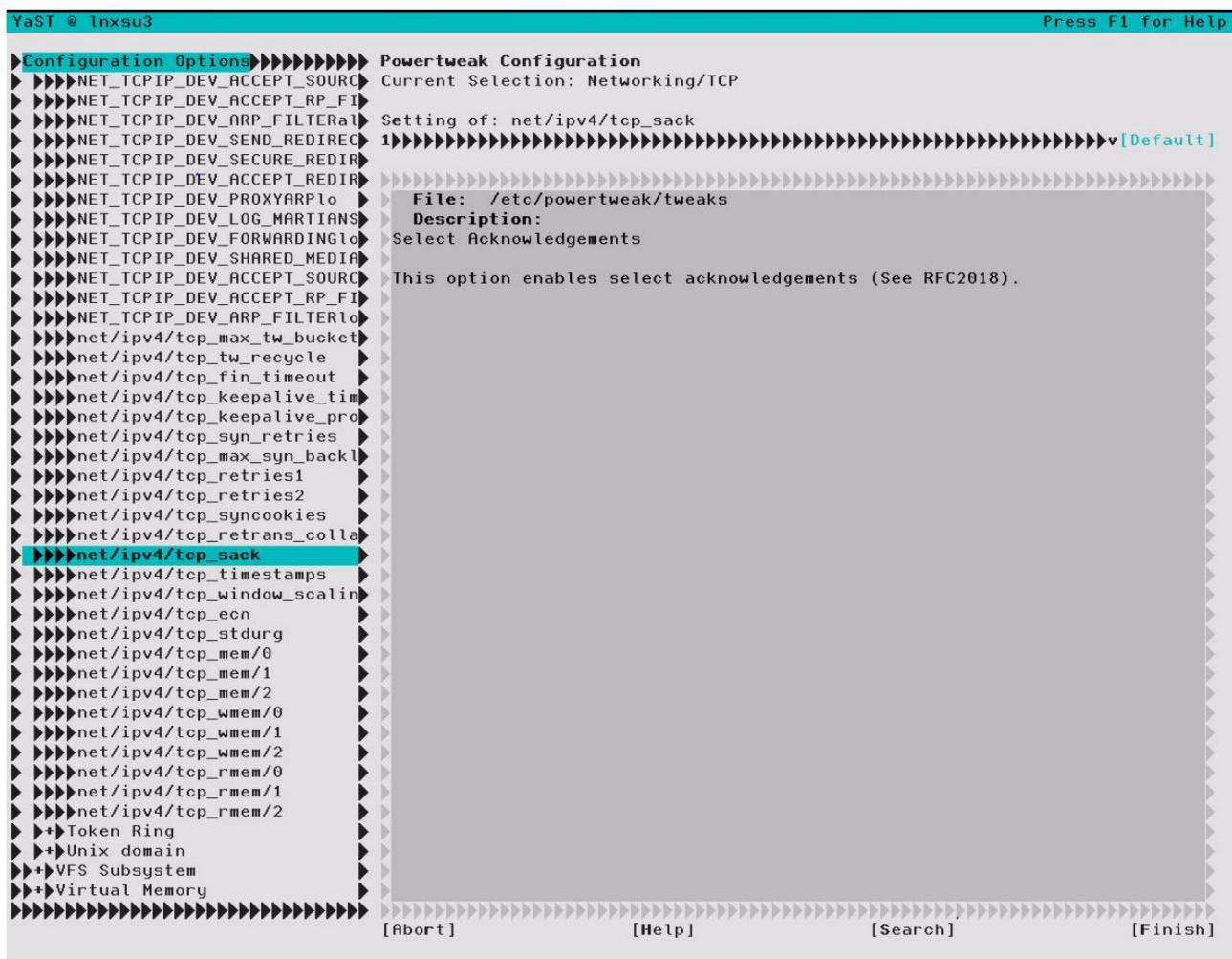


图 4-6 powertweak 实用程序

通过 `sysctl` 使用 powertweak 的一大优势在于,所有调整参数都会附有简短的说明。请注意,所有使用 powertweak 进行的更改都将存储在 `/etc/powertweak/tweaks` 下。

4.3.1 参数存储在哪里

控制内核行为的内核参数存储在 `/proc` (特别是 `/proc/sys`)中。

读取 `/proc` 目录树中的文件提供了一种简单的方法来查看与内核、进程、内存、网络和其他组件相关的配置参数。系统中运行的每个进程在 `/proc` 中都有一个以进程 ID (PID) 命名的目录。第 107 页的表 4-3 列出了一些包含内核信息的文件。

表4-3 /proc中的参数文件

文件/目录	目的
/proc/sys/abi/*	用于提供对 Linux 以外的“外部”二进制文件的支持,这些二进制文件是在其他 UNIX 变体(如 SCO UnixWare 7,SCO OpenServer 和 SUN Solaris™ 2)下编译的。默认情况下会安装此支持,但可以在安装过程中将其删除。
/proc/sys/fs/*	用于增加操作系统允许打开的文件数量并处理配额。
/proc/sys/内核/*	为了进行调整,您可以启用热插拔、操作共享内存以及指定 PID 文件的最大数量和系统日志中的调试级别。
/proc/sys/net/*	总体网络调整,IPV4 和 IPV6。
/proc/sys/vm/*	缓存和缓冲区的管理。

4.3.2 使用 sysctl 命令

sysctl命令使用 /proc/sys 目录树中的文件名作为参数。

例如,要修改 shmmmax 内核参数,可以显示 (使用cat)并更改 (使用echo)文件 /proc/sys/kernel/shmmmax:

```
#cat /proc/sys/kernel/shmmmax
33554432
#echo 33554430 > /proc/sys/kernel/shmmmax
#cat /proc/sys/kernel/shmmmax
33554430
```

但是,使用这些命令很容易引入错误,因此我们建议您使用sysctl命令,因为它会在进行任何更改之前检查数据的一致性。例如:

```
#sysctl kernel.shmmmax
内核.shmmmax = 33554432 #sysctl -w 内
核.shmmmax=33554430
内核.shmmmax = 33554430 #sysctl 内
核.shmmmax
内核.shmmmax = 33554430
```

此内核更改仅在下次重启前有效。如果您想永久更改,可以编辑 /etc/sysctl.conf 文件并添加相应的命令。在我们的示例中:

```
内核.shmmmax = 33554439
```

下次重启时,系统会读取该参数文件。您可以执行以下命令,无需重启即可完成相同的操作:

```
#sysctl -p
```

4.4 调整处理器子系统

在任何计算机中,无论是手持设备还是用于科学应用的集群,其主要子系统都是执行实际计算的处理器。在过去十年中,摩尔定律使得处理器子系统的发展速度显著快于其他子系统。因此,除非系统仅用于数字运算,否则 CPU 很少会出现瓶颈。英特尔兼容服务器系统的平均 CPU 利用率低于 10% 就证明了这一点。了解

处理器级别可能出现的瓶颈,并了解可能的调整参数以提高 CPU 性能。

4.4.1 调整进程优先级

正如我们在第 5 页 1.1.4 节 “进程优先级和 nice 级别”中所述,无法更改进程的优先级。这只能通过使用进程的 nice 级别间接实现,但即使这样也并非总是可行。如果某个进程运行速度过慢,您可以通过降低 nice 级别为其分配更多 CPU。当然,这意味着所有其他程序的处理器周期都会减少,运行速度也会变慢。

Linux 支持的 nice 级别从 19 (最低优先级) 到 -20 (最高优先级)。默认值为 0。要将程序的 nice 级别更改为负数 (以提高其优先级),需要登录或使用 su 命令以 root 身份运行。

要以良好级别 -5 启动程序 xyz,请发出以下命令:

```
很好-n-5 xyz
```

要更改已运行程序的良好级别,请发出以下命令:

```
renice 级别 pid
```

要将 PID 为 2500 的程序的优先级更改为良好级别 10,请发出:

```
renice 10 2500
```

4.4.2 中断处理的 CPU 亲和性

在中断处理方面,已证明有两个原则最有效 (有关中断处理的回顾,请参阅第 6 页 1.1.6 “中断处理”):

将导致大量中断的进程绑定到 CPU。

CPU 亲和性使系统管理员能够将中断绑定到一组或单个物理处理器 (当然,这不适用于单 CPU 系统)。要更改任何给定 IRQ 的亲和性,请进入 /proc/irq/%{各个 irq 的数量}/ 并更改存储在文件 smp_affinity 中的 CPU 掩码。要将 IRQ 19 的亲和性设置为系统中的第三个 CPU (无 SMT),请使用示例 4-6 中的命令。

示例 4-6 设置中断的 CPU 亲和性

```
[root@linux /]#echo 03 > /proc/irq/19/smp_affinity
```

让物理处理器处理中断。

在对称多线程 (SMT) 系统 (例如支持多线程的 IBM POWER 5+ 处理器) 中,建议您将中断处理绑定到物理处理器,而不是 SMT 实例。物理处理器的 CPU 编号通常较低,因此在启用多线程的双向系统中,CPU ID 0 和 2 指的是物理 CPU,而 1 和 3 指的是多线程实例。

如果您不使用 smp_affinity 标志,则不必担心这个问题。

4.4.3 NUMA 系统的注意事项

非统一内存架构 (NUMA) 系统正在赢得市场份额,并被视为传统对称多处理器系统的自然演进。尽管当前 Linux 发行版使用的 CPU 调度程序非常适合 NUMA 系统,但应用程序可能并非总是如此。非 NUMA 感知应用程序造成的瓶颈可能会导致

难以识别的性能下降。numactl 软件包中最新发布的numastat实用程序有助于识别在处理 NUMA 架构时遇到困难的进程。

为了帮助发现瓶颈， numastat工具提供的统计信息可在 /sys/devices/system/node/%{node number}/numastat 文件中获取。numa_miss 和 other_node 字段中的值较高表示可能存在 NUMA 问题。如果您发现某个进程分配的内存不在该进程的本地节点（即运行应用程序的处理器所在的节点），请尝试将该进程重新路由到另一个节点或使用 NUMA 关联性。

4.5 调整 vm 子系统

调整内存子系统是一项颇具挑战性的任务，需要持续监控以确保更改不会对服务器中的其他子系统产生负面影响。如果您选择修改虚拟内存参数（位于 /proc/sys/vm 中），我们建议您一次只更改一个参数，并监控服务器的性能。

请记住，Linux 下的大多数应用程序不会直接写入磁盘；它们会写入由虚拟内存管理器维护的文件系统缓存，最终会将数据刷新出来。使用 IBM ServeRAID 控制器或 IBM TotalStorage 磁盘子系统时，应尝试减少刷新次数，从而有效增加每次刷新产生的 I/O 流。高性能磁盘控制器可以更高效地处理较大的 I/O 流，而不是多个较小的 I/O 流。

4.5.1 设置内核交换和 pdflush 行为

随着 Linux 内核 2.6 中改进的虚拟内存子系统的引入，管理员现在可以通过简单的界面来微调内核的交换行为。

/proc/sys/vm/swappiness 中存储的参数可用于定义将内存页交换到磁盘的积极程度。第 14 页的“页框回收”介绍了 Linux 虚拟内存管理器以及 Linux 中交换空间的一般用法。文中指出，即使有足够的可用内存，Linux 也会将一段时间内未访问的内存页移动到交换空间。

通过更改 /proc/sys/vm/swappiness 中的百分比，您可以根据系统配置控制该行为。如果不需要交换，则 /proc/sys/vm/swappiness 应设置为较低的值。对于内存受限且运行批处理作业（长时间休眠的进程）的系统，积极的交换行为可能会更有利。要更改交换行为，请使用echo或sysctl，如示例 4-7 所示。

示例 4-7 更改 swappiness 行为

```
# sysctl -w vm.swappiness=100
```

尤其对于快速磁盘子系统，可能还需要触发大量脏内存页的刷新。/proc/sys/vm/dirty_background_ratio 中存储的值定义了 pdflush 守护进程应以主内存的百分比将数据写入磁盘。

如果需要更大的刷新频率，则将默认值 10% 增加到更大的值可以降低刷新频率。如上例所示，可以按照示例 4-8 中所示更改该值。

示例 4-8 增加 pdflush 的唤醒时间

```
# sysctl -w vm.dirty_background_ratio=25
```

虚拟内存子系统中的另一个相关设置是应用程序磁盘写入操作创建的脏页刷新到磁盘的速率。如第 1.3.1 章“虚拟文件系统”(第 15 页)中所述,对文件系统的写入操作不会立即写入,而是写入页面缓存,并在稍后刷新到磁盘子系统。

系统管理员可以使用存储在 /proc/sys/vm/dirty_ratio 中的参数来定义实际磁盘写入的级别。dirty_ratio 中存储的值

是主内存的百分比。值为 10 表示数据将一直写入系统内存,直到文件系统缓存的大小达到服务器 RAM 的 10%。与前两个示例一样,脏页写入磁盘的比率可以按如下方式更改为系统内存的 20%:

示例 4-9 改变脏比率

```
# sysctl -w vm.dirty_ratio=20
```

4.5.2 交换分区

当物理 RAM 已满且系统需要额外内存时,将使用交换设备。Linux 还使用交换空间将长期未访问的内存区域分页到磁盘。当系统上没有可用内存时,它会开始将内存中使用最少的数据分页到磁盘上的交换区域。初始交换分区是在 Linux 安装过程中创建的,当前指南规定交换分区的大小应为物理 RAM 的两倍。Linux 内核 2.4 及更高版本支持每个分区最大 24 GB 的交换大小,对于 32 位系统,理论上最大可达 8 TB。交换分区应位于单独的磁盘上。

如果在初始安装后向服务器添加了更多内存,则必须配置额外的交换空间。初始安装后,有两种方法可以配置额外的交换空间:

磁盘上的可用分区可以创建为交换分区。如果磁盘子系统没有可用空间,这可能会比较困难。在这种情况下,可以创建一个交换文件。

如果可以选择,首选方案是创建额外的交换分区。这样做可以提高性能,因为交换分区的 I/O 操作可以绕过文件系统以及写入文件所涉及的所有开销。

提高交换分区和文件性能的另一种方法是创建多个交换分区。Linux 可以利用多个交换分区或文件,并并行执行对磁盘的读写操作。创建额外的交换分区或文件后, /etc/fstab 文件将包含如示例 4-10 所示的条目。

示例 4-10 /etc/fstab 文件

/dev/sda2 /	交换	交换	西南	0 0
dev/sdb2 /	交换	交换	西南	0 0
dev/sdc2 /	交换	交换	西南	0 0
dev/sdd2	交换	交换	西南	0 0

正常情况下,Linux 会首先使用 /dev/sda2 交换分区,然后是 /dev/sdb2,依此类推,直到分配到足够的交换空间。这意味着,如果不需要较大的交换空间,可能只会使用第一个分区 /dev/sda2。

将数据分散到所有可用的交换分区可以提高性能,因为所有读/写请求都会同时对所有选定的分区执行。按照第 111 页的示例 4-11 所示更改文件,会为前三个分区分配更高的优先级。

示例 4-11 修改 /etc/fstab 以创建并行交换分区

/dev/sda2 /	交换	交换	sw,pri=3	0 0
dev/sdb2 /	交换	交换	sw,pri=3	0 0
dev/sdc2 /	交换	交换	sw,pri=3	0 0
dev/sdd2	交换	交换	sw,pri=1	0 0

交换分区的使用优先级从高到低（其中 32767 为最高，0 为最低）。如果将前三个磁盘的优先级设置为相同，则数据会写入所有三个磁盘；系统不会等到第一个交换分区写满后才开始写入下一个分区。系统会并行使用前三个分区，性能通常会得到提升。

如果前三个分区完全填满后需要额外的空间用于交换，则使用第四个分区。也可以为所有分区赋予相同的优先级，以便将数据条带化到所有分区上，但如果一个驱动器比其他驱动器慢，性能就会下降。一般规则是，交换分区应该位于速度最快的可用驱动器上。

重要提示：虽然有一些不错的工具可以调整内存子系统，但应尽可能避免频繁的页面调出。交换空间不能替代 RAM，因为它存储在访问时间比内存慢得多的物理驱动器上。因此，频繁的页面调出（或交换出）几乎从来都不是一个好的行为。在尝试改进交换过程之前，请确保您的服务器有足够的内存或没有内存泄漏。

4.5.3 巨型TLBfs

此内存管理功能对于使用大型虚拟地址空间的应用程序非常有用，尤其适用于数据库应用程序。

CPU 的转换后备缓冲区 (TLB) 是一个小型缓存，用于存储虚拟地址到物理地址的映射信息。使用 TLB，转换无需引用映射虚拟地址的内存页表条目。但是，为了尽可能加快转换速度，TLB 通常较小。大内存应用程序超出 TLB 的映射容量的情况并不少见。

HugeTLBfs 功能允许应用程序使用比正常情况大得多的页面大小，从而使单个 TLB 条目可以映射更大的地址空间。HugeTLB 条目的大小可能有所不同。例如，在 Itanium® 2 系统中，大页面可能比正常页面大 1000 倍。这使得 TLB 能够映射正常进程 1000 倍的虚拟地址空间，而不会导致 TLB 缓存未命中。为简单起见，此功能通过文件系统接口提供给应用程序。

要分配大页面，您可以使用 sysctl 命令配置 /proc/sys/vm/nr_hugepages 的值来定义大页面的数量。

```
sysctl -w vm.nr_hugepages=512
```

如果您的应用程序通过 mmap() 系统调用使用大页面，则必须挂载 hugetlbfs 类型的文件系统，如下所示：

```
安装 -t hugetlbfs none /mnt/hugepages
```

/proc/meminfo 文件将提供有关 hugetlb 页的信息，如第 112 页上的示例 4-12 所示。

示例 4-12 /proc/meminfo 中的大页面信息

```
[root@lnxsu4 ~]# cat /proc/meminfo
内存总量:4037420 kB
可用内存:386664 kB
缓冲区:60596 kB
缓存:238264 kB
交换缓存:0 kB
活动:364732 kB
未激活:53908 kB
总计:0 kB
高可用:0 kB
低总计:4037420 kB
低免费:386664 kB
交换总量:2031608 kB
交换空间:2031608 kB
脏:0 kB
回写:0 kB
映射:148620 kB
板坯:24820 kB
提交限制:2455948 kB
已提交:166644 kB
页表:2204 kB
Vmalloc总计:536870911 kB
已使用的 Vmalloc: 263444 千字节
VmallocChunk:536607255 kB
HugePages_Total:1557
HugePages_Free:1557
大页面大小:2048 kB
```

请参阅 Documentation/vm/hugetlbpage.txt 中的内核文档以获取更多信息。

4.6 调整磁盘子系统

最终,所有数据都必须从磁盘检索并存储到磁盘。磁盘访问通常以毫秒为单位,比其他组件(例如内存和 PCI 操作,以纳秒或微秒为单位)慢至少数千倍。Linux 文件系统是数据存储和管理磁盘的方法。

Linux 有许多不同的文件系统,它们的性能和可扩展性各不相同。

除了存储和管理磁盘上的数据外,文件系统还负责保证数据的完整性。较新的 Linux 发行版默认安装包含日志文件系统。日志记录功能可防止系统崩溃时出现数据不一致。所有对文件系统元数据的修改都保存在单独的日志中,系统崩溃后可应用这些修改,使其恢复到一致状态。日志记录还可以缩短恢复时间,因为系统重启时无需执行文件系统检查。与计算的其他方面一样,您会发现性能和完整性之间存在权衡。然而,随着 Linux 服务器逐渐进入企业数据中心和企业环境,诸如高可用性之类的需求可以得到满足。

除了各种文件系统之外,Linux 内核 2.6 还支持 4 种不同的 I/O 调度算法,这些算法也可用于根据特定任务定制系统。每种 I/O 调度算法都有其独特的特性,这些特性可能使其适合或不适合特定的硬件配置和所需的任务。有些 I/O 调度算法强调流式 I/O,因为它们通常用于多媒体或桌面 PC 环境,而另一些 I/O 调度算法则侧重于数据库工作负载所需的低延迟访问时间。

在本节中,我们介绍标准文件系统 (如 ReiserFS 和 Ext3) 的特性和调整选项,以及内核 2.6 I/O 提升中的调整潜力。

4.6.1 安装 Linux 之前的硬件考虑

当前 Linux 发行版对 CPU 速度和内存的最低要求都有详尽的文档。这些说明还提供了完成安装所需的最小磁盘空间指导。然而,它们未能解释如何初始设置磁盘子系统。Linux 服务器涵盖了各种各样的工作环境,因为服务器整合会影响数据中心。首先要回答的问题之一是:正在安装的服务器的功能是什么?

服务器的磁盘子系统是整个系统性能的主要组成部分。

了解服务器的功能是确定 I/O 子系统是否会对性能产生直接影响的关键。

磁盘 I/O 是最重要的子系统的服务器示例:

文件和打印服务器必须在用户和磁盘子系统之间快速移动数据。

由于文件服务器的目的是将文件传送给客户端,因此服务器必须首先从磁盘读取所有数据。

数据库服务器的最终目标是从磁盘上的存储库中搜索和检索数据。即使内存充足,大多数数据库服务器也会执行大量的磁盘 I/O 操作,将数据记录载入内存,并将修改后的数据刷新到磁盘。

磁盘 I/O 不是最重要的子系统的服务器示例:

电子邮件服务器充当电子邮件的存储库和路由器,往往会产生繁重的通信负载。对于此类服务器来说,网络连接更为重要。

负责托管网页 (静态、动态或两者) 的 Web 服务器受益于良好调整的网络和内存子系统。

驱动器数量

磁盘驱动器的数量会显著影响性能,因为每个驱动器都会影响系统总吞吐量。容量需求通常是确定服务器配置磁盘驱动器数量的唯一考虑因素。吞吐量需求通常不太为人所知,甚至会被完全忽略。高性能磁盘子系统的关键在于最大限度地增加能够处理 I/O 请求的读写磁头数量。

借助 RAID (独立磁盘冗余阵列) 技术,您可以将 I/O 分散到多个磁盘轴上。在 Linux 环境中实现 RAID 有两种方案:软件 RAID 和硬件 RAID。除非您的服务器硬件标配硬件 RAID,否则您可能希望从 Linux 发行版自带的软件 RAID 选项开始。如果有需要,您可以逐步过渡到更高效的硬件 RAID 解决方案。

如果需要实现硬件 RAID 阵列,则系统需要一个 RAID 控制器。在这种情况下,磁盘子系统由物理硬盘和控制器组成。

提示:一般来说,添加驱动器是提高服务器性能的最有效的改变之一。

务必记住,磁盘子系统的性能最终取决于给定设备能够处理的输入输出请求数量。一旦操作系统缓存和磁盘子系统的缓存无法再容纳读写请求的数量或大小,物理磁盘主轴就必须工作。请考虑以下示例。一个磁盘设备每秒能够处理 200 次 I/O。您的应用程序在文件系统上的随机位置执行 4 KB 的写入请求,因此流式传输或请求合并不适用。指定磁盘子系统的最大吞吐量现在为:

$$\text{物理磁盘每秒 I/O * 请求大小} = \text{最大吞吐量}$$

因此,上面的例子导致:

$$200 * 4 \text{ KB} = 800 \text{ KB}$$

由于 800 KB 是物理最大值,因此在这种情况下提升性能的唯一途径是添加更多主轴或物理磁盘,或者让应用程序写入更大的 I/O。DB2 等数据库可以配置为使用更大的请求大小,这在大多数情况下可以提高磁盘吞吐量。

有关可用 IBM 存储解决方案的更多信息,请参阅:

IBM 系统存储解决方案手册,SG24-5250

存储区域网络简介,SG24-5470

设置分区的指南

分区是驱动器上一组连续的块,它们被视为独立的磁盘。当今企业级 Linux 发行版的默认安装通过创建一个或多个逻辑卷来使用灵活的分区布局。

在 Linux 圈子里,关于最佳磁盘分区的争论非常激烈。如果您决定根据新的或更新的需求重新定义分区,那么单根分区方法将来可能会导致问题。另一方面,过多的分区可能会导致文件系统管理问题。在安装过程中,Linux 发行版允许您创建多分区布局。

在多分区甚至逻辑卷磁盘上运行 Linux 有以下好处:

通过更细粒度的文件系统属性提高了安全性。

例如, /var 和 /tmp 分区具有允许系统上的所有用户和进程轻松访问的属性,并且容易受到恶意访问。

通过将这些分区隔离到单独的磁盘,您可以减少在需要重建或恢复这些分区时对系统可用性的影响。

提高了数据完整性,因此磁盘崩溃造成的数据丢失将被隔离到受影响的分区。

例如,如果系统上没有 RAID 实现(软件或硬件),并且服务器遭遇磁盘崩溃,则只需要修复或恢复该坏磁盘上的分区。

可以在不影响其他更静态的分区的情况下进行新的安装和升级。

例如,如果 /home 文件系统尚未分离到另一个分区,它将在操作系统升级期间被覆盖,从而丢失存储在其中的所有用户文件。

更高效的备份流程

设计分区布局时必须考虑备份工具。务必了解备份工具是在分区边界上运行,还是在更细粒度的级别 (例如文件系统) 上运行。

表 4-4 列出了您可能需要考虑从根中分离出来的一些分区,以便在您的环境中提供更大的灵活性和更好的性能。

表 4-4 Linux 分区和服务器环境

分割	内容和可能的服务器环境
/家	对于文件服务器环境来说,将 /home 目录划分到单独的分区会非常有益。如果没有设置磁盘配额,那么这个目录就是系统上所有用户的主目录,因此,划分这个目录应该可以避免用户过度消耗磁盘空间。
/tmp	如果您正在运行高性能计算环境,则在计算期间需要大量临时空间,然后在完成后释放。
/usr	内核源代码树和 Linux 文档 (以及大多数可执行二进制文件) 都位于此目录。/usr/local 目录存储系统上所有用户必须访问的可执行文件,也是存储为您的环境开发的自定义脚本的理想位置。如果将其划分到单独的分区,则升级时无需重新安装文件,或者只需选择不重新格式化分区即可重新安装。
/var	/var 分区在邮件、Web 和打印服务器环境中非常重要,因为它包含这些环境的日志文件以及整个系统日志。长期邮件可能会淹没并填满此分区。如果发生这种情况,并且该分区未与 / 分区分开,则可能会导致服务中断。根据环境的不同,可以进一步划分此分区,例如,将邮件服务器的 /var/spool/mail 分区与系统日志的 /var/log 分区分开。
/选择	某些第三方软件产品 (例如 Oracle 数据库服务器) 的安装默认在此分区进行。如果没有单独设置,安装将继续在 / 下进行,如果分配的空间不足,安装可能会失败。

要详细了解 Linux 发行版如何处理文件系统标准,请参阅
文件系统层次结构标准的主页位于:

<http://www.pathname.com/fhs>

4.6.2 I/O 电梯调节与选择

Linux 内核 2.6 引入了新的 I/O 调度算法,以便在处理不同的 I/O 模式时提供更大的灵活性。系统管理员现在需要根据给定的硬件和软件布局选择最合适的 I/O 提升算法。此外,每个 I/O 提升算法都提供一组调优选项,以便进一步根据特定工作负载定制系统。

在内核 2.6 中选择正确的 I/O 提升器

对于大多数服务器工作负载,完全公平队列 (CFQ) 提升器或截止期限提升器都是合适的选择,因为它们针对典型服务器运行的多用户、多进程环境进行了优化。企业发行版通常默认使用 CFQ 提升器。然而,在 IBM System z 的 Linux 上,截止期限调度器被优先用作默认提升器。某些环境可以从选择不同的 I/O 提升器中受益。在 Red Hat Enterprise Linux 5.0 和 Novell SUSE Linux Enterprise Server 10 中,现在可以基于每个磁盘子系统选择 I/O 调度器,而不是像 Red Hat Enterprise Linux 那样需要全局设置。

Hat Enterprise Linux 4.0 和 Novell SUSE Linux Enterprise Server 9。由于每个磁盘子系统可以有不同的 I/O 升降机,管理员现在可以隔离磁盘子系统上的特定 I/O 模式 (例如写入密集型工作负载) 并选择适当的升降机算法。

同步文件系统访问

某些类型的应用程序需要同步执行文件系统操作。对于甚至可能使用原始文件系统的数据库,或者对于无法缓存异步磁盘访问的大型磁盘子系统来说,情况可能确实如此。在这些情况下,预期电梯的性能通常吞吐量最低,延迟最高。其他三个调度器的性能同样出色,直到 I/O 大小达到约 16 KB 时,CFQ 和 NOOP 电梯的性能开始优于截止期限电梯 (除非磁盘访问非常频繁地寻道),如图 4-7 所示。

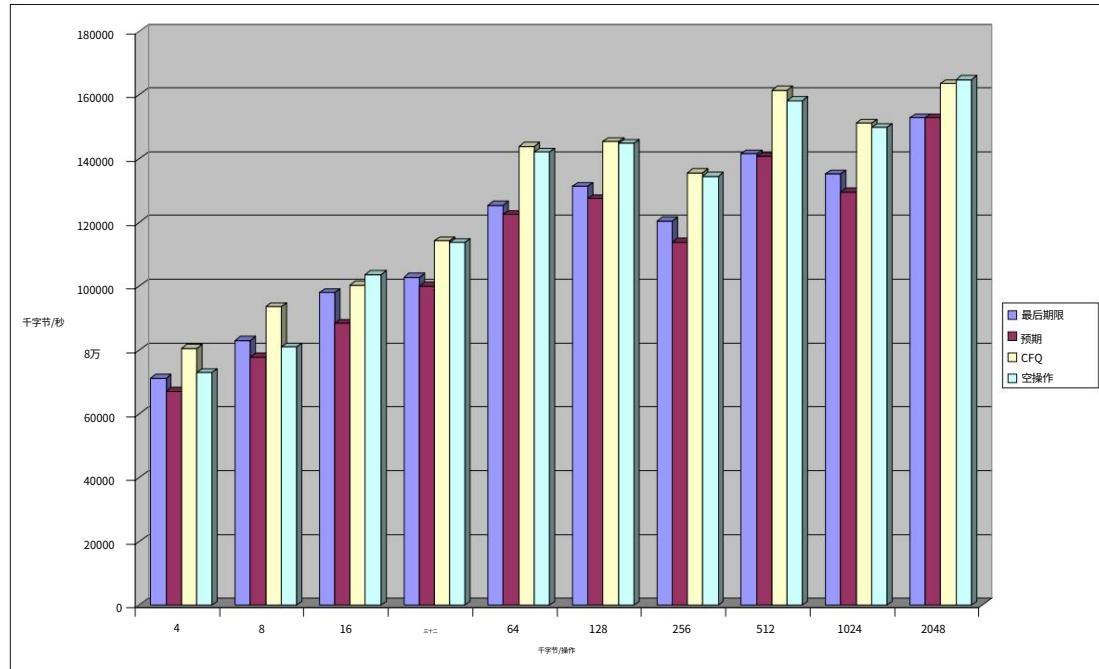


图 4-7 每个 I/O 电梯的随机读取性能 (同步)

复杂的磁盘子系统

基准测试表明,NOOP 提升器在高端服务器环境中是一个有趣的替代方案。当使用 IBM ServeRAID 或 TotalStorage® DS 级磁盘子系统的非常复杂的配置时,NOOP 提升器缺乏排序功能反而成为了它的优势。企业级磁盘子系统可能包含多个 SCSI 或光纤通道磁盘,每个磁盘都有独立的磁头,数据分布在多个磁盘上。对于 I/O 提升器来说,准确预测此类复杂子系统的 I/O 特性变得非常困难,因此,使用 NOOP I/O 提升器时,您通常可以观察到至少相同的性能,而开销却更低。大多数使用数百个磁盘的大规模基准测试很可能使用 NOOP 提升器。

数据库系统

由于大多数数据库工作负载都具有面向寻道的特性,因此在为这些工作负载选择截止期限提升器时可以获得一定的性能提升。

虚拟机

虚拟机,无论是 VMware 还是 System z 的 VM,通常都通过虚拟化层与底层硬件进行通信。因此,虚拟

机器无法识别 TotalStorage DS8000™ 上分配的磁盘设备是单个 SCSI 设备还是光纤通道磁盘阵列。虚拟化层负责必要的 I/O 重新排序以及与物理块设备的通信。

CPU密集型应用程序

虽然某些 I/O 调度器可以提供卓越的吞吐量,但它们同时也会产生更大的系统开销。例如,CFQ 或 Deadline 电梯调度器带来的开销就来自于频繁合并和重新排序 I/O 队列。

有时,工作负载并非完全受磁盘子系统性能的限制,而是受 CPU 性能的限制。这种情况可能发生在科学计算工作负载或处理非常复杂查询的数据仓库中。在这种情况下,NOOP 电梯比其他电梯更具优势,因为它的 CPU 开销更小,如下图所示。然而,也需要注意的是,在比较 CPU 开销和吞吐量时,对于大多数异步文件系统的访问模式来说,截止期限和 CFQ 电梯仍然是最佳选择。

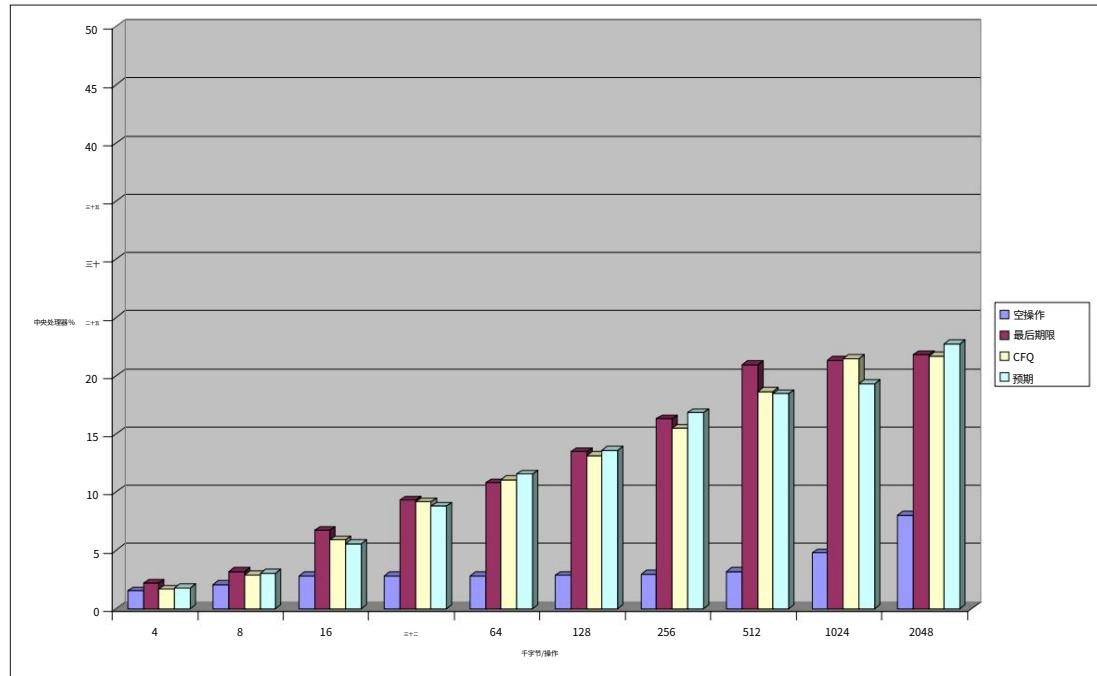


图 4-8 I/O 电梯的 CPU 利用率 (异步)

单 ATA 或 SATA 磁盘子系统

如果您选择使用单个物理 ATA 或 SATA 磁盘,请考虑使用预期 I/O 升降机,它会重新排序磁盘写入以适应这些设备中的单个磁盘头。

nr_requests 的影响

内核 2.6 的可插入式 I/O 调度器实现还提供了一种增加或减少可发送到磁盘子系统的请求数量的方法。nr_requests 与许多其他调优参数一样,没有最佳设置。请求数量的正确值很大程度上取决于底层磁盘子系统,甚至更多地取决于工作负载的 I/O 特性。不同 nr_requests 值的影响也可能因您计划使用的文件系统和 I/O 调度器而异,这一点可以从第 118 页的图 4-9 和图 4-10 中显示的两个基准测试中轻松看出。

在第 119 页上。如图 4-9 中所示,与 CFQ 电梯相比,Deadline 电梯不太容易受到 nr_requests 值不同所导致的变化的影响。

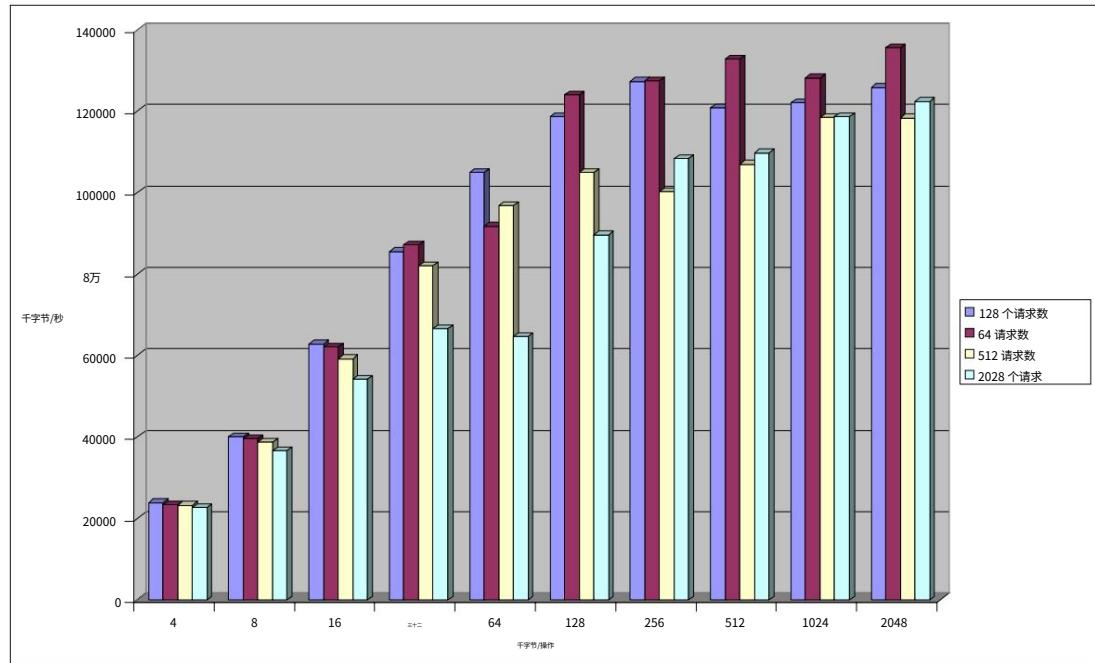


图4-9 nr_requests对Deadline电梯的影响 (随机写入ReiserFS)

对于写入大量小文件的工作负载,更大的请求队列可能会提供更高的吞吐量。如第 119 页图 4-10 所示,8192 的设置对于高达 16 KB 的 I/O 大小可提供最高的性能。在 64 KB 的情况下,从 64 到 8192 的 nr_requests 分析值提供的性能大致相同。然而,随着 I/O 大小的增加,在大多数情况下,较低的 nr_requests 级别会带来更优的性能。可以使用以下命令更改请求数量:

示例 4-13 更改 nr_requests

```
# echo 64 > /sys/block/sdb/queue/nr_requests
```

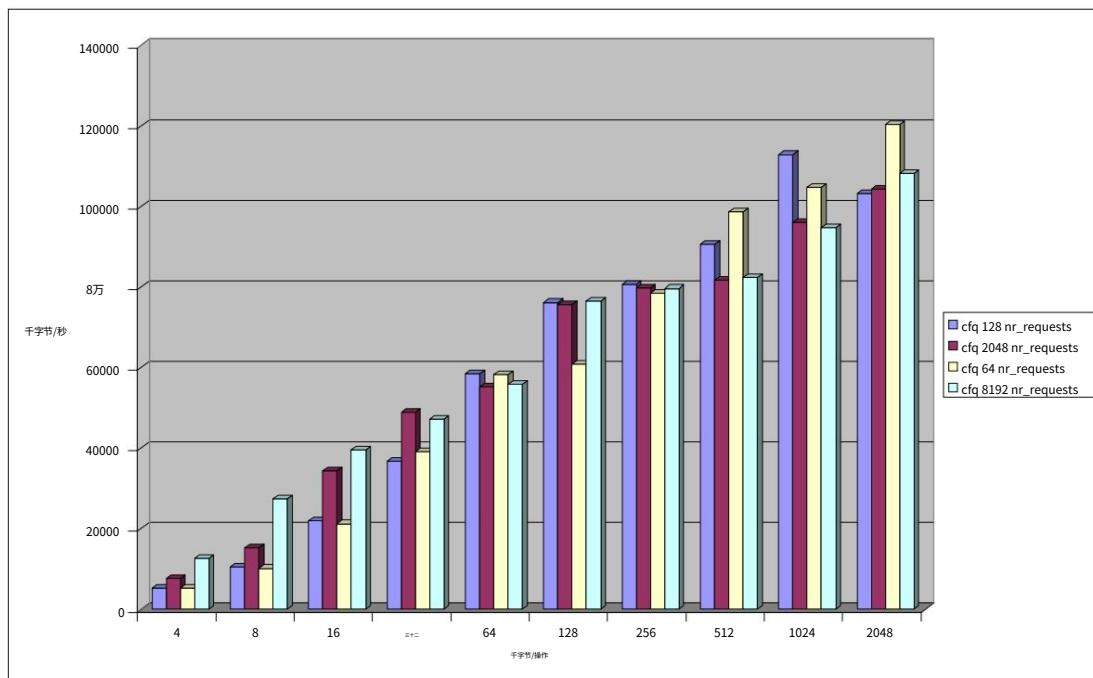


图 4-10 nr_requests 对 CFQ 电梯的影响 (随机写入 Ext3)

需要指出的是,Red Hat 和 Linux 的当前企业发行版提供了基于每个磁盘子系统设置 nr_requests 的选项。因此,可以隔离 I/O 访问模式并进行最佳调优。例如,一个数据库系统将日志分区和数据库存储在专用磁盘或磁盘子系统上(例如 DS8300 上的存储分区)。在这个例子中,对于必须容纳大量小型写入 I/O 的日志分区,使用较大的 nr_requests 值;对于可能遇到高达 128 KB 的读取 I/O 的数据库分区,使用较小的值将非常有益。

提示:要了解如何测量和计算平均 I/O 大小,请参阅第 48 页上的 2.3.6 “iostat”。

read_ahead_kb 的影响

对于大量流式读取,增加预读缓冲区的大小可能会提高性能。请记住,增加此值不会提高大多数服务器工作负载的性能,因为这些工作负载主要是随机 I/O 操作。read_ahead_kb 中的值定义了预读操作的最大大小。/sys/block/<磁盘子系统>/queue/read_ahead_kb 中的值定义了读取操作的最大大小(以 KB 为单位)。可以使用 cat 或 echo 命令解析或更改该值,如示例 4-14 所示。

示例 4-14 解析并设置预读操作的大小

```
# cat /sys/block/<磁盘子系统>/queue/read_ahead_kb
# echo 64 > /sys/block/<磁盘子系统>/queue/read_ahead_kb
```

4.6.3 文件系统选择和调整

如第 15 页 1.3 节“Linux 文件系统”所述,Linux 可用的不同文件系统在设计时考虑了不同的工作负载和可用性特性。如果您的 Linux 发行版和应用程序允许选择不同的文件系统,则可能需要研究一下 Ext、日志文件系统 (JFS)、ReiserFS 或扩展文件系统 (XFS) 是否是计划工作负载的最佳选择。一般而言,ReiserFS 更适合处理小型 I/O 请求,而 XFS 和 JFS 则适用于非常大的文件系统和非常大的 I/O 大小。Ext3 填补了 ReiserFS 和 JFS/XFS 之间的空白,因为它可以处理小型 I/O 请求,同时提供良好的多处理器可扩展性。

JFS 和 XFS 这两种工作负载模式最适合高端数据仓库、科学计算工作负载、大型 SMP 服务器或流媒体服务器。而 ReiserFS 和 Ext3 通常用于文件、Web 或邮件服务。对于写入密集型工作负载 (产生的 I/O 大小不超过 64 KB), ReiserFS 可能比采用默认日志模式的 Ext3 更具优势,如图 4-11 所示。然而,这仅适用于同步文件操作。

一个值得考虑的选项是 Ext2 文件系统。由于缺乏日志功能,Ext2 在同步文件系统访问方面的表现优于 ReiserFS 和 Ext3,无论访问模式和 I/O 大小如何。因此,当性能比数据完整性更重要时,Ext2 可能是一个不错的选择。

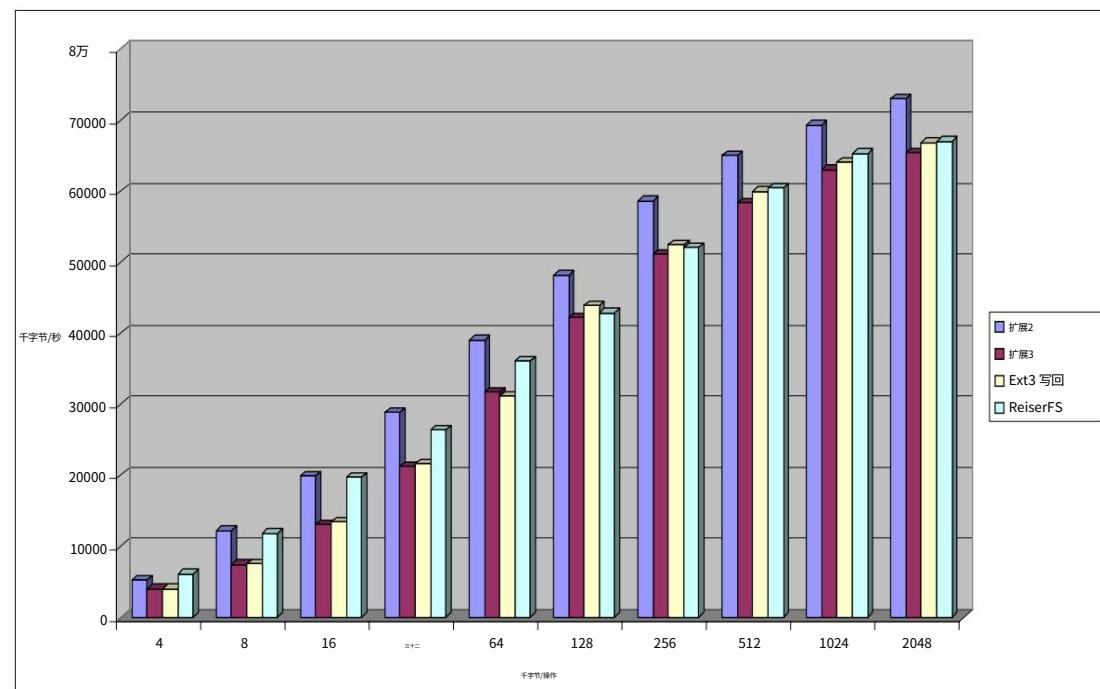


图4-11 Ext与ReiserFS随机写吞吐量对比（同步）

在最常见的异步文件系统场景中,ReiserFS 通常能够提供稳定的性能,并且在默认日志模式 (data=ordered) 下优于 Ext3。然而,需要注意的是,一旦默认日志模式切换为回写模式,Ext3 的性能就会与 ReiserFS 相当,如下图所示 (参见第 121 页的图 4-12)。

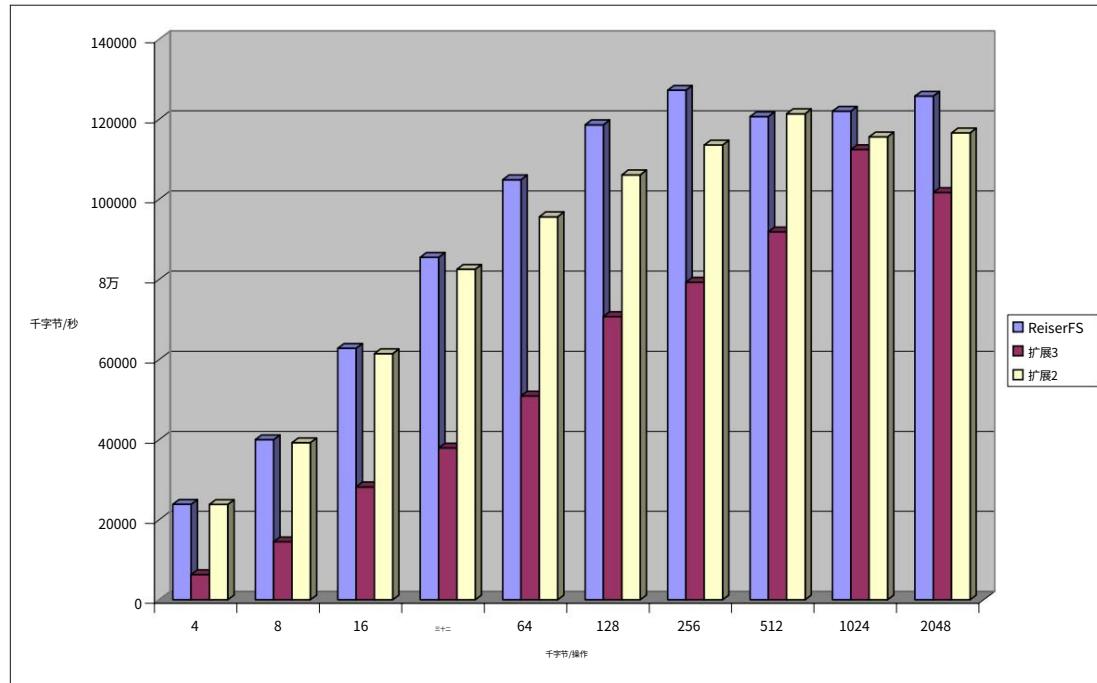


图4-12 Ext3与ReiserFS随机写吞吐量对比 (异步)

使用 ionice 分配 I/O 优先级

CFQ I/O 电梯的一个新功能是可以在进程级别分配优先级。

使用ionice实用程序,现在可以限制特定进程的磁盘子系统利用率。在撰写本文时,可以使用ionice 分配三个优先级,分别是:

空闲:I/O 优先级为空闲的进程,只有在没有其他优先级为“尽力而为”或更高的进程请求访问数据时,才会被授予访问磁盘子系统的权限。此设置对于那些仅在系统有空闲资源时运行的任务 (例如updatedb任务)非常有用。

尽力而为:默认情况下,所有未请求特定 I/O 优先级的进程都会被分配到此类。进程将继承其各自 CPU 优先级的 8 个等级,并将其分配给 I/O 优先级类别。

实时:可用的最高 I/O 优先级为实时,这意味着相应进程将始终被赋予访问磁盘子系统的优先权。实时优先级设置也可以接受 8 个优先级。为线程分配实时优先级时应谨慎,因为该进程可能会导致其他任务资源匮乏。

ionice工具接受以下选项:

- c<#> I/O 优先级1 表示实时,2 表示尽力,3 表示空闲
- n<#> I/O 优先级类别数据 0 至 7
- p<#> 正在运行的任务的进程 ID,不使用 -p 来启动具有相应 I/O优先级

第 122 页的示例 4-15 显示了运行ionice的示例,其中ionice用于为 PID 为 113 的进程分配空闲 I/O 优先级。

示例 4-15 ionice 命令

```
# ionice -c3 -p113
```

访问时间更新

Linux 文件系统保存文件创建、更新和访问的时间记录。

默认操作包括在读写文件期间更新文件的上次读取属性。由于写入操作开销巨大，因此消除不必要的 I/O 可以提升整体性能。然而，在大多数情况下，禁用文件访问时间更新只会带来非常小的性能提升。

使用noatime选项挂载文件系统可防止更新 inode 访问时间。如果文件和目录的更新时间对您的实现并不重要（例如在 Web 服务环境中），管理员可以选择在 /etc/fstab 文件中使用 noatime 标志挂载文件系统，如示例 4-16 所示。禁用写入文件系统的访问时间更新的性能提升范围为 0% 到 10%，对于文件服务器工作负载，平均提升 3%。

示例 4-16 在已挂载的文件系统上设置 noatime 选项来更新 /etc/fstab 文件

```
/dev/sdb1 /mountlocation ext3 默认值,noatime 1 2
```

提示:通常，最好有一个单独的 /var 分区并使用 noatime 选项挂载它。

选择文件系统的日志模式

大多数文件系统的三个日志选项可以通过mount中的 data 选项来设置

命令。但是，日志模式对 Ext3 文件系统的性能影响最大，因此我们建议您主要将此调整选项用于 Red Hat 的默认文件系统：

数据=日志

此日志选项通过同时记录文件数据和元数据来提供最高级别的数据一致性。但它的性能开销也更高。

数据=有序（默认）

在此模式下，仅写入元数据。但是，保证首先写入文件数据。

这是默认设置。

数据=回写

此日志选项以牺牲数据一致性为代价，提供最快的数据访问速度。由于元数据仍在记录中，因此可以保证数据的一致性。

但是，由于没有对实际文件数据进行特殊处理，这可能导致系统崩溃后文件中仍保留旧数据。需要注意的是，使用回写模式时实现的元数据日志记录类型与 ReiserFS、JFS 或 XFS 的默认设置相当。回写日志记录模式可提升 Ext3 的性能，尤其是在处理小规模 I/O 时，如图 4-13（第 123 页）所示。随着 I/O 规模的增加，使用回写日志记录的优势会逐渐减弱。另请注意，文件系统的日志记录模式仅影响写入性能。因此，主要执行读取操作的工作负载（例如 Web 服务器）不会因更改日志记录模式而受益。

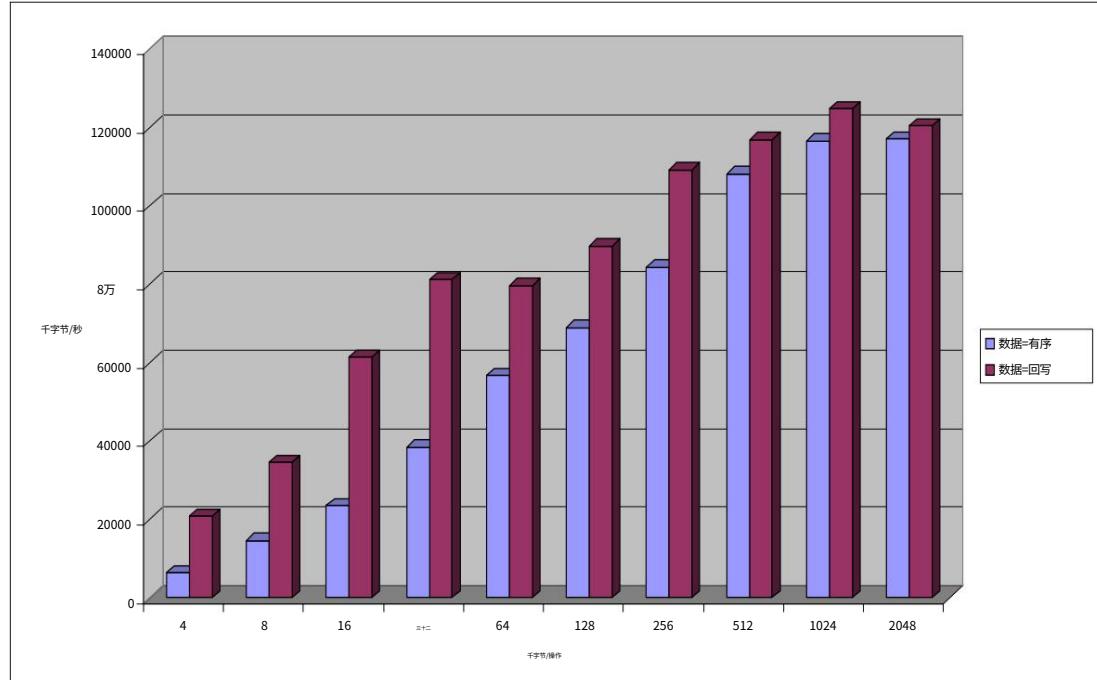


图 4-13 data=writeback 对随机写性能的影响

有三种方法可以更改文件系统上的日志模式：

执行挂载命令时：

```
mount -o数据=写回/dev/sdb1 /mnt/mountpoint
```

- /dev/sdb1 是正在挂载的文件系统。

将其包含在 /etc/fstab 文件的选项部分中：

```
/dev/sdb1 /testfs ext3默认值,数据=写回0 0
```

如果您想修改根分区上的默认 data=ordered 选项，请修改上面列出的 /etc/fstab 文件，然后执行mkinitrd 命令扫描 /etc/fstab 文件中的更改并创建新的映像。更新 grub 或 lilo 以指向新映像。

块大小

块大小是驱动器上可读写的最小数据量，它会直接影响服务器的性能。一般来说，如果您的服务器处理大量小文件，那么较小的块大小会更高效。如果您的服务器专用于处理大文件，那么较大的块大小可能会提高性能。现有文件系统上的块大小无法动态更改，只有重新格式化才能修改当前的块大小。大多数 Linux 发行版允许的块大小在 1 K、2 K 和 4 K 之间。基准测试表明，更改文件系统的块大小几乎不会带来任何性能提升，因此通常最好将其保留为默认值 4 K。

使用硬件 RAID 解决方案时，必须仔细考虑阵列（或光纤通道中为段）的条带大小。条带单元大小是指数据存储在阵列中一个驱动器上，后续数据存储在阵列中下一个驱动器上的粒度。选择正确的条带大小取决于了解特定应用程序执行的主要请求大小。与文件系统的块大小相比，硬件阵列的条带大小对整体磁盘性能有显著影响。

流式和顺序内容通常会因较大的条带大小而受益,因为它可以减少磁盘头寻道时间并提高吞吐量,但对于更随机类型的活动(例如数据库中的活动),当条带大小等于记录大小时,其性能会更好。

4.7 调整网络子系统

首次安装操作系统时,以及当网络子系统出现明显瓶颈时,应该调整网络子系统。此处的问题可能会影响其他子系统:例如,CPU利用率会受到显著影响,尤其是在数据包过小的情况下;如果TCP连接数量过多,内存占用也会增加。

4.7.1 交通特性的考虑

网络性能调优最重要的考虑因素之一是尽可能准确地理解网络流量模式。性能会因网络流量特性的不同而有很大差异。

例如,以下两张图显示了使用netperf测试吞吐量性能的结果,它们展现了不同的性能特征。唯一的区别在于流量类型。图4-14显示了TCP_RR类型流量和TCP_CRR类型流量的结果(请参阅2.4.3,“netperf”,第73页)。这种性能差异主要是由TCP会话连接和关闭操作的开销造成的,而主要因素是Netfilter连接跟踪(请参阅4.7.6,“Netfilter对性能的影响”,第132页)。

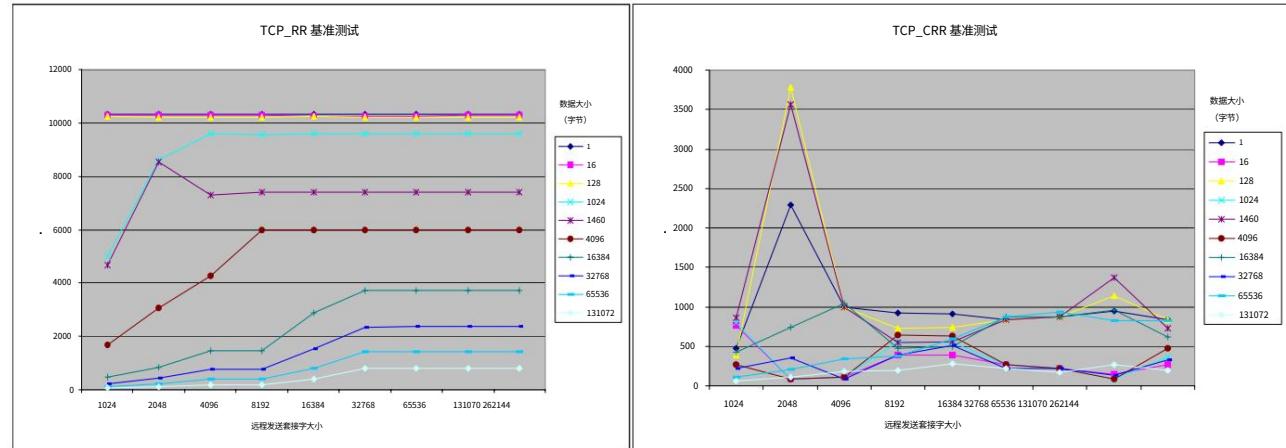


图4-14 netperf TCP_RR 和 TCP_CRR 基准测试的示例结果

正如我们在此处所示,即使配置完全相同,流量特性的细微差异也会导致性能差异巨大。您应该熟悉以下网络流量特性和要求:交易吞吐量要求(峰值、平均值)

- 数据传输吞吐量要求(峰值、平均)
- 延迟要求
- 传输数据大小
- 发送和接收的比例
- 连接建立和关闭的频率或并发连接数
- 协议(TCP、UDP和应用协议,如HTTP、SMTP、LDAP等)

netstat、tcpdump和ethereal是获取更准确特性的有用工具（请参阅第 53 页上的 2.3.11 “netstat”和第 55 页上的 2.3.13 “tcpdump / ethereal”）。

4.7.2 速度和双工

提高网络性能最简单的方法之一是检查网络接口的实际速度，因为网络组件（例如交换机或集线器）和网络接口卡之间可能存在问题是。不匹配可能会对性能造成很大影响，如示例 4-17 所示。

示例 4-17 使用 ethtool 检查实际速度和双工设置

```
[root@linux ~]# ethtool eth0
eth0 的设置:
支持的端口:[ MII ]
支持的链接模式:10baseT/Half 10baseT/Full
100baseT/半双工 100baseT/全双工
1000baseT/半双工 1000baseT/全双工

支持自动协商:是
公布的链接模式:10baseT/Half 10baseT/Full
100baseT/半双工 100baseT/全双工
1000baseT/半双工 1000baseT/全双工

公告自动协商:是
速度:100Mb/s
双工:全双工
```

从图 4-15 所示的基准测试结果可以看出，当网络速度协商不正确时，小数据传输受到的影响小于大数据传输。大于 1 KB 的数据传输会显著影响性能（吞吐量下降 50-90%）。

确保速度和双工设置正确。

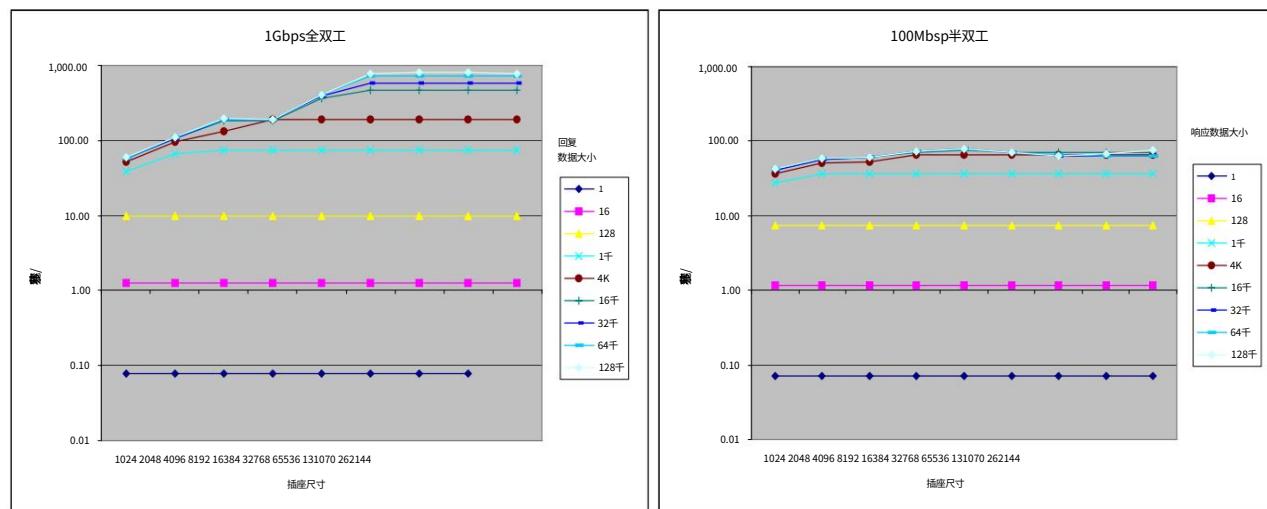


图4-15 自协商失败导致性能下降

许多网络设备默认设置为 100 Mb 半双工模式，以防自动协商过程中出现轻微不匹配。要检查网络连接的实际线路速度和双工设置，请使用 ethtool 命令。

请注意，大多数网络管理员认为将网络接口连接到网络的最佳方式是在 NIC 和交换机或集线器端口上指定静态速度。

更改配置时,如果设备驱动程序支持 ethtool 命令,则可以使用ethtool。对于某些设备驱动程序,您可能需要更改 /etc/modules.conf。

4.7.3 MTU大小

尤其是在千兆网络中,较大的最大传输单元 (MTU) 大小 (也称为 JumboFrames) 可以提供更佳的网络性能。较大的 MTU 大小的挑战在于,大多数网络不支持它们,而且许多网卡也不支持较大的 MTU 大小。如果您的目标是以千兆速度传输大量数据 (例如在 HPC 环境中),则增加默认 MTU 大小可以显著提升性能。要更改 MTU 大小,请使用 /sbin/ifconfig。

示例 4-18 使用 ifconfig 更改 MTU 大小

```
[root@linux ~]# ifconfig eth0 mtu 9000 up
```

注意:要使较大的 MTU 尺寸能够工作,网络接口卡和网络组件必须同时支持它们。

4.7.4 增加网络缓冲区

Linux 网络栈在为网络缓冲区分配内存资源时非常谨慎。在连接服务器系统的现代高速网络中,应增加这些值,以使系统能够处理更多的网络数据包。

初始整体 TCP 内存是根据系统内存自动计算的;您可以在以下位置找到实际值:

/proc/sys/net/ipv4/tcp_mem

将接收套接字内存的默认值和最大值设置为更高的值:

/proc/sys/net/core/rmem_default
/proc/sys/net/core/rmem_max

将发送套接字的默认数量和最大数量设置为更高的值:

/proc/sys/net/core/wmem_default
/proc/sys/net/core/wmem_max

将选项内存缓冲区的最大数量调整为更高的值:

/proc/sys/net/core/optmem_max

调整窗口大小

最大窗口大小可以通过上面描述的网络缓冲区大小参数进行调整。

理论上最佳的窗口大小可以通过使用BDP (带宽延迟积) 来获得。

BDP 是指在传输过程中驻留在网络上的数据总量。BDP 的计算公式很简单:

$$\text{BDP} = \text{带宽 (字节/秒)} * \text{延迟 (或往返时间) (秒)}$$

为了保持网络管道满负荷并充分利用线路,网络节点应预留缓冲区来存储与 BDP 大小相同的数据。否则,发送方必须停止发送数据并等待接收方的确认 (请参阅第 32 页的“流量控制”)。

例如,在延迟为 1 毫秒的千兆以太网 LAN 中,BDP 为:

$$125\text{M字节/秒 (1Gbit/秒)} * 1\text{毫秒} = 125\text{K字节}$$

在大多数企业发行版中，`rmem_max`和`wmem_max`的默认值约为 128 KB，这对于低延迟的通用网络环境来说可能足够了。但是，如果延迟较大，则默认大小可能太小。

再举一个例子，假设一个 Samba 文件服务器需要支持来自不同位置的 16 个并发文件传输会话，则默认配置下每个会话的套接字缓冲区大小为 8 KB。如果数据传输量较大，这个值可能会相对较小。

将所有协议的队列的最大 OS 发送缓冲区大小 (`wmem`) 和接收缓冲区大小 (`rmem`) 设置为 8 MB：

```
sysctl -w net.core.wmem_max=8388608
sysctl -w net.core.rmem_max=8388608
```

这些指定了创建每个 TCP 套接字时为其分配的内存量。

此外，您还应该使用以下命令来发送和接收缓冲区。

它们指定三个值：最小大小、初始大小和最大大小：

```
sysctl -w net.ipv4.tcp_rmem= 4096 87380 8388608
sysctl -w net.ipv4.tcp_wmem= 4096 87380 8388608
```

第三个值必须小于或等于 `wmem_max` 和 `rmem_max` 的值。不过，我们也建议在高速、高质量的网络上增加第一个值，以便 TCP 窗口的初始值足够高。

增加 `/proc/sys/net/ipv4/tcp_mem` 中的值。这三个值分别代表 TCP 内存的最小、压力和最大内存分配。

您可以使用 `tcpdump` 查看套接字缓冲区调整后的变化。如示例所示，将套接字缓冲区限制为较小大小会导致窗口大小过小，并导致频繁发送确认数据包和低效使用（示例 4-19）。相反，将套接字缓冲区设置得较大会导致窗口大小过大（示例 4-20）。

示例 4-19 小窗口大小 (`rmem、wmem=4096`)

```
[root@lnxsu5 ~]# tcpdump -ni eth1
22:00:37.221393 IP plnxsu4.34087 > plnxsu5.32837: P 18628285:18629745(1460) ack 9088 win 46
22:00:37.221396 IP plnxsu4.34087 > plnxsu5.32837: . 18629745:18631205(1460) ack 9088 win 46
22:00:37.221499 IP plnxsu5.32837 > plnxsu4.34087: . ack 18629745 win 37
22:00:37.221507 IP plnxsu4.34087 > plnxsu5.32837: P 18631205:18632665(1460) ack 9088 win 46
22:00:37.221511 IP plnxsu4.34087 > plnxsu5.32837: . 18632665:18634125(1460) ack 9088 win 46
22:00:37.221614 IP plnxsu5.32837 > plnxsu4.34087: . ack 18632665 win 37
22:00:37.221622 IP plnxsu4.34087 > plnxsu5.32837: P 18634125:18635585(1460) ack 9088 win 46
22:00:37.221625 IP plnxsu4.34087 > plnxsu5.32837: . 18635585:18637045(1460) ack 9088 win 46
22:00:37.221730 IP plnxsu5.32837 > plnxsu4.34087: . ack 18635585 win 37
22:00:37.221738 IP plnxsu4.34087 > plnxsu5.32837: P 18637045:18638505(1460) ack 9088 win 46
22:00:37.221741 IP plnxsu4.34087 > plnxsu5.32837: . 18638505:18639965(1460) ack 9088 win 46
22:00:37.221847 IP plnxsu5.32837 > plnxsu4.34087: . ack 18638505 win 37
```

示例 4-20 大窗口大小 (`rmem、wmem=524288`)

```
[root@lnxsu5 ~]# tcpdump -ni eth1
22:01:25.515545 IP plnxsu4.34088 > plnxsu5.40500: . 136675977:136677437(1460) ack 66752 win 46
22:01:25.515557 IP plnxsu4.34088 > plnxsu5.40500: . 136687657:136689117(1460) ack 66752 win 46
22:01:25.515568 IP plnxsu4.34088 > plnxsu5.40500: . 136699337:136700797(1460) ack 66752 win 46
22:01:25.515579 IP plnxsu4.34088 > plnxsu5.40500: . 136711017:136712477(1460) ack 66752 win 46
22:01:25.515592 IP plnxsu4.34088 > plnxsu5.40500: . 136722697:136724157(1460) ack 66752 win 46
22:01:25.515601 IP plnxsu4.34088 > plnxsu5.40500: . 136734377:136735837(1460) ack 66752 win 46
22:01:25.515610 IP plnxsu4.34088 > plnxsu5.40500: . 136746057:136747517(1460) ack 66752 win 46
```

```
22:01:25.515617 IP plnxsu4.34088 > plnxsu5.40500: . 136757737:136759197(1460) ack 66752 win 46
22:01:25.515707 IP plnxsu5.40500 > plnxsu4.34088: .ack 136678897 win 3061
22:01:25.515714 IP plnxsu5.40500 > plnxsu4.34088: .ack 136681817 win 3061
22:01:25.515764 IP plnxsu5.40500 > plnxsu4.34088: .ack 136684737 win 3061
22:01:25.515768 IP plnxsu5.40500 > plnxsu4.34088: .ack 136687657 win 3061
22:01:25.515774 IP plnxsu5.40500 > plnxsu4.34088: .ack 136690577 win 3061
```

套接字缓冲区大小的影响

当服务器处理大量并发大文件传输时,较小的套接字缓冲区可能会导致性能下降。如图 4-16 所示,使用较小的套接字缓冲区时,性能下降明显。即使对端有足够的大的套接字缓冲区,较低的 rmem_max 和 wmem_max 值也会限制可用的套接字缓冲区大小。这会导致窗口较小,并给大数据传输带来性能上限。虽然此图表未涵盖小数据(小于 4 KB)传输,但并未观察到明显的性能差异。

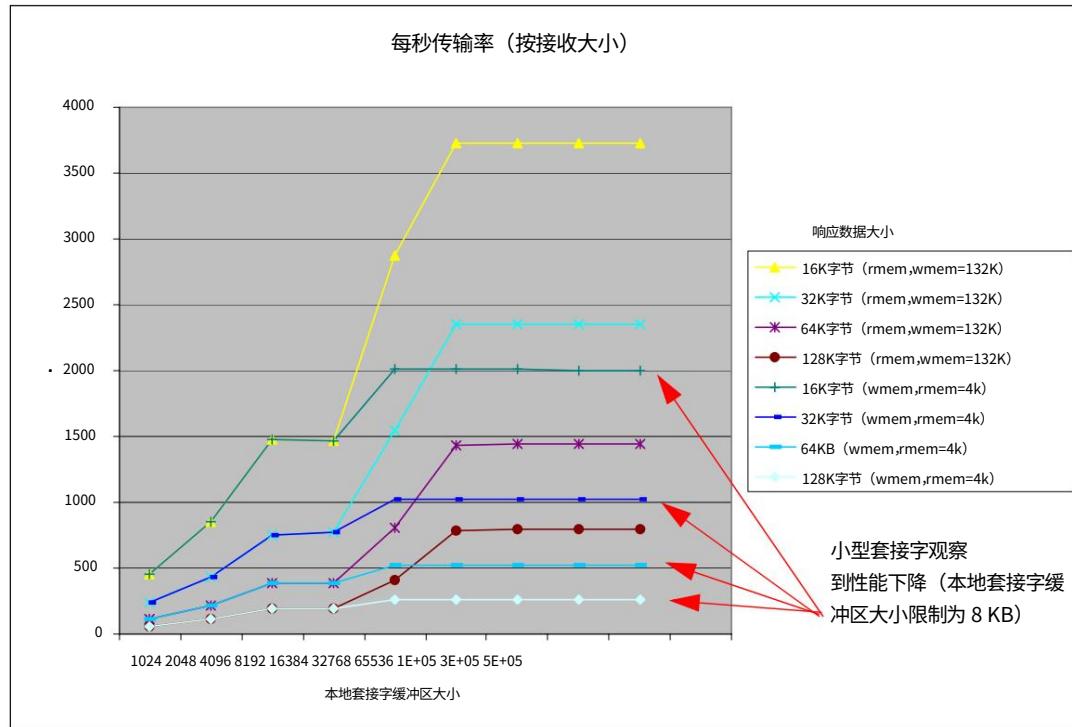


图 4-16 套接字缓冲区 4KB 与 132 字节的比较

4.7.5 额外的 TCP/IP 调整

还有许多其他配置选项可以提高或降低网络性能。下面介绍的参数可以帮助防止网络性能下降。

调整 IP 和 ICMP 行为

以下 sysctl 命令用于调整 IP 和 ICMP 行为：

禁用以下参数可防止黑客使用欺骗攻击服务器的 IP 地址：

```
sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
```

```
sysctl -w net.ipv4.conf.default.accept_source_route=0
sysctl -w net.ipv4.conf.all.accept_source_route=0
```

这些命令将服务器配置为忽略来自被列为网关的计算机的重定向。重定向可用于执行攻击,因此我们只允许来自受信任来源的重定向:

```
sysctl -w net.ipv4.conf.eth0.secure_redirects=1
sysctl -w net.ipv4.conf.lo.secure_redirects=1
sysctl -w net.ipv4.conf.default.secure_redirects=1
sysctl -w net.ipv4.conf.all.secure_redirects=1
```

您可以允许接口接受或拒绝任何 ICMP 重定向。ICMP 重定向是路由器向主机传递路由信息的一种机制。例如,当网关从其所连接网络上的某台主机收到 Internet 数据报时,它可以向该主机发送重定向消息。网关会检查路由表以获取下一个网关的地址,然后第二个网关会将 Internet 数据报路由到网络目标。使用以下命令禁用这些重定向:

```
sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0
```

如果此服务器不充当路由器,则它不必发送重定向,因此可以禁用它们:

```
sysctl -w net.ipv4.conf.eth0.send_redirects=0
sysctl -w net.ipv4.conf.lo.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
sysctl -w net.ipv4.conf.all.send_redirects=0
```

配置服务器忽略广播 ping 和 smurf 攻击:

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

忽略所有类型的 icmp 数据包或 ping:

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1
```

某些路由器会对广播帧发送无效的响应,每个路由器都会生成一条警告,并由内核记录。以下响应可以忽略:

```
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
```

我们应该设置 ipfrag 参数,特别是对于 NFS 和 Samba 服务器。在这里,我们可以设置用于重组 IP 片段的最大和最小内存。当为此目的分配了 ipfrag_high_thresh 字节的内存值时,片段处理程序将丢弃数据包,直到达到 ipfrag_low_thresh 的值。

当 TCP 数据包传输过程中出现错误时,就会发生碎片。

有效数据包存储在内存中(如这些参数所定义),而损坏的数据包则被重新传输。

例如,要将可用内存范围设置为 256 MB 至 384 MB 之间,请使用:

```
sysctl -w net.ipv4.ipfrag_low_thresh=262144
sysctl -w net.ipv4.ipfrag_high_thresh=393216
```

调整 TCP 行为

这里我们描述了将改变 TCP 行为的调整参数。

以下命令可用于调整支持大量多重连接的服务器：

对于同时接收大量连接的服务器,可以重用新连接的 TIME-WAIT 套接字。这在 Web 服务器中很有用,例如:

```
sysctl -w net.ipv4.tcp_tw_reuse=1
```

如果启用此命令,则还应启用 TIME-WAIT 套接字状态的快速回收:

```
sysctl -w net.ipv4.tcp_tw_recycle=1
```

图 4-17 显示,启用这些参数后,连接数显著减少。这有利于提高性能,因为每个 TCP 事务都会维护一个缓存,其中包含每个远程客户端的协议信息。此缓存中存储了往返时间、最大段大小和拥塞窗口等信息。更多详细信息,请参阅 RFC 1644。

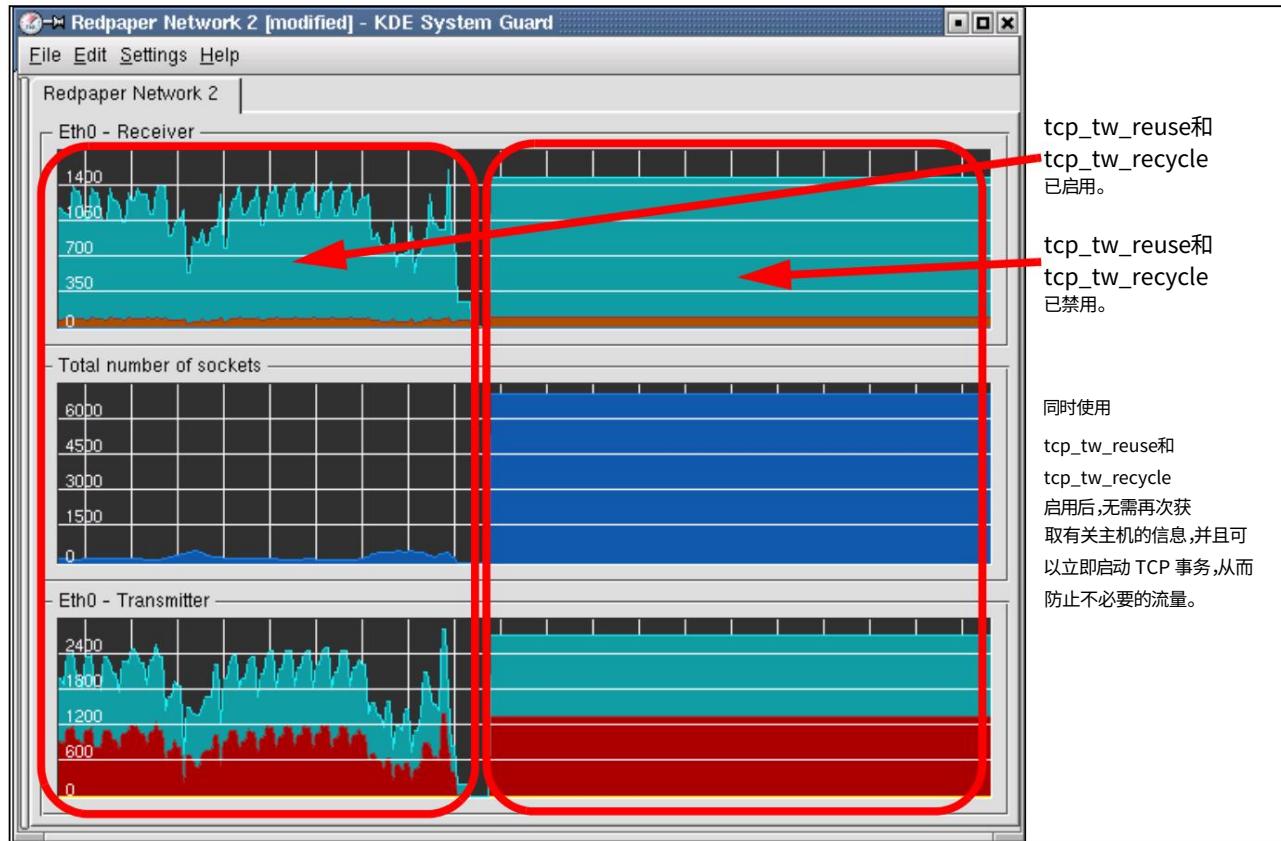


图 4-17 参数重用和回收启用 (左)和禁用 (右)

参数 `tcp_fin_timeout` 是当服务器关闭套接字时,将套接字保持在 FIN-WAIT-2 状态的时间。

TCP 连接以三段同步 SYN 序列开始,以三段 FIN 序列结束,这两个序列均不包含数据。通过更改 `tcp_fin_timeout` 值,可以缩短从 FIN 序列到释放内存以用于新连接的时间,从而提高性能。但是,此值

应在仔细监控后进行更改,因为存在因死套接字数量过多而导致内存溢出的风险:

```
sysctl -w net.ipv4.tcp_fin_timeout=30
```

在同时运行大量 TCP 连接的服务器中,一个常见的问题是存在大量打开但未使用的连接。TCP 具有 keepalive 功能,可以探测这些连接,并默认在 7200 秒 (2 小时)后丢弃它们。这个时间长度对于您的服务器来说可能过长,可能会导致内存占用过高并降低服务器性能。

例如,将其设置为 1800 秒 (30 分钟)可能更合适:

```
sysctl -w net.ipv4.tcp_keepalive_time=1800
```

当服务器负载过重或大量客户端连接不良且延迟较高时,可能会导致半开连接数增加。这对于 Web 服务器来说很常见,尤其是在拨号用户较多的情况下。这些半开连接会存储在积压连接队列中。您应该将此值至少设置为 4096。(默认值为 1024。)

即使您的服务器没有接收这种连接,设置此值也是有用的,因为它仍然可以免受 DoS (syn-flood) 攻击。

```
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
```

虽然 TCP SYN Cookie 有助于保护服务器免受 Syn-flood 攻击 (包括拒绝服务 (DoS) 和分布式拒绝服务 (DDoS)) ,但它们可能会对性能产生不利影响。我们建议仅在有明确需要时才启用 TCP SYN Cookie。

```
sysctl -w net.ipv4.tcp_syncookies=1
```

注意:该命令仅当内核使用CONFIG_SYNCOOKIES编译时才有效。

调整 TCP 选项

以下 TCP 调整选项可用于进一步调整 Linux TCP 堆栈。

选择性确认是显著优化 TCP 流量的一种方法。然而,

SACK 和 DSACK 会对千兆网络的性能产生负面影响。虽然 tcp_sack 和 tcp_dsack 默认启用,但它们会阻碍高速网络中 TCP/IP 的最佳性能,因此应将其禁用。

```
sysctl -w net.ipv4.tcp_sack=0  
sysctl -w net.ipv4.tcp_dsack=0
```

每次以太网帧转发到 Linux 内核的网络栈时,它都会收到一个时间戳。此行为对于防火墙和 Web 服务器等边缘系统来说非常有用且必要,但对于后端系统来说,禁用 TCP 时间戳可能会更有利,因为它可以减少一些开销。您可以通过以下调用禁用 TCP 时间戳:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

我们还了解到,窗口缩放可以作为扩大传输窗口的一种选择。然而,基准测试表明,窗口缩放不适用于网络负载非常高的系统。此外,某些网络设备不遵循 RFC 指南,可能会导致窗口缩放功能出现故障。我们建议禁用窗口缩放并手动设置窗口大小。

```
sysctl -w net.ipv4.tcp_window_scaling=0
```

4.7.6 Netfilter 的性能影响

由于 Netfilter 提供 TCP/IP 连接跟踪和数据包过滤功能（请参阅第 29 页的“Netfilter”），在某些情况下，它可能会对性能产生较大影响。当连接建立数量较多时，这种影响尤为明显。图 4-18 和图 4-19 分别展示了连接建立数量较多和较少时的基准测试结果。这些结果清晰地展现了 Netfilter 的效果。

当没有应用 Netfilter 规则时（图 4-18），结果显示与很少发生连接建立的基准类似的性能特征（请参阅第 124 页图 4-14 的左图），但由于连接建立开销，绝对吞吐量仍然不同。

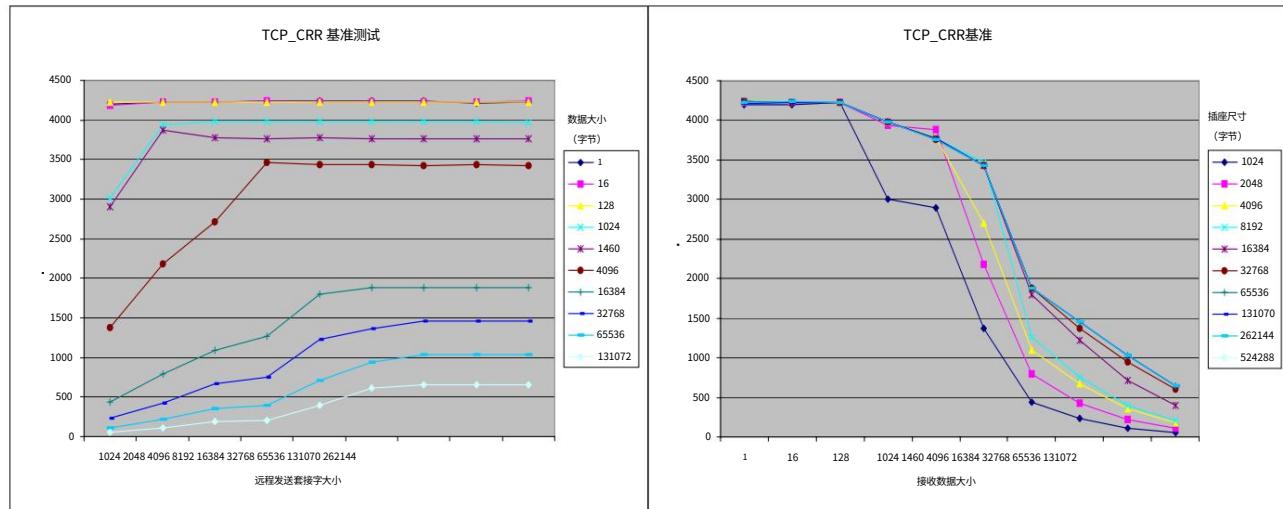


图 4-18 未应用 Netfilter 规则

然而，当应用过滤规则时，可以看到相对不一致的行为（图4-19）。

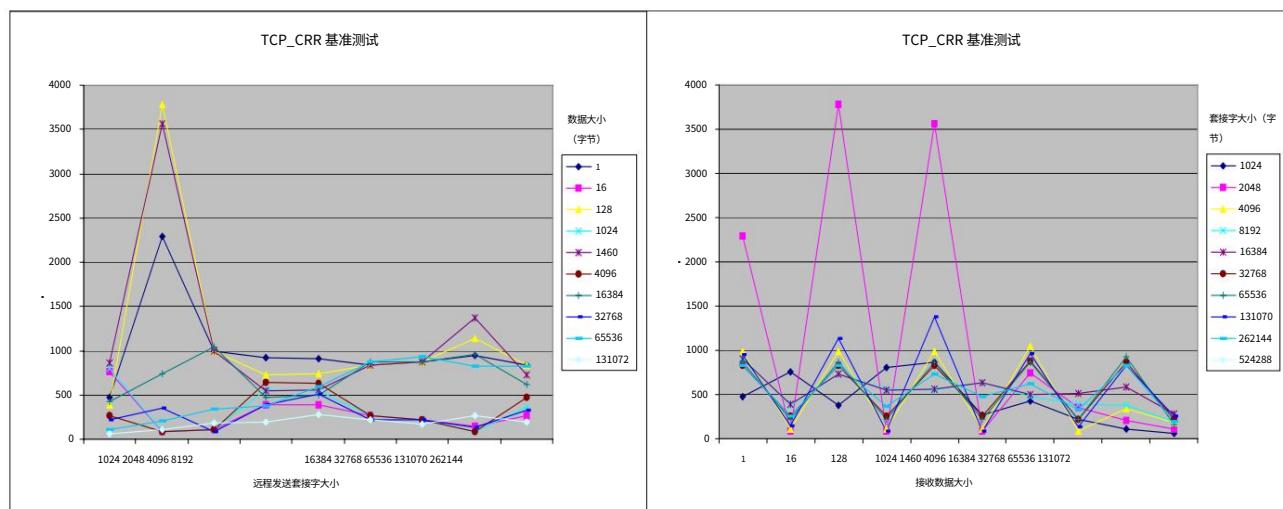


图 4-19 Netfilter 规则应用

然而,Netfilter 提供了数据包过滤功能,并增强了网络安全。这需要在安全性和性能之间做出权衡。Netfilter 的性能影响取决于以下因素:

- 规则数量
- 规则顺序
- 规则的复杂性
- 连接跟踪级别 (取决于协议)
- Netfilter内核参数配置

4.7.7 卸载配置

正如我们在 1.5.3 “卸载” (第 33 页)中所述,某些网络操作可以卸载到支持此功能的网络接口设备上。您可以使用ethtool 命令检查当前的卸载配置。

示例 4-21 检查卸载配置

```
[root@lnxsu5 plnxsu4]# ethtool -k eth0
eth0 的卸载参数:
接收校验和:关闭
tx校验和:关闭
分散-聚集:关闭
TCP分段卸载:关闭
UDP碎片卸载:关闭
通用分段卸载:关闭
```

更改配置命令语法如下:

```
ethtool -K DEVNAME [ rx 开启|关闭 ] [ tx 开启|关闭 ] [ sg 开启|关闭 ] [ tso 开启|关闭 ] [ ufo 开
启|关闭 ] [ gso 开启|关闭 ]
```

示例 4-22 卸载配置更改示例

```
[root@lnxsu5 plnxsu4]# ethtool -k eth0 sg 开启 tso 开启 gso 关闭
```

支持的卸载功能可能因网络接口设备、Linux 发行版、内核版本以及您选择的平台而异。如果您输入了不受支持的卸载参数,可能会收到错误消息。

卸载的影响

基准测试表明,通过 NIC 卸载可以降低 CPU 利用率。

图 4-20 (第 134 页)显示了大数据量 (超过 32 KB)下 CPU 利用率的提升。大数据包利用了校验和卸载功能,因为校验和计算需要计算整个数据包,因此随着数据量的增加,处理能力的消耗也会增加。

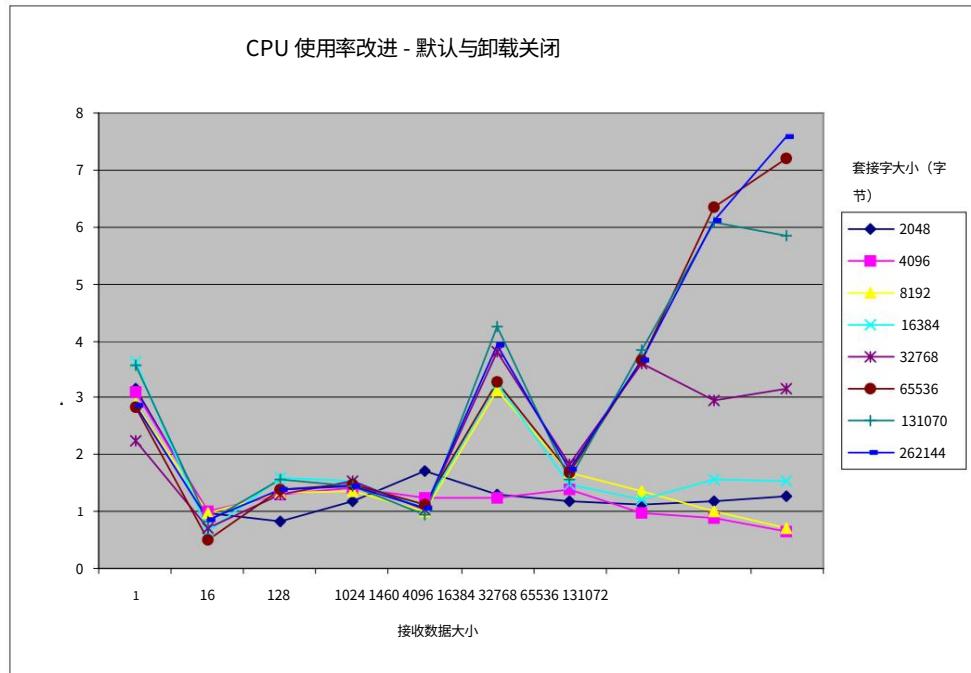


图 4-20 通过卸载改善 CPU 使用率

然而,使用卸载功能时会观察到性能略有下降 (图 4-21)。如此高的数据包速率下,校验和的处理会给某些 LAN 适配器处理器带来巨大的负担。随着数据包大小的增加,每秒生成的数据包数量会减少 (因为发送和接收所有数据需要更长的时间),因此将校验和操作卸载到适配器上是明智之举。

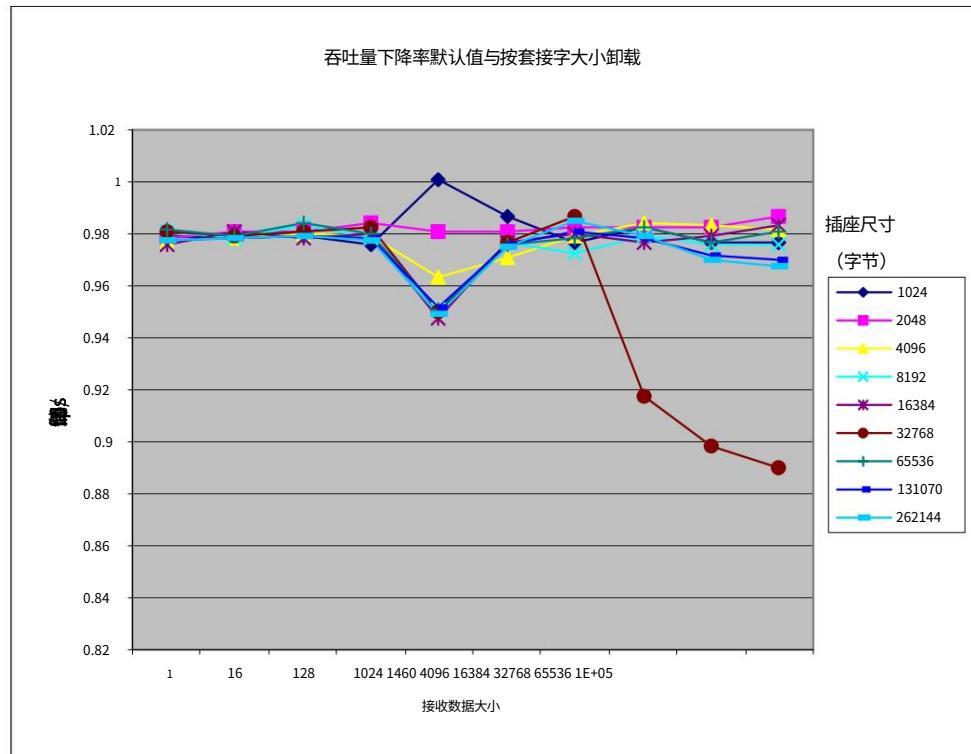


图 4-21 卸载导致的吞吐量下降

当网络应用程序请求数据时,LAN 适配器效率较高,因为这些应用程序会生成大帧请求。而当应用程序请求小块数据时,LAN 适配器通信处理器需要花费更多时间来执行传输的每个字节的开销代码。这就是为什么大多数 LAN 适配器无法在所有帧大小下都保持全线速的原因。

有关更多详细信息,请参阅《调整 IBM System x 服务器性能》(SG24-5287)第 10.3 节“高级网络功能”。

4.7.8 增加数据包队列

增加各个网络缓冲区的大小后,建议增加允许的未处理数据包数量,以便内核在丢弃数据包之前等待更长时间。为此,请编辑 /proc/sys/net/core/netdev_max_backlog 中的值。

4.7.9 增加传输队列长度

将每个接口的 txqueuelength 参数值增加到 1000 到 20000 之间。这对于执行大量同构数据传输的高速连接尤其有用。可以使用 ifconfig 命令调整传输队列长度,如示例 4-23 所示。

示例 4-23 设置传输队列长度

```
[root@linux ipv4]# ifconfig eth1 txqueuelen 2000
```

4.7.10 减少中断

处理网络数据包需要 Linux 内核处理大量的中断和上下文切换,除非使用 NAPI。对于基于 Intel e1000 的网卡,请确保网卡驱动程序已使用 CFLAGS_EXTRA -DCONFIG_E1000_NAPI 标志进行编译。Broadcom tg3 模块的最新版本应该已经内置了 NAPI 支持。

如果您需要重新编译 Intel e1000 驱动程序以启用 NAPI,您可以在构建系统上发出以下命令来实现:

使 CFLAGS_EXTRA -DCONFIG_E1000_NAPI

此外,在多处理器系统上,将网络接口卡的中断绑定到物理 CPU 可能会带来额外的性能提升。要实现此目标,首先必须通过相应的网络接口识别 IRQ。通过 ifconfig 获取的数据

命令将告知您中断号。

示例 4-24 识别中断

```
[root@linux ~]# ifconfig eth1
eth1 链路封装:以太网 HWaddr 00:11:25:3F:19:B3
      inet 地址:10.1.1.11 广播:10.255.255.255 掩码:255.255.0.0
      inet6 地址:fe80::211:25ff:fe3f:19b3/64 范围:链接
          上行广播运行多播 MTU:1500 度量:1
      RX 数据包:51704214 错误:0 丢弃:0 超限:0 帧:0
      TX 数据包:108485306 错误:0 丢弃:0 超限:0 载波:0
      碰撞:0 txqueuelen:1000
      RX 字节数:4260691222 (3.9 GiB) TX 字节数:157220928436 (146.4 GiB)
      中断:169
```

获取中断号后,您可以使用 /proc/irq/%{irq number} 中的 smp_affinity 参数将中断绑定到 CPU。示例 4-25 演示了上述输出,其中 eth1 的 169 号中断被绑定到系统中的第二个处理器。

示例 4-25 设置中断的 CPU 亲和性

```
[root@linux ~]# echo 02 > /proc/irq/169/smp_affinity
```



一个

附录 A. 测试配置

本附录列出了用于加载和测试各种调整参数、监控软件和基准运行的硬件和软件配置。

硬件和软件配置

本红皮书所进行的测试、调整修改、基准运行和监控均是在两个不同的硬件平台上安装的 Linux 上执行的：

IBM z/VM 系统上的来宾

IBM System x 服务器上原生

在客户 IBM z/VM 系统上安装 Linux

IBM z/VM V5.2.0 安装在 IBM z9 处理器的 LPAR 上。已安装的 z/VM 组件包括 tcip、dirmaint、rscs、pvm 和 vswitch。

各种 Linux 客户 VM 系统的配置如表 A-1 所示。

表 A-1 在来宾 z/VM 系统上安装的 Linux

系统名称	LNXSU1	LNXSU2	LNXRH1
Linux 发行版	SUSE Linux 企业服务器 10	SUSE Linux 企业服务器 10	红帽企业 Linux 5
安装	默认使用 sysstat 6.0.2-16.4	默认使用 sysstat 6.0.2-16.4	默认使用 sysstat 7.0.0-3.el5
记忆	512 MB	512 MB	512 MB
交换 (2105 鲨鱼 <small>DASD (数据访问控制)</small>	710 MB	710 MB	710 MB
/root (2105 鲨鱼 <small>DASD (数据访问控制)</small>	6.1 GB	6.1 GB	6.1 GB
/perf (2107 DS8000 <small>DASD (数据访问控制)</small>	ReiserFS 6.8 GB	Ext3 6.8 GB	Ext3 6.8 GB

在 IBM System x 服务器上安装 Linux

三台 IBM System x x236 服务器的配置如表 A-2 所示。

表 A-2 System x 服务器上安装的 Linux

系统名称	LNXSU3	LNXSU4	LNXSU5
Linux 发行版	SUSE Linux 企业服务器 10 (运行级别 3)	红帽企业 Linux 4 (运行级别 5)	红帽企业 Linux 5 (运行级别 5)
安装	默认使用 sysstat 6.0.2-16.4 和 powertweak	默认使用 sysstat	默认使用 sysstat
记忆	4096 MB	4096 MB	4096 MB
交换 (RAID 1, 2*74GB)	2 GB	2 GB	2 GB
/根 (RAID 1, 2*74GB) 70 GB		70 GB	70 GB

/perf (RAID 5EE, 4*74GB)	ReiserFS 200 GB	Ext3 200 GB	Ext3 200 GB
-----------------------------	-----------------	-------------	-------------

缩写和首字母缩略词

ACK确认字符	日志文件系统
ACPI高级配置和电源接口	KDE K 桌面环境
AIX 高级交互式执行	LAN局域网
API 应用程序编程接口	LDAP轻量级目录访问协议
ATA AT 附件	后进先出法 后进先出
AVC访问向量缓存	LRU最近最少使用
BDP带宽延迟积	大规模集成
BSD伯克利软件发行版	LSM Linux 安全模块
BSS块存储段	LWP轻量级进程
CEC中央电子综合体	MAC介质访问控制
CFQ完全公平排队	MTU最大传输单元
CPU中央处理器	NAPI网络 API
CSV逗号分隔值	NFS网络文件系统
CUPS通用 UNIX 打印系统	NGPT下一代 POSIX 线程
DF 决策联邦者	网卡 网络信息中心
DMA直接内存访问	NLWP轻量级进程数
DNAT动态网络地址转换	NOOP无操作
DNS域名系统	NPTL本机 POSIX 线程库
DS 目录服务	NUMA非统一内存访问
胖的 文件分配表	开放系统互连
先进先出 先进先出	个人电脑 路径控制
FQDN完全限定域名	PCI 外围组件互连
FS 光纤通道服务	PID 进程 ID
FTP 文件传输协议	POSIX可移植操作系统接口
GNU GNU 不是 Unix	计算机环境
GPL通用公共许可证	PPID父进程 ID
GRUB大统一引导程序	优先等级 一次群速率接口
图形用户界面	RAID独立磁盘冗余阵列
HBA主机总线适配器	RAM随机存取存储器
HPC高性能计算	RFC征求意见稿
HTML超文本标记语言	RPM Redhat 软件包管理器
HTTP超文本传输协议	RSS丰富的站点摘要
IBM 国际商业机器公司	SACK选择性确认
ICMP互联网控制消息协议	SATA串行 ATA
集成开发环境 集成驱动电子设备	SCSI小型计算机系统接口
互联网协议	SMP对称多处理器
IRC 跨区域通信	SMT对称多线程
中断请求	SMTP简单邮件传输协议
独立软件供应商	SUSE软件和系统介绍
ITSO国际技术支持组织	SWAT Samba Web 管理工具

SYN同步字符

TCQ标记命令队列

TFTP简单文件传输协议

网桥负载平衡 翻译后备缓冲区

TSO TCP分段卸载

终端电话 电传打字机

UDP用户数据报协议

唯一标识符 唯一标识符

向上 单处理器

USB通用串行总线

虚拟金融服务 虚拟文件系统

VM虚拟机

XFS 扩展文件系统

XML可扩展标记语言

YaST另一个安装工具

相关出版物

本节列出的出版物被认为特别适合对本文涵盖的主题进行更详细的讨论。

IBM 红皮书

有关订购这些出版物的信息,请参阅第 145 页上的“如何获取 IBM Redbooks”。请注意,这里引用的一些文档可能仅以软拷贝形式提供。

Linux 手册 IBM Linux 解决方案和资源指南,SG24-7000

调优 IBM System x 服务器性能,SG24-5287

IBM 系统存储解决方案手册,SG24-5250

IBM TotalStorage 生产力中心,用于 Linux 上的复制,SG24-7411

存储区域网络简介,SG24-5470

TCP/IP 教程和技术概述,GG24-3376

其他出版物

以下出版物也可作为进一步的信息来源:

Beck,M.,等人,《Linux Kernel Internals》第二版,Addison-Wesley Pub Co,1997 年,ISBN 0201331438

Bovet,Daniel P.,Cesati,Marco,《理解 Linux 内核》,O'Reilly Media,Inc.
2005,ISBN-10:0596005652

Kabir, M.,《Red Hat Linux 安全与优化》,John Wiley & Sons,2001 年,ISBN 0764547542

Musumeci,Gian-Paolo D.,Loukides,Mike,《系统性能调优》,第 2 版,O'Reilly Media,Inc. 2002,
ISBN-10:059600284X

Stanfield,V.,等人,《Linux 系统管理》,第二版,Sybex Books,2002 年,ISBN 0782141382

在线资源

以下网站也可作为进一步的信息来源:

高性能可扩展服务器上的 Linux 网络可扩展性

<http://www.ibm.com/servers/eserver/xseries/benchmarks/>

System z 上的 Linux 调优提示和技巧

<http://www.ibm.com/developerworks/linux/linux390/perf/index.html>

Linux 服务器的系统调优信息

http://people.redhat.com/alikins/system_tuning.html

保护和优化 Linux (Red Hat 6.2)

<http://www.faqs.org/docs/securing/index.html>
Linux 2.6 在企业数据中心的性能
http://www.osdl.org/docs/linux_2_6_datacenter_performance.pdf ReiserFS 的开发者
<http://www.namesys.com>
V2.6内核新特性
http://www.infoworld.com/infoworld/article/04/01/30/05FElinux_1.html
2.4 和 2.6 上的 WebServing
<http://www.ibm.com/developerworks/linux/library/l-web26/>
关于ab命令的手册页
<http://cmpp.linuxforum.net/cman-html/man1/ab.1.html>
Linux 2.6 中的网络性能改进
http://developer.osdl.org/shemminger/LWE2005_TCP.pdf
RADIANT 出版物和演示文稿
<http://public.lanl.gov/radiant/pubs.html>
RFC:多播
<http://www.ietf.org/rfc/rfc2365.txt>
RFC:互联网控制消息协议
<http://www.networksorcery.com/enp/RFC/Rfc792.txt>
RFC:故障隔离和恢复
<http://www.networksorcery.com/enp/RFC/Rfc816.txt>
RFC:Internet 协议套件中的服务类型
<http://www.networksorcery.com/enp/rfc/rfc1349.txt>
使用 OpenLDAP 进行性能调整
<http://www.openldap.org/faq/data/cache/190.html>
RFC:长延迟路径的 TCP 扩展
<http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1072.html>
RFC:高性能 TCP 扩展
<http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1323.html>
RFC:扩展 TCP 以进行事务处理 概念
<http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1379.html>
RFC:T/TCP 事务的 TCP 扩展
<http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc1644.html>
LOAD 负载和性能测试工具
<http://www.softwareqatest.com/qatweb1.html>
Web100项目
<http://www.web100.org/>
有关超线程的信息
<http://www.intel.com/business/bss/products/hyperthreading/server/>

有关 EM64T 的信息

<http://www.intel.com/technology/64bitextensions/>

如何获取 IBM 红皮书

您可以在此网站上搜索、查看或下载红皮书、红皮书、提示和技巧、出版物草稿和其他材料，以及订购印刷版红皮书或 CD-ROM：

ibm.com/redbooks

IBM 的帮助

IBM 支持和下载

ibm.com/support

IBM 全球服务

ibm.com/services

指数

符号

/proc

参数文件 107

数字32 位架构 10 三次握手
手 30 64 位架构 11

参见“完整公平队列”更改管理 92 更改内核参数 104-107 校验和卸载 33 子进程 3 chkconfig 命令 98 clone() 5 冲突数据包 88 编译内核 104

完全公平排队 24,115-118,121 连接建立 30 三次握手 30

一个

访问向量缓存 104

ACK数据包30

ACPI

查看高级配置和电源界面

高级配置和电源接口 59 预期 24, 116-117 apmd 97 arptables 97 autofs 97

连接跟踪 30 上下文 5

上下文切换 5

CPU 亲和性 81

CPU 瓶颈 81-82 操作 82

CPU 调度程序 9-10 cpufreq 命令

61 cpuspeed 97 cups 97

AVC

参见访问向量缓存

D守

护进程 97-100 默认 97 可调

97 数据段 8 截止

时间 24, 115-117

脏缓冲区 22, 109 刷新 22,

110 dirty_ratio 110 禁用 SELinux

103 磁盘驱动器 113 磁盘瓶颈

84-87 iostat 命令 86 解决

方案 87 vmstat 命令 85

磁盘 I/O 子系统 19-25 块层 23-

24 缓存 21-22

B带

宽延迟积126基准工具 70-76

功能概述 40

IOzone 72

LMBench 71

netperf 73-75 将进程

绑定到 CPU 81 将中断绑定到 CPU 6, 108 块

设备指标 36 块层 23-24 块大小 123 绑定驱动程序

34 绑定模块 34 分析服务器性能的瓶

颈 80

CPU 瓶颈 81-82 磁盘瓶颈 84-87 收集

信息 78 内存瓶颈 82-84 网络瓶颈 87-

89 总线子目录 62

I/O 子系统体系结构 20 磁盘子系统 112-124 添加

驱动器 87 文件系统选择 120-124 文件

系统调整 120-124 硬件注意

事项 113

碳

C09 编译器标志 104 缓存 21-22 缓存优化 81

I/O 升降机选择 115-119

I/O 升降机调优 115-119 dmesg 命令 94 丢弃

数据包 88 双工 125 动态内存分配 8

容量管理器 67-70 cat 命令 105, 107

CFQ

E

电梯

参见 I/O 升降机

企业 Linux 发行版 93 ethereal 命令 55, 57
ethtool 命令 125–126 exec() 4 exit() 4

中断处理 6 将中断绑定到 CPU
6, 108
CPU 亲和性 108 中断减少
135 ionice
121 iostat 命令 48, 86

扩展 2 17–18, 120
Ext3 18–19, 120, 122 扩展 2 文件
系统
参见 Ext2 扩展 3 文件系统
请参阅 Ext3
扩展文件系统 19, 120

IOzone 72 iptraf
54 irq 子目录
62 irqbalance 97 isdn 97

F

故障适配器 88
光纤通道控制器 85 文件系统 15–19

扩展 2 17–18
Ext3 18–19 扩展文件系统 19
日志文件系统 19 日志记录 16
ReiserFS 19 选择
120–124 调整 120–124 虚拟文件系统 15 文件系统选择 93

FIN 数据包 30 fork()
3 free 命令
46 区域中使用的内存 47

K

KDE 系统防护 62–67, 82
流程表 65
系统负载 64 内核更改参数 104–
107 编译 104 交换行为 109 查看当前配置 105
内核崩溃 87 kudzu 97

G
盖蒂 15
Gnome 系统监视器 67 gpm 97

左
LD_ASSUME_KERNEL 5 libpcap 库 55
轻量级流程 4
Linux
发行版 93 安装注意事项 92–104 性能指标 34–37

H
硬件考虑因素 113 hpoj 97
大型TLBfs 111

Linux 安全模块 104
Linux线程 4
LMbench 71 参考
位置 21
激光扫描模块
参见 Linux 安全模块
轻量级
参见 轻量级流程

I/O 升降机 20, 23–24, 113, 115–119 预期 24, 116–117

完全公平排队 24, 115–118, 121 截止时间 24, 115–117
NOOP 24, 116–117 选择 115–
119 调整 115–119
I/O子系统架构 20
IBM Director 67 init 命令 101 安装注意事项 92–
104

M最
大传输单元 33, 126–128 大小 126
内存 32 位
架构 10 64 位架构 11 内存架构 10–
15 内存瓶颈 82–84 操作 84

内存层次结构 21 内存指标 35 区域
 中使用的内存 47 mmap() 111
 监控工具 41–70

容量管理器 67–70 ethereal 命令 55, 57
 free 命令 46 功能概述 40

Gnome 系统监视器 67 iostat 命令 48 iptraf
 54

KDE System Guard 62–67 mpstat 命令
 51 netstat 53 nmon 58
 numastat 命令
 52 pmap 命令
 52 proc 文件系统 60–62 ps 命令 44
 pstree 命令 44 sar 命令 50
 strace 命令 59 tcpdump 命令 55–
 56 top 命令 41 uptime 命
 令 43 vmstat 命令 42

交通特征 124
 网络实施
 Netfilter 29–30 网络
 API 28–29 网络子系统
 网络实现
 26–30

下一代 POSIX 线程 5 nfslock 98

NGPT
 参见下一代 POSIX 线程
 nice 命令 108 nice 级别 5, 108

Nigel 的监视器 58 nmon 58
 noatime 122

非统一内存架构 9, 52, 108
 NOOP 24, 116–117
 国家运输部
 参见本机 POSIX 线程库 nr_requests 117

非易失性存储器
 参见非统一内存架构 numastat 命令 52, 109 磁盘驱动器数量
 113

哦

mount 命令 122 mpstat 命令 51
 最大传输单元
 参见最大传输单元

O(1)算法 9
 O(n) 算法 9 卸载 33 校验和
 卸载 33 配置 133–
 135 影响 133

**北
NAPI**

查看网络 API
 本机 POSIX 线程库 5 net 子目录 62
 Netfilter 29–30 连接跟
 跟踪 30 性能影响 132–133 使用注意事项
 项目 94 netfs 98 netperf 73–75, 124 netstat 53
 网络 API 28–29 网络瓶颈 87–89 网络缓
 冲区 126 套接
 字缓冲区大小 128 调整窗口大
 小 126 网络接口
 指标 36 网络子系统 26–34, 124–
 136 双工 125 最大传输单元 126–128

TCP 分段卸载 33
 P
 包选择注意事项 94 数据包队列 135 页面缓存 20 分页与交换
 的比较 83 定义 83

分区布局注意事项 93 分区设置 114–115 pcmcia 98
 pdflush 20,
 22, 109 pdflush 行为 109 pmap
 命令 8, 52

UNIX 的可移植操作系统接口 4 端口映射 98

POSIX

请参阅 UNIX 的可移植操作系统接口 powertweak 106 proc 文件系统 60–62

Netfilter 性
 能影响 132–133 网络缓冲区 126 卸载配置 133–
 135

ACPI 61 bus
 子目录 62 更改内核设置 104 irq
 子目录 62 net 子目录 62

TCP/IP 30–34
 TCP/IP 调整 128–131

scsi 子目录 62 sys 子目录 62
 tty 子目录 62

子进程 3 定义 2 描述符 3 生命周期 3 进程管理 2-10 进程内存段 8 段 8 堆栈段 8 文本段 8 进程优先级 5 数组 9 调整 108 进程状态

参见小型计算机系统接口 SCSI 总线 85 scsi 子目录 62 安全增强型Linux SELinux 94, 102-104 禁用 103 sendmail 98 顺序工作负载 85 小型计算机系统接口 24 smartd 98

SMP 81

表面贴装技术 请参阅对称多线程套接字缓冲区 26-27 套接字缓冲区大小 128 套接字接口 26 速度 125 堆栈段 8 strace 命令 59 流式传输 85 条带大小 123 条带单元大小 123 交换文件 12, 93 与交换分区相比 93 交换分区 15, 93, 110-111 多个 110 交换空间 13 交换与分页相比 83 已定义 83 对称多线程 108 与交换文件相比 93

问

服务质量 完整的公平排队 24

R

RAID 87, 113, 123
 RAID-0 85
 RAID-10 85
 RAID-5 85 原始设备 98
 read_ahead_kb 119
 红皮书网站 145 联系我们 xii 独立磁盘
 冗余阵列 查看 RAID
 ReiserFS 19, 120, 122 renice 命令 108 重新传输 33 rpc 98 运行级别更改 101 选择 94 运行级别命令 101

SYN 数据包 30 sys 子目录 62 sysctl 命令 105, 107, 128 系统调用 clone() 5 exec() 4 exit() 4 fork() 3 wait() 4

系统负载 64

T

任务可中断 6 任务运行 6 任务停止 6 任务僵尸 7 TASK_UNINTERRUPTIBLE 7 TCP 分段卸载 33 TCP/IP 30-34 绑定模块 34 连接建立 30 卸载 33 重新传输 33 流量控制 32 传输窗口 32 调整 128-131 tcpdump 127

S sa1 50 sa2 50 sar 命令 50 SCSI

tcpdump 命令 55–56 文本段 8 线程 4–5
定义 4

LD_ASSUME_KERNEL 5
轻量级流程 4
Linux线程4
原生 POSIX 线程库 5
下一代 POSIX 线程 5 top 命令 41 流量特性 124 流量
控制 32 传输窗口 32

转换后备缓冲区 111 传输队列长度 135

请参阅 TCP 分段卸载 tty 子目录 62 可调守护进程 97 调整磁盘子系统 112–
124

ICMP 128
IP 128 网络
子系统 124–136 处理器子系统 107–109

TCP 130
TCP 选项 131 虚拟内存子
系统 109–112 窗口大小 126

U
ulimit 命令 96 uptime 命令
43, 81

五

虚拟金融服务
参见虚拟文件系统
虚拟文件系统 15 虚拟内存 10 虚拟内存管理器 12 虚拟内存子
系统 109–112 虚拟终端 102 vmstat 命令
42, 51, 61, 85

W
等待 () 4

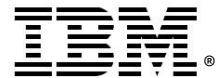
+

XFS
参见扩展文件系统 xfs 98

Z
僵尸进程 7

Machine Translated by Google

到底



Linux 性能和 调优指南



红皮书

操作系统调优方 法

IBM® 已拥抱 Linux，并认为 Linux 是适合在 IBM 系统上运行企业级应用程序的操作系统。现在，大多数企业应用程序都可以在 Linux 上运行，包括文件和打印服务器、数据库服务器、Web 服务器以及协作和邮件服务器。

性能监控工具

在企业级服务器中使用 Linux 时，需要监控性能，并在必要时调优服务器以消除影响用户的瓶颈。这份 IBM 红皮书介绍了可用于调优 Linux 的方法，可用于监控和分析服务器性能的工具，以及特定服务器应用程序的关键调优参数。本红皮书旨在帮助您理解、分析和调优 Linux 操作系统，以便为您计划在这些系统上运行的任何类型的应用程序提供卓越的性能。

性能分析

我们测试环境中使用的调优参数、基准测试结果和监控工具均在 IBM System x 服务器和 IBM System z 服务器上运行的 Red Hat 和 Novell SUSE Linux 内核 2.6 系统上执行。不过，本红皮书的信息应该适用于所有 Linux 硬件平台。

国际的
技术的
支持
组织

建筑技术
信息基于
实践经验

IBM 红皮书由 IBM 国际技术支持组织开发。来自世界各地的 IBM 专家、客户和合作伙伴根据实际场景创建及时的技术信息。

提供具体建议以帮助您在您的环境中更有效地实施 IT 解决方案。

更多信息：
ibm.com/redbooks