红帽企业版 **Linux 7**
性能调节指南

在红帽企业版 Linux 7 中优化子系统吞吐量

作者：罗拉·贝莉 – Laura Bailey
翻译、校对：付莹莹 – Yingying (Angelina) Fu
翻译、校对：龚心星 – Xinxing Gong
校对、编辑：任浩 – Hao (Naomi) Ren
校对、责任编辑：鄭中 – Chester Cheng

红帽企业版 Linux 7 性能调节指南

在红帽企业版 Linux 7 中优化子系统吞吐量

作者：罗拉・贝莉 – Laura Bailey
红帽 工程部出版中心

翻译、校对：付莹莹 – Yingying (Angelina) Fu
澳大利亚昆士兰大学 笔译暨口译研究所
AngelinaBlack@163.com

翻译、校对：龚心星 – Xinxing Gong
澳大利亚昆士兰大学 笔译暨口译研究所
xx.gong@hotmail.com

校对、编辑：任浩 – Hao (Naomi) Ren
澳大利亚昆士兰大学 笔译暨口译研究所
renhao0823@gmail.com

校对、责任编辑：鄭中 – Chester Cheng
红帽工程部翻译中心 & 澳大利亚昆士兰大学笔译暨口译研究所
ccheng@redhat.com, uqcchun1@uq.edu.au

法律通告

摘要

红帽企业版 Linux 7 《性能调节指南》 阐述了优化红帽企业版 Linux 7 性能的方法，收录了红帽企业版 Linux 7 的性能升级。 《 性能调节指南 》仅陈述现场试验过和经过证实的程序。尽管如此，在所有可能的配置应用到产品系统之前，都应在测试环境下进行设置和测试，也建议在调节之前备份所有的数据和配置设定。

目录

第 1 章 红帽企业版 Linux 7 的性能特性

红帽企业版 Linux 7 中与性能有关的变更的简要概述，请阅读本章节。

1.1. 7.0版本中的新变化

- 此指南是为红帽企业版 Linux 7 而完全重新撰写和架构的。
- 红帽企业版 Linux 7 中 **deadline** 作为默认的 I/O 调度器替代了 **cfq**。这个变更为大多数的用例提供更好的性能表现。
- XFS 文件系统替代 ext4 成为默认的文件系统，并且现在支持最大容量可达 500 TB 的文件系统，及最大容量可达 8 EB（稀疏文件）的文件偏移。为协助清晰度，更新了 XFS 的调整推荐。
- Ext4 文件系统现在支持最大容量为 50 TB 的文件系统和最大可达 16 TB 的文件。调整推荐也做了相应的更新。此外，ext4 的驱动提供对 ext2 和 ext3 文件系统的支持。
- 现在提供作为技术预览的 Btrfs 文件系统。
- 红帽企业版 Linux 7 包括一些为 GFS2 的细微的性能提升。
- 为了提供对配置文件和添加/保存 **tuned** 配置文件的支持，更新了 **Tuna**。为了消耗更少的处理器资源，这个更新版使用基于事件的采样。为了允许实时监控，图形化版本也同样进行了更新。**Tuna**的文档参见：〈第 2.4 节 “Tuna”〉，〈第 3.3.8 节 “使用 Tuna 配置 CPU、线程和中断关联”〉和〈第 A.2 节 “Tuna”〉。
- **tuned** 默认配置文件更新为 **throughput-performance**。它替代了现在被移除的 **enterprise-storage** 配置文件。为了网络化和虚拟化，添加了一些新的配置文件。此外，**tuned** 现在提供外壳脚本标注和 ***includes*** 功能。
- **tuned-adm** 工具现在提供 **recommend** 子命令，它为您的系统推荐适当的调整配置文件。它同时为您的系统在安装时设置默认的配置文件，因此可用于恢复默认配置。
- 红帽企业版 Linux 7 提供对自动 NUMA 平衡的支持。系统内核自动探测主动使用的内存页进程线程，并且将这些线程和它们的内存进行 NUMA 节点内或跨节点的分组。系统内核重新调度线程并迁移内存，从而为最佳 NUMA 对齐方式和性能平衡系统。
- 启用文件系统 barrier（屏障）的性能惩罚现在是可忽略的（少于3%）。因此，**tuned** 配置文件不会禁用文件系统 barrier。
- 使用新的 **operf** 工具，OProfile 增加了对基于 Linux 性能事件子系统配置文件的支持。这个新工具能替代 **opcontrol** 后台程序用于收集数据。
- 控制组作为一种分配资源到您系统中的某些进程组的方法仍然可用。红帽企业版 Linux 7 的实现的具体信息请参见《*红帽企业版 Linux 7 资源管理指南*》，可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

第 2 章 性能监控工具

本章简要说明了一些红帽企业版 Linux 7 可用的性能监控和配置工具。在可能的情况下，本章会进一步指导用户使用该工具并举例说明可以使用该工具来解决的实际问题。

以下知识库文章提供适用于红帽企业版 Linux 的更为全面的性能监控工具列表：https://access.redhat.com/site/solutions/173863。

2.1. /proc

/proc "文件系统"是一个目录，其中包含的文件层次结构代表了 Linux 内核的当前状态。它允许用户和管理员查看系统的内核视图。

/proc 目录中还包含关于系统硬件及任何当前正在运行的程序信息。大部分在 **/proc** 中的文件是只读文件，但一些文件（主要是 /proc/sys 文件）能够被用户和应用程序操作，以便向内核传达配置的修改信息。

关于查看及编辑 **/proc** 目录中文件的更多信息，请参见红帽企业版 Linux 7 系统管理员参考指南，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获得。

2.2. GNOME 系统监控器

GNOME 桌面系统包含一个图形工具，即系统监控器来协助用户监控及修改系统性能。系统监控器显示基本的系统信息并能让用户监控系统进程，以及资源或文件系统的用量。

系统监控器有四个选项卡，每个都显示不同的系统信息。

系统

此选项卡显示关于系统硬件和软件的基本信息。

进程

此选项卡显示关于活动进程和进程间相互关系的具体信息。显示的进程可进行筛选，以便更容易找到某些特定的进程。

资源

此选项卡显示目前 CPU 的时间使用量，内存和交换空间用量以及网络使用情况。

文件系统

此选项卡列举了所有安装的文件系统，并提供每个的基本信息，例如文件系统类型、安装点和内存使用情况。

要启动系统监控器，按超级键进入活动概览，输入 "*System Monitor*" ，然后按回车键。

关于系统监控器的更多信息，参见应用程序中的帮助菜单，或红帽企业版 Linux 7 《*System Administrator's Guide*》，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获得。

2.3. PCP

红帽企业版 Linux 7 引入了对 PCP（性能协驾）的支持，PCP 是一套工具、服务及程序库，用于获取、存储及分析系统级的性能测量值。其轻量级分布式体系结构尤其适合复杂系统的集中化分析。可以使用 Python，Perl，C++ 和 C 接口来添加性能指标。分析工具可以直接使用客户 API （应用编程接口）（Python, C++, C），而且丰富的网页应用程序可以通过一个 JSON 接口来搜索所有可用的性能数据。

pcp 数据包提供命令行工具及底层功能。图形工具也需要 *pcp-gui* 数据包。

关于 PCP，详情请见 第 A.8 节 “PCP（性能协驾）”。此外，*pcp-doc* 数据包提供全面的文档，默认安装于 **/usr/share/doc/pcp-doc**。PCP 也提供每个工具的手册页，在命令行输入 **man *toolname*** 来查看该工具的手册页。

2.4. Tuna

Tuna 调试配置细节，如调度器策略、线程优先级以及 CPU 和中断关联。*tuna* 数据包提供了命令行工具和有同样功能的图形界面。

第 3.3.8 节 “使用 Tuna 配置 CPU、线程和中断关联” 说明了使用 Tuna 在命令行配置系统的方法。关于使用 Tuna 的方法，请见 第 A.2 节 “Tuna” 或手册页：

```
$ man tuna
```

2.5. 内置命令行工具

红帽企业版 Linux 7 提供大量工具来从命令行监控用户系统，使用户可在运行级 5 外监控系统。本章会简要论述每个工具，并提供更多关于在何处使用及如何使用这些工具的链接。

2.5.1. top

top 工具由 *procps-ng* 数据包提供，提供运行系统中进程的动态视图。它能显示多种信息，包括系统摘要和当前由 Linux 内核管理的任务列表。它也具有有限的控制流程的能力，并且在系统重启时永久保存配置更改。

默认情况下，显示的进程会根据 CPU 使用率比例来调整，以便容易地看出消耗最多资源的进程。显示的信息及操作都是高度可配置型的，这使用户可以专注于所需的不同用量的统计数据。

关于使用 top 工具的更多信息，请见手册页：

```
$ man top
```

2.5.2. ps

ps 工具由 *procps-ng* 数据包提供，提供选定的一组运行进程快照。默认情况下，检测过的组别受限于当前用户所占有的进程，并与运行 ps 的终端相关联。

较之于 top 工具，ps 可以提供更多关于进程的信息，但默认情况下提供的是这一数据的单一快照，并通过进程标识符来调整。

关于使用 ps 工具的更多信息，请见手册页：

```
$ man ps
```

2.5.3. vmstst

虚拟内存统计数据工具，即 vmstat，提供关于系统进程、内存、分页、输入/输出、中断和 CPU 活动的即时报告。vmstat 让用户设置一组采样间隔以接近实时地观察系统活动。

vmstat 由 *procps-ng* 数据包提供。关于使用 vmstat 的更多信息，请见手册页：

```
$ man vmstat
```

2.5.4. sar

系统活动报告，即 sar，收集及报告系统当天到目前为止发生的活动信息。默认的输出以十分钟为间隔，显示当天 CPU 的使用情况（自系统时间00:00:00 始）。

用户也可以使用 **-i** 选项来以秒为单位设置间隔时间，例如，**sar -i 60** 意味着 sar 每一分钟检查一次 CPU 使用率。

sar 是通过 top 工具来手动创建系统活动定期报告的一种有效方法。由 *sysstat* 数据包提供。关于使用 sar 的更多信息，请见手册页：

```
$ man sar
```

2.6. tuned 和 tuned-adm

tuned 是一种后台调节程序，可通过设置调节配置文件使操作系统在特定工作负载下发挥更好的性能。tuned-adm 是一个命令行工具，允许用户在不同调节配置文件中进行切换。

常见用例包含一些预定义文件，但是 tuned-adm 允许用户自定义配置文件，既可以是预定义文件中的一个，也可以从头定义。在红帽企业版 Linux 7 中，默认文件是 **throughput-performance**。

tuned-adm 提供的文件分为两类：节能文件和性能提升文件。性能提升文件的内容如下，其侧重点分别为：

- 存储和网络的低延迟
- 存储和网络的高吞吐量
- 虚拟计算机性能
- 虚拟主机性能

启用 tuned 的方法，请见 第 A.5 节 “tuned”。

tuned-adm 提供的性能提升文件，请见 第 A.6 节 “tuned-adm”。

tuned-adm 提供的节能文件，请见红帽企业版 Linux 7 电源管理指南，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

关于 tuned 和 tuned-adm 的使用详情，请参见其各自的手册页：

```
$ man tuned
```

```
$ man tuned-adm
```

2.7. perf

perf 工具通过使用硬件性能计数器和内核跟踪程序来追踪其他命令和应用程序对系统的影响。不同 perf 子命令显示并记录常见性能活动的统计数据，并对数据进行分析和报告。

关于 perf 及其子命令，详情请见 第 A.7 节 “perf”。

此外，欲了解更多信息，请参见红帽企业版 Linux 7 《*Developer Guide*》，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

2.8. turbostat

turbostat 由 *kernel-tools* 数据包提供。是对 Intel® 64 位处理器中处理器的拓扑、频率、空闲的电源状态统计、温度和用电量的报告。

turbostat 有助于识别服务器的用电量或空闲时间低效问题，也有助于识别系统的管理中断率 （SMI），并能验证电源管理调节的效果。

turbostat 需要 root 特权来运行，也需要处理器支持以下需求：

- 不变的时间戳计数器
- APERF 特定模型寄存器
- MPERF 模型特定寄存器

turbostat 的输出及读取方法，详情请见 第 A.11 节 “turbostat”。

关于 turbostat 的更多信息，请见手册页：

```
$ man turbostat
```

2.9. iostat

iostat 工具由 *sysstat* 数据包提供。它对系统输入/输出设备负载进行监控和报告，以协助管理员就如何在物理磁盘间输入/输出负载做出决定。它是对处理器或设备自 iostat 上次运行或启动的使用率的报告。用户可以通过使用在 iostat 手册页中定义的参数来在特定设备上关注这些报告的输出：

```
$ man iostat
```

2.10. irqbalance

irqbalance 是一个通过分配处理器硬件中断以提高系统性能的命令行工具。**irqbalance** 详情，请见 第 A.1 节 “irqbalance（中断平衡）” 或手册页：

```
$ man irqbalance
```

2.11. ss

ss 是一个命令行实用程序，显示 socket 的统计信息，使管理员能够根据时间来评估设备性能。ss 默认打开已建立连接但未在列表内的 TCP socket，也提供很多有用选项来协助管理员筛选出特定 socket 的统计数据。

红帽推荐在红帽企业版 Linux 7 中使用 ss 代替 netstat。

一个常见的用法是 **ss -tmpie**，它显示（包括内核信息在内）关于 TCP socket、内存使用率和使用 socket 进程的详细信息。

ss 由 *iproute* 数据包提供。更多信息请见手册页：

```
$ man ss
```

2.12. numastat

numastat 工具以每一个 NUMA 节点为基础了，显示处理器和操作系统的内存统计数据。

默认情况下，**numastat** 显示每个节点的 NUMA 从内核的内存分配器中缺失的系统统计数据。最佳性能表现为高 **numa_hit** 值和低 **numa_miss** 值。**Numastat** 也提供很多命令行选项来说明系统和进程的内存如何通过系统中 NUMA 节点进行分配。

交叉引用每个节点的 **numastat** 输出和每个 CPU 的 **top** 输出是很有用的，可以用来验证进程线程是在内存分配的同一节点上运行。

Numastat 由 *numactl* 数据包提供。关于使用 numastat 的方法，详情请见 第 A.12 节 “numastat”。关于 numastat 的更多信息，请见手册页：

```
$ man numastat
```

2.13. numad

numad 是一个自动的 NUMA 关联管理后台程序。它监控系统的 NUMA 拓扑和资源使用率，目的是为了动态地提高 NUMA 的资源分配和管理（从而提高系统性能）。numad 可以根据系统的工作负载来为性能基准提供高达 50% 的改善。它还提供预先安置咨询服务，可以通过多种工作关联系统进行查询，以此为最初绑定的 CPU 和内存资源的进程提供协助。

numad 以一个节点为基础，通过定期访问 **/proc** 文件系统中的信息来监控可用的系统资源。它能尝试将资源使用率保持在一个特定的等级，并在必要时通过移动 NUMA 节点间的进程来调整资源分配。numad 可以试图通过本地化和隔离系统中 NUMA 节点子集的重要进程来获得 NUMA 的最佳性能。

numad 主要对长时间运行程序的系统有用，这些程序消耗大量的资源并包含在总系统资源中的一个子集里。它也对消耗多个 NUMA 节点资源值的应用程序有用，但当系统资源消耗比例增加时，numad 的优势会减小。

当程序只运行几分钟或不消耗太多资源时，numad 不大可能会提高性能。拥有连续且不可预测的内存访问模式的系统，如大内存数据库，也不大可能受益于 numad。

关于使用 numad 的更多信息，请见 第 3.3.5 节 “使用 numad 进行自动化 NUMA 关联管理” 或 第 A.14 节 “numad” 或参见手册页：

```
$ man numad
```

2.14. SystemTap

SystemTap 是一个跟踪和探测工具，使用户可以详尽地监控和分析操作系统的活动，尤其是内核活动。它提供的信息类似于 top、ps、netstat 和 iostat 工具的输出，但包括筛选和分析所收集数据的额外选项。

SystemTap 提供对系统活动和应用程序性能更为深入且更为准确的分析，使用户能够精确地找到系统和应用程序的瓶颈。

关于 SystemTap 的更多信息，请见红帽企业版 Linux 7 《*SystemTap Beginner's Guide*》 和红帽企业版 Linux 7 《*SystemTap TapSet Reference*》。两本书都可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

2.15. OProfile

OProfile 是一个全系统性能监控工具。它使用处理器的专业性能监控硬件来检索关于内核和系统可执行文件的信息，以便确定某些事件的频率，比如引用内存时，就接收第二级缓存请求数和硬件请求数。OProfile 也可用于确定处理器的使用率，并确定最常使用的应用程序和服务。

但 OProfile 有一些局限性：

- 性能监控样本可能不太准确。由于处理器可能无法执行指令，样本就会记录一个附近的指令，而不是记录触发中断的指令。
- OProfile 希望程序多次启动和停止。这样的话多次运行得到的样本就可以累积，用户可能需要清除上一次运行的样本数据。
- OProfile 集中识别 CPU 访问限制的进程问题。因此，对于等待锁定其他事件来识别睡眠状态的进程并无帮助。

关于 OProfile 的更多信息，请见 第 A.15 节 “OProfile” 或 *Red Hat Enterprise Linux 7 System Administrator's Guide*，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。此外还可参见系统中位于 **/usr/share/doc/oprofile-*version*** 的文档。

2.16. Valgrind

Valgrind 提供大量的检测及分析工具以帮助提高应用程序的性能。这些工具能检测内存和与线程相关的错误，同样也能检测堆、栈和数组的超限运行，使用户能够轻松地定位和修改应用程序代码错误。它也可以配置缓存、堆和分支预测来确定能增加应用程序速度并最小化内存使用率的因素。

Valgrind 通过在合成的 CPU 上运行应用程序来进行分析，并在执行时对现有的应用程序代码进行测试。之后显示的评论会清楚地标识出每个参与应用程序执行的进程至用户指定文件、文件描述符或网络 socket。要注意的是执行测试代码会花上正常执行情况下四至五十倍的时间。

Valgrind 可以按原样应用于应用程序，不需要进行重新编制。但由于 Valgrind 使用调试信息来定位编码中的问题，若应用程序和支持库没有启用调试信息编制，红帽建议进行程序编制以便能包含该信息。

Valgrind 也与 GNU 项目调试器 （gdb）一起使用来提高调试效率。

Valgrind 及其附属工具有助于内存分析。关于使用 Valgrind 分析系统内存，详情请见 第 4.2.2 节 “用 Valgrind 分析应用程序的内存使用量”。

关于 Valgrind，详情请见红帽企业版 Linux 7 开发者指南，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

关于使用 Valgrind，详情请见手册页：

```
$ man valgrind
```

当 valgrind 数据包安装后，附带文档可以在 **/usr/share/doc/valgrind-*version*** 中找到。

第 3 章 CPU

本章对红帽企业版 Linux 7 中会影响应用性能的 CPU（中央处理器）硬件细节及配置选择进行了概述。＜第 3.1 节 “注意事项” ＞详述了与 CPU 相关的会影响性能的因素。 ＜第 3.2 节 “监控和诊断性能问题” ＞教您如何使用红帽企业版 Linux 7 的工具来诊断与 CPU 硬件或配置细节相关的性能问题。 ＜ 第 3.3 节 “配置建议” ＞ 详述了可用以解决在红帽企业版 Linux 7 中与 CPU 相关的性能问题的工具和策略。

3.1. 注意事项

阅读本章来了解以下因素是如何影响系统和应用程序性能的。

- 处理器如何互相连接，并且如何连接到相关资源，如内存。
- 处理器如何为执行操作调度线程。
- 处理器如何处理红帽企业版 Linux 7 中的间断。

3.1.1. 系统拓扑

在现代计算机技术中，一个 " *中央* " 处理单元的观念是误导性的，因为大部分现代化的系统都有多个处理器。这些处理器是如何相互连接，并且如何连接至其他系统资源 — " *系统拓扑* " — 会对系统和应用程序的性能以及系统调节选项产生巨大的影响。

现代计算机技术主要运用两种主要的拓扑类型

SMP 拓扑

SMP（对称多处理器）拓扑允许所有的处理器同时访问内存。然而，由于内存访问权限的共享性和平等性，固然会迫使所有 CPU 及 SMP 系统序列化的内存访问权限的局限性增加，目前这种情况常不被接受。因此，几乎所有现代服务器系统都是 NUMA（非一致性内存访问）机器。

NUMA 拓扑

比起 SMP 拓扑，NUMA（非一致性内存访问）拓扑是近来才开发的。在 NUMA 系统中，多个处理器物理分组至一个 socket。每个 socket 都有一个专用内存区，对该内存进行本地访问的服务器统称为一个节点。

同一个节点上的服务器能高速访问该节点的存储体，但访问其他节点上的存储体速度就较慢。因此，访问非本地存储体会造成性能的损失。

考虑到性能损失，服务器执行应用程序时，NUMA 拓扑结构系统中对性能敏感的应用程序应访问同一节点的内存，并且应尽可能地避免访问任何远程内存。

因此，在调节 NUMA 拓扑结构系统中的应用程序性能时，重要的是要考虑这一应用程序的执行点以及最靠近此执行点的存储体。

在 NUMA 拓扑结构系统中，**/sys** 文件系统包含处理器、内存及外围设备的连接信息。**/sys/devices/system/cpu** 目录包含处理器在系统中相互连接的详情。**/sys/devices/system/node** 目录包含系统中 NUMA 的节点信息以及节点间的相对距离。

3.1.1.1. 确定系统拓扑结构

很多指令能帮助用户了解系统的拓扑结构。**numactl --hardware** 指令概述了系统的拓扑结构。

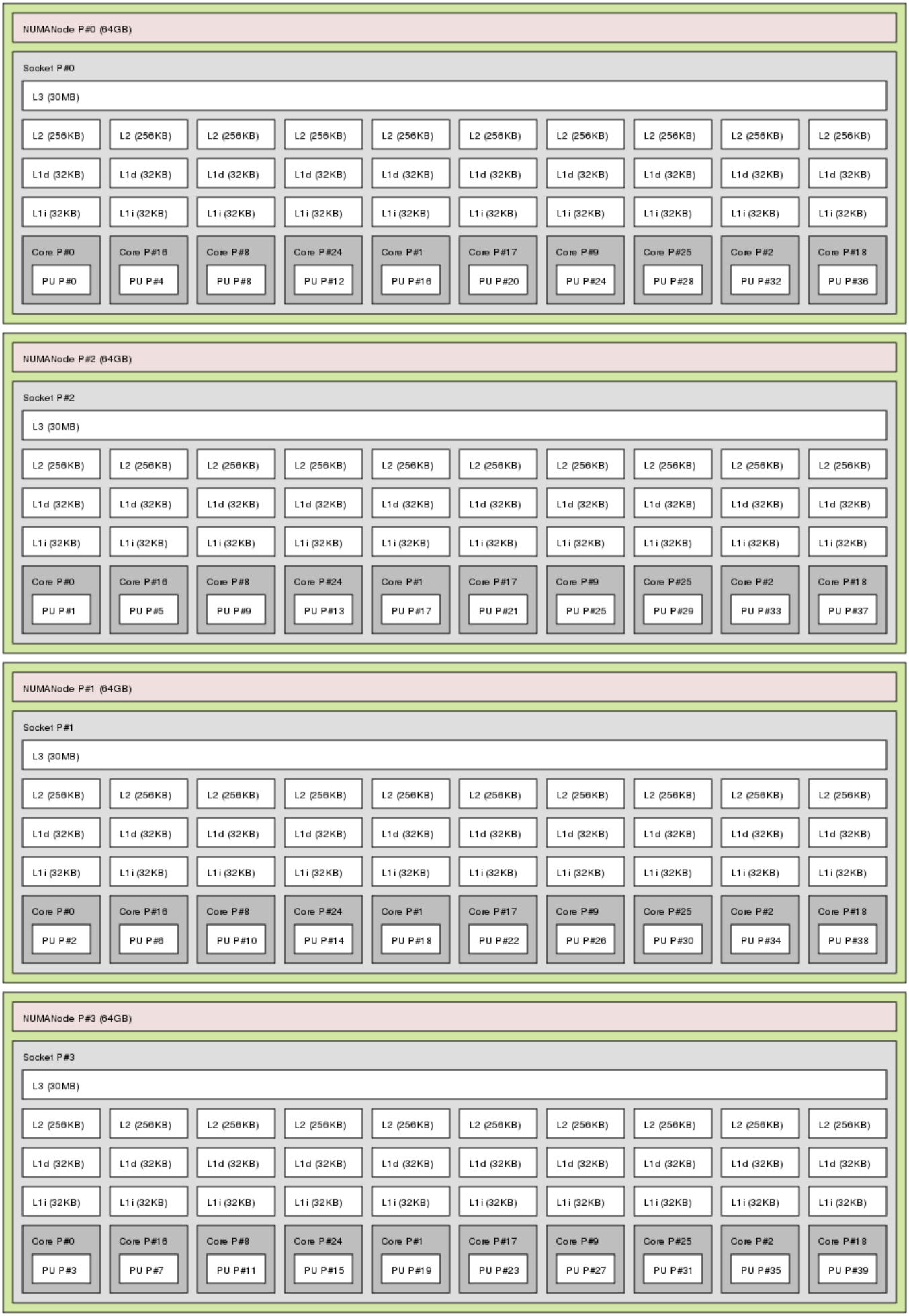
```
$ numactl --hardware
available: 4 nodes (0-3)
```

```
node 0 cpus: 0 4 8 12 16 20 24 28 32 36
node 0 size: 65415 MB
node 0 free: 43971 MB
node 1 cpus: 2 6 10 14 18 22 26 30 34 38
node 1 size: 65536 MB
node 1 free: 44321 MB
node 2 cpus: 1 5 9 13 17 21 25 29 33 37
node 2 size: 65536 MB
node 2 free: 44304 MB
node 3 cpus: 3 7 11 15 19 23 27 31 35 39
node 3 size: 65536 MB
node 3 free: 44329 MB
node distances:
node   0   1   2   3
  0:  10  21  21  21
  1:  21  10  21  21
  2:  21  21  10  21
  3:  21  21  21  10
```

lscpu 指令由 *util-linux* 数据包提供，包括 CPU 体系结构信息，如 CPU 数量、线程数、内核数、 socket 数量以及 NUMA 节点数等。

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                40
On-line CPU(s) list:   0-39
Thread(s) per core:    1
Core(s) per socket:    10
Socket(s):             4
NUMA node(s):          4
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 47
Model name:            Intel(R) Xeon(R) CPU E7- 4870  @ 2.40GHz
Stepping:              2
CPU MHz:               2394.204
BogoMIPS:              4787.85
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0,4,8,12,16,20,24,28,32,36
NUMA node1 CPU(s):     2,6,10,14,18,22,26,30,34,38
NUMA node2 CPU(s):     1,5,9,13,17,21,25,29,33,37
NUMA node3 CPU(s):     3,7,11,15,19,23,27,31,35,39
```

lstopo 指令由 *hwloc* 数据包提供，创建了用户的系统示意图。**lstopo-no-graphics** 指令提供详尽的文本输出。


NUMANode P#0 (64GB)
Socket P#0
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#0
Core P#16 PU P#4
Core P#8 PU P#8
Core P#24 PU P#12
Core P#1 PU P#16
Core P#17 PU P#20
Core P#9 PU P#24
Core P#25 PU P#28
Core P#2 PU P#32
Core P#18 PU P#36
NUMANode P#2 (64GB)
Socket P#2
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#1
Core P#16 PU P#5
Core P#8 PU P#9
Core P#24 PU P#13
Core P#1 PU P#17
Core P#17 PU P#21
Core P#9 PU P#25
Core P#25 PU P#29
Core P#2 PU P#33
Core P#18 PU P#37
NUMANode P#1 (64GB)
Socket P#1
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#2
Core P#16 PU P#6
Core P#8 PU P#10
Core P#24 PU P#14
Core P#1 PU P#18
Core P#17 PU P#22
Core P#9 PU P#26
Core P#25 PU P#30
Core P#2 PU P#34
Core P#18 PU P#38
NUMANode P#3 (64GB)
Socket P#3
L3 (30MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 PU P#3
Core P#16 PU P#7
Core P#8 PU P#11
Core P#24 PU P#15
Core P#1 PU P#19
Core P#17 PU P#23
Core P#9 PU P#27
Core P#25 PU P#31
Core P#2 PU P#35
Core P#18 PU P#39

lstpo 指令的输出

3.1.2. 调度

在红帽企业版 Linux 中，执行进程的最小单元叫做一个 “*线程*”。系统调度器决定运行线程的处理器和运行的时间。但由于调度器主要关注的是保持系统繁忙，因此可能不会为应用程序的性能而对线程进行最佳调度。

例如，在 NUMA 系统中，一个处理器在节点 B 可用，一个应用程序在节点 A 运行，要使在节点 B 的处理器保持忙碌， 调度器会把应用程序的一个线程转移到节点 B。但是，线程上的应用程序仍然需要访问在节点 A 的内存。由于该线程目前在节点 B 运行，并且对于此线程来说节点 A 的内存已不再是本地内存，访问起来就要花更长的时间。较于在节点 A 等待可用的处理器，并且在能够进行本地内存访问的源节点上执行线程，此线程在节点 B 结束运行可能就更加费时。

设计器或管理员确定线程的运行位置能使对性能敏感的应用程序从中受益。如何保证适当地调度线程，以满足对性能敏感的应用程序的需要，详情请参见 < 第 3.3.6 节 “调节调度策略” >。

3.1.2.1. 内核滴答信号

在早期红帽企业版 Linux 版本中，Linux 内核会定期中断每个 CPU 以查看需要完成的任务。查看的结果用来决定进程调度及负载均衡。这种常规性的中断叫做一个内核 “*滴答信号*”。

此标记的出现不考虑内核是否有任务要执行。这意味着为了回应这些中断，即使是空闲的内核也会被迫定期进入高能状态（每秒高达1000次）。这阻止了系统有效地利用新近 x 86 代处理器的深睡眠状态。

在红帽企业版 Linux 6 和 7 中，默认情况下内核不再中断趋于低功率状态的空闲 CPU，这种性能叫做无时钟内核。当一个或几个任务在运行时，按需中断取代了定时中断，使 CPU 可以更久地处于空闲或低功率状态，减少了电量的消耗。

红帽企业版 Linux 7 提供一种动态的无时钟设置（ **`nohz_full`** ），通过用户空间的任务来减少内核干扰以进一步改善其确定性。这一设置可以在指定的内核中通过 ***`nohz_full`*** 内核参数来启用。当这一设置在一个内核中启用时，所有的计时活动将会被移动至无延迟敏感性的内核。这对于高性能计算和实时计算工作负载来说都很有用，因为其用户空间任务对由于内核计时器滴答信号造成的微秒级的延迟尤为敏感。

启用红帽企业版 Linux 7 中动态无时钟性能的方法，请见 < 第 3.3.1 节 “配置内核滴答记号时间” >。

3.1.3. 中断请求管理

中断请求或 IRQ 是请求及时关注的信号，是从硬件发送至处理器的。系统中的每个设备都分配到一个或多个 IRQ 号，以便能发送独一的中断信号。当启用中断时，收到中断请求的处理器会立即暂停执行当前应用程序线程，这是为了处理该中断请求。

因为中断了正常的运行，高中断率会严重降低系统性能，但减少中断的时间是可能的，可以设置中断关联或发送一批低优先率的中断（“*组合* 中断”）

关于调节中断请求的更多信息，请见 < 第 3.3.7 节 “设置中断关联” > 或 < 第 3.3.8 节 “使用 Tuna 配置 CPU、线程和中断关联” >。针对网络中断信息，请见 < 第 6 章 *网络* >。

3.2. 监控和诊断性能问题

红帽企业版 Linux 7 提供了大量工具，这些工具对与处理器及其配置相关的系统性能监和性能问题诊断很有帮助。本章概述了可用的工具并举例说明如何使用它们来监测和诊断与处理器相关的性能问题。

3.2.1. turbostat

Turbostat 在规定的间隔中给出计时器的结果以协助管理员识别服务器异常，例如过度耗电，无法进入深睡眠状态或是创建了不必要的系统管理中断（SMIs）。

turbostat 工具是 *内核工具* 数据包的一部分。支持在 AMD 64 和 Intel® 64 处理器的系统中使用。需要 root 特权来运行，处理器支持时间戳计时器以及 APERF 和 MPERF 型号的特定寄存器。

使用实例，请参见手册页：

```
$ man turbostat
```

3.2.2. numastat

重要

该工具在红帽企业版 Linux 6 的整个使用周期中都会接收到大量更新，虽然默认的输出和 Andi Kleen 写的原始工具相兼容，但提供任何选项或参数给 numastat 都会显著地改变其输出格式。

numastat 工具会列举每个 NUMA 节点内存数据给所有的进程和操作系统，并会告知管理员进程内存是散布于系统还是集中于某个节点。

通过处理器的 **top** 输出进行交互参照 **numastat** 输出，以确认进程线程是在同一个节点运行，此节点是进程内存分配节点。

Numastat 由 *numactl* 数据包提供。关于 **numastat** 输出的更多信息，请参见手册页：

```
$ man numastat
```

3.2.3. /proc/ 中断

/proc/interrupts 文件列举了从一个特殊的 I/O 设备发送至各处理器的中断数量，显示了中断请求（IRQ）数量、系统中各处理器处理该类型中断请求的数量，发送的中断类型以及以逗号分隔开的回应所列中断请求的设备列表。

如果一个特定的应用程序或是设备生成大量的中断请求给远程处理器处理，其性能就会受到影响。这种情况下，当应用程序或设备在处理中断请求时，可以在同一节点设置一个处理器，以此来缓解性能不佳的状况。将中断处理分配给特定处理器的方法，请见 < 第 3.3.7 节 “设置中断关联” >。

3.3. 配置建议

红帽企业版 Linux 提供了大量工具来协助管理员配置系统。本章概述了可用的工具并提供了使用它们在红帽企业版 Linux 7 中解决与处理器相关的性能问题的实例。

3.3.1. 配置内核滴答记号时间

默认情况下，红帽企业版 Linux 7 使用无时钟内核，它不会中断空闲 CPU 来减少用电量，并允许较新的处理器利用深睡眠状态。

红帽企业版 Linux 7 同样提供一种动态的无时钟设置（默认禁用），这对于延迟敏感型的工作负载来说是很有帮助的，例如高性能计算或实时计算。

要启用特定内核中的动态无时钟性能，在内核命令行中用 ***nohz_full*** 参数进行设定。在 16 核的系统中，设定 **nohz_full=1-15** 可以在 1 到 15 内核中启用动态无时钟内核性能，并将所有的计时移动至唯一未设定的内核中（0 内核）。这种性能可以在启动时暂时启用，也可以在 **/etc/default/grub** 文件中永久启用。要持续此性能，请运行 **grub2-mkconfig -o /boot/grub2/grub.cfg** 指令来保存配置。

启用动态无时钟性能需要一些手动管理。

- 当系统启动时，必须手动将 rcu 线程移动至对延迟不敏感的内核，这种情况下为 0 内核。

```
# for i in `pgrep rcu` ; do taskset -pc 0 $i ; done
```

- 在内核命令行上使用 ***isolcpus*** 参数来将特定的内核与用户空间任务隔离开。
- 可以选择性地为辅助性内核设置内核回写式 bdi-flush 线程的 CPU 关联：

```
echo 1 > /sys/bus/workqueue/devices/writeback/cpumask
```

验证动态无时钟配置是否正常运行，执行以下命令，其中 *stress* 是在 CPU 中运行 1 秒的程序。

```
# perf stat -C 1 -e irq_vectors:local_timer_entry taskset -c 1 stress -t
1 -c 1
```

可替代 *stress* 的是一个脚本，该脚本的运行类似 **`while :; do d=1; done`**。以下链接中的程序是另一个合适的替代程序： https://dl.fedoraproject.org/pub/epel/6/x86_64/repoview/stress.html。

默认的内核计时器配置在繁忙 CPU 中显示 1000 次滴答记号：

```
# perf stat -C 1 -e irq_vectors:local_timer_entry taskset -c 1 stress -t
1 -c 1
1000 irq_vectors:local_timer_entry
```

动态无时钟内核配置下，用户只会看到一次滴答记号：

```
# perf stat -C 1 -e irq_vectors:local_timer_entry taskset -c 1 stress -t
1 -c 1
1 irq_vectors:local_timer_entry
```

3.3.2. 设置硬件性能策略

`x86_energy_perf_policy` 工具允许管理员定义性能与能效的相对重要性。当处理器在性能与能效间权衡选择时，此信息可用来改变支持这一特征的处理器。

默认情况下适用于所有在 **`performance`** 模式下的处理器，它要求处理器的支持，由 **`CPUID.06H.ECX.bit3`** 显示，且必须在有 root 特权的情况下运行。

`x86_energy_perf_policy` 由 *kernel-tools* 数据包提供。如何使用 **`x86_energy_perf_policy`**，详情请见 <第 A.10 节 “x86_energy_perf_policy” > 或参见手册页：

```
$ man x86_energy_perf_policy
```

3.3.3. 使用 taskset 设置处理器关联

`taskset` 工具由 *util-linux* 数据包提供。**`Taskset`** 允许管理员恢复和设置进程中的处理器关联，或通过特定的处理器关联来启动一个进程。

重要

taskset 不能保证本地的内存配置。若需要本地内存配置的额外性能收益，红帽推荐使用 **numactl** 来替代 **taskset**。

关于 **taskset** 的更多信息，请见 <第 A.16 节 “taskset” > 或手册页：

```
$ man taskset
```

3.3.4. 使用 numaactl 管理 NUMA 关联

管理员可以通过特定的调度或内存安置策略来使用 **numactl** 运行进程。**Numactl** 也可以为共享内存片段或文件设置永久性策略，并设置处理器关联和进程的内存关联。

在 NUMA 拓扑系统中，处理器访问内存的速度会由于处理器和存储体之间距离的增加而减慢。因此，重要的是要对性能敏感的应用程序进行配置，以便它们能够从最近的且可能的存储体分配内存。最好是使用在同一 NUMA 节点的内存和 CPU。

对性能敏感的多线程应用程序经配置后在特定的 NUMA 节点上运行会比在特定的处理器上运行好处更多。这是否适合则取决于用户系统及应用程序的需求。如果多个应用程序线程访问同一缓存数据，那么对那些线程进行配置，使其在同一处理器上运行可能是合适的。但是，如果在同一处理器上运行的多线程访问及缓存的是不同数据，那么每个线程可能会收回之前线程访问的缓存数据。这就意味着每个线程会“缺失”缓存，会浪费运行时间来从磁盘中获取数据并在缓存中替代它。用户可以使用 **perf** 工具，收录于 <第 A.7 节 “perf” > 中，用它来查看大量的缓存缺失。

Numactl 提供大量的选择来协助管理处理器及内存关联。详情请见 <第 A.12 节 “numastat” > 或手册页：

```
$ man numactl
```

注意

numactl 数据包包括 **libnuma** 库。这个库提供了一个简单的编程接口至内核支持的 NUMA 策略，比起 **numactl** 应用程序，它可以用来进行更细致的调节。更多信息请参见手册页：

```
$ man numa
```

3.3.5. 使用 numad 进行自动化 NUMA 关联管理

numad 是一种自动化的 NUMA 关联管理后台程序。它对系统中的 NUMA 拓扑及资源用量进行监控，以便动态地改善 NUMA 资源配置及管理。

numad 也同样提供预先安置咨询服务，这一服务可以通过不同的作业管理系统来查询，并为处理器 CPU 的初始绑定及内存资源提供帮助。无论 numad 是以可执行程序或服务在运行，这一预先安置咨询都可用。

使用 **numad** 的方法，请见 <第 A.14 节 “numad” > 或参见手册页：

```
$ man numad
```

3.3.6. 调节调度策略

Linux 调度器执行大量的调度原则，以此决定线程运行的位置和时长。调度原则主要有两类：普通原则和实时原则。普通线程用于普通优先级任务，实时原则用于具有时效性且必须无中断完成的任务。

实时线程不受时间间隔的控制，这意味着它们将一直运行直至它们阻拦、退出、主动让步或是被更高优先权的线程预先安置。最低优先权的实时线程会先于其他普通原则的线程进行调度。

3.3.6.1. 调度原则

3.3.6.1.1. SCHED_FIFO 静态优先级调度

SCHED_FIFO （也叫做静态优先级调度）是一项实时策略，定义了每个线程的固定优先级。这一策略让管理员能改进事件响应的时间并减少延迟，这一策略建议无法运行较长时间且具有时效性的任务使用。

在使用 **SCHED_FIFO** 时，调度器会按优先级顺序扫描所有的 **SCHED_FIFO** 线程，并对准备运行的最高优先级线程进行调度。一个 **SCHED_FIFO** 线程的优先级级别可以是 1 至 99 之间的任何整数，99 是最高优先级。红帽建议一开始使用较小的数字，在确定了延迟问题后再增加优先级。

警告

由于实时线程不受时间间隔的控制，红帽不推荐设置 99 优先级。这会使同优先级的进程成为迁移线程或监控线程，如果线程进入一个计算机回路且这些线程被阻拦，它们将无法运行。这种情况下，单处理器系统最终会暂停。

管理员可以限制 **SCHED_FIFO** 的带宽以防止实时应用程序的程序员启用独占处理器的实时任务。

/proc/sys/kernel/sched_rt_period_us

该参数以微秒为单位来定义时间，是百分之百的处理器带宽。默认值为 **1000000** µs, 或1秒。

/proc/sys/kernel/sched_rt_runtime_us

该参数以微秒为单位来定义时间，用来运行实时线程。默认值为 **950000** µs, 或0.95秒。

3.3.6.1.2. SCHED_RR 轮循优先级调度

SCHED_RR 是 **SCHED_FIFO** 的一个轮循变形。这一策略在同优先级的多线程需要运行时很有用。

正如 **SCHED_FIFO** ，**SCHED_RR** 是一项实时策略，定义了每个线程的固定优先级。调度器会按优先级顺序扫描所有的 **SCHED_RR** 线程，并对准备运行的最高优先级线程进行调度。但是，和 **SCHED_FIFO** 不同，同优先级的线程在一定的时间间隔内是以循环的方式进行调度的。

用户可以使用 ***sched_rr_timeslice_ms*** 内核参数，并毫秒为单位设定这一时间间隔 (**/proc/sys/kernel/sched_rr_timeslice_ms**) 。最小值为1毫秒。

3.3.6.1.3. SCHED_OTHER 普通调度

SCHED_OTHER 是红帽企业版 Linux 7 中默认的调度策略。这一策略使用 CFS （完全公平排程器）让处理器能够平等地访问用此策略调度的所有线程。这一策略在有大量线程或数据吞吐量优先时最为有用，因为它能够随着时间而更为有效地调度线程。

在使用这一策略时，调度器会创建一个动态优先级列表，此列表一部分是基于每个进程线程的进程优先级。管理员可以改变一个进程的进程优先级，但是不能直接改变调度器的动态优先级列表。

改变进程的进程优先级，详情请见 *< Red Hat Enterprise Linux 7 Deployment Guide >* ，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获得。

3.3.6.2. 隔离 CPU

用户可以使用 ***isolcpus*** 开机参数来从调度器隔离一个或多个 CPU，以此防止调度器在此 CPU 上调度任何用户空间的线程。

一旦 CPU 被隔离，用户须手动分配进程至被隔离的 CPU，或使用 CPU 关联系统呼叫或 numactl 命令。

将系统中第三和第六 CPU 隔离至第八 CPU，添加如下至内核命令行：

```
isolcpus=2,5-7
```

用户也可使用 **Tuna** 工具来隔离CPU。**Tuna** 可以随时隔离 CPU，不仅仅局限于启动时。但这种隔离方法与 ***isolcpus*** 参数略有不同，并且目前尚未实现与 ***isolcpus*** 相关的性能收益。关于此工具的详情，请参见 <第 3.3.8 节 “使用 Tuna 配置 CPU、线程和中断关联” >。

3.3.7. 设置中断关联

中断请求有一个相关的关联属性***smp_affinity***，它能确定处理中断请求的处理器。若要提高应用程序的性能，就将中断关联和进程关联分配至同一处理器或分配至同一内核的处理器。这可以使特定的中断和应用程序线程共享高速缓存线路。

特定中断请求的中断关联值存储在相关的 **/proc/irq/*irq_number*/smp_affinity** 文件下。***smp_affinity*** 是存储为十六进制的位掩码来代表系统中所有的处理器。默认值为 **f**，这意味着一个中断请求可以在系统中任何处理器中处理。如果将这个值设为 **1** 则意味着只有 0 位处理器可以处理这一中断。

在超过 32 个处理器的系统中，用户须将 ***smp_affinity*** 的值限定为分散的 32 位组。例如，如果一开始只想使用 64 位处理器系统中的 32 位处理器来处理一个中断请求，可以运行：

```
# echo 0xffffffff,00000000 > /proc/irq/IRQ_NUMBER/smp_affinity
```

此外，如果 BIOS 导出其 NUMA 拓扑，**irqbalance** 服务可以使用此信息来处理节点上的中断请求，对请求服务的硬件来说此节点是本地节点。关于 **irqbalance**，详情请见 <第 A.1 节 “irqbalance（中断平衡）” >。

注意

如果在支持中断驱动且可以修改一个中断请求的 ***smp_affinity*** 系统中设置硬件，那么特定的处理器处理一个中断请求的决策就是硬件级别，它不会受内核的干扰。关于中断驱动，详情请见 <第 6 章 *网络* >。

3.3.8. 使用 Tuna 配置 CPU、线程和中断关联

Tuna 能够控制 CPU、线程及中断关联，并能给其所能控制的每类实体提供大量操作。完整的 **Tuna** 功能列表，请参见 <第 A.2 节 “Tuna” >。

要从一个或多个特定的 CPU 中移除所有线程，请运行如下命令，使用想要隔离的 CPU 数量来替换 *CPUs*。

```
# tuna --cpus CPUs --isolate
```

要在可运行特定线程的 CPU 列表中加入一个 CPU，请运行如下命令，使用想要加入的 CPU 数量来替换 *CPUs*。

```
# tuna --cpus CPUs --include
```

要将一个中断请求移动至特定的 CPU，请运行如下命令，用 CPU 数量替换 *CPU*，用想要移动且使用逗号分隔的中断请求列表替换 *IRQs*。

```
# tuna --irqs IRQs --cpus CPU --move
```

此外，用户可以使用如下命令来找到所有 **sfc1*** 模式的中断请求。

```
# tuna -q sfc1* -c7 -m -x
```

要改变一个线程的策略和优先级，请运行如下命令，使用想要改变的线程替换 *thread*，使用需要的线程运行策略名称替换 *policy*，用从 0（最低优先级）至 99（最高优先级）间的一个整数替换 *level*。

```
# tuna --threads thread --priority policy:level
```

第 4 章 内存

本章概述了红帽企业版 Linux 7 的内存管理功能。<第 4.1 节 “注意事项”> 探讨了与内存相关的会影响性能的因素。<第 4.2 节 “监控及诊断性能问题”> 教您如何使用红帽企业版 Linux 7 的工具来诊断与内存使用情况或配置细节相关的性能问题。<第 4.3 节 “配置工具”> 探讨了可以使用的工具和策略，以此解决红帽企业版 Linux 7 中与内存相关的性能问题。

4.1. 注意事项

对于适中的工作负载，红帽企业版 Linux 7 会默认优化。如果用户的应用程序或用例需要大量的内存，那么改变系统处理虚拟内存可以提高应用程序的性能。

4.1.1. 页面大小

物理内存管理区块称为页面。每一个页面的物理位置都映射到一个虚拟位置以便处理器能够访问内存。这种映射存储于一个叫做页面表的数据结构中。

默认情况下，一个页面大约有 4 KB。由于页面的默认大小非常小，因此用户需要很多页面来管理大量的内存。但是，页面表只能存储有限的地址映射，增加其存储地址映射的数量既昂贵又困难，因为要考虑到将性能等级保持在内存需求的范围内。

红帽企业版 Linux 通过静态大型分页来给每个页面管理大内存的能力。静态大型分页可以配置到 1 GB 大小。但很难对其进行手动管理，必须在启动时就分配好。

透明大型分页很大程度上是之余静态大型页面的一个自动选择。透明大型页面大小为 2 MB 且默认启动。它们有时会干扰对延迟敏感的应用程序，因此常常在延迟严重时被禁用。

配置大页面以提高应用程序性能，详情请见 <第 4.3.1 节 “配置大页面”>。

4.1.2. 转换后背缓冲区大小

从页面表中读取地址映射很耗时且资源很宝贵，因此 Linux 操作系统提供最近使用地址的缓存：转换后背缓冲区（TLB）。但默认的 TLB 只能缓存一定量的地址映射。如果需要的地址映射不在 TLB 中（即，TLB *missed*），系统仍然需要读取页面表以决定物理到虚拟的地址映射。

由于应用程序内存需求与用来缓存地址映射的页面大小之间的关系，较于对内存需求小的应用程序来说，对内存需求大的应用程序更容易受到 TLB 缺失造成的性能下降问题。因此无论何时，避免 TLB 缺失很重要。

红帽企业版 Linux 提供大型转换后背缓冲区 （大型 TLB），可以将内存分为大片段进行管理。这使大量的地址映射能同时进行缓存，以此降低 TLB 缺失的可能性，并提高需要大内存的应用程序的性能。

配置大型 TLB，详情请见 <第 4.3.1 节 “配置大页面”>。

4.2. 监控及诊断性能问题

红帽企业版 Linux 7 提供大量有用的工具来监控系统性能并诊断与系统内存相关的性能问题。本章概述了可用的工具，并举例说明如何使用它们来监控和诊断与内存相关的性能问题。

4.2.1. 使用 vmstat 监控内存使用量

Vmstat 由 *procps-ng* 数据包提供，输出用户系统进程、内存、网页、字块输入/输出、中断以及 CPU 活动等的报告。这是在机子最后一次启动或上一个报告之后提供的关于这些活动平均数的即时报告。

以下指令显示了各种事件计数和内存统计信息列表。

```
$ vmstat -s
```

使用 **vmstat** 的方法，详情请见 <第 A.9 节 “vmstat” > 或手册页：

```
$ man vmstat
```

4.2.2. 用 Valgrind 分析应用程序的内存使用量

Valgrind 是一个为用户提供空间二进制文件测量方法的框架。它包含大量的工具来概述和分析程序性能。本章列出的 **valgrind** 工具能帮助用户检测内存错误，例如未初始化的内存使用和不适当的内存分配及解除分配。

要使用 **valgrind** 或其工具，请安装 *valgrind* 数据包：

```
# yum install valgrind
```

4.2.2.1. 使用 Memcheck 分析内存使用量

Memcheck 是默认的 **valgrind** 工具。它检测并报告大量难以检测和诊断到的内存错误，例如：

- 不应发生的内存访问
- 使用未定义或未初始化的值
- 不正确的释放堆内存
- 指示字重叠
- 内存泄露

注意

Memcheck 只能报告这些错误，但并不能阻止它们发生。如果程序以通常会引起段错误的方式来访问内存的话，段错误仍然会发生。但**memcheck** 会在发生错误之前立刻记录一条信息。

由于 **memcheck** 使用测量工具，通过 **memcheck** 执行的应用程序会比平常运行起来慢 10-30 倍。

要在应用程序上运行 **memcheck**， 请执行以下指令：

```
# valgrind --tool=memcheck application
```

用户也可以使用以下选项来使 **memcheck** 的输出集中在特定的问题类型上。

--leak-check

在应用程序结束运行后，**memcheck** 会搜索内存泄露问题。默认值为 **--leak-check=summary**，在找到内存泄露时会显示其数量。用户可以指定 **--leak-check=yes** 或 **--leak-check=full** 来输出每个泄露问题的详情。禁用请设定 **--leak-check=no**。

--undef-value-errors

默认值为 **--undef-value-errors=yes**，使用未定义的值时会报错。用户还可设定 **--undef-value-errors=no** ，这将禁用此报告并略微提高 Memcheck 的速度。

--ignore-ranges

在查看可寻址内存时指定一个或多个 **memcheck** 应忽略的范围，例如， **--ignore-ranges=0xPP-0xQQ,0xRR-0xSS**。

要查看 **memcheck** 选项的完整列表，请参见 **/usr/share/doc/valgrind-version/valgrind_manual.pdf** 中的文档。

4.2.2.2. 使用 Cachegrind 分析缓存使用量

Cachegrind 会模拟应用程序与系统缓存层次结构和分支预测器间的交互作用。它会追踪模拟的第一级指令和数据缓存使用情况，以此检测出该级缓存与代码间不良的交互作用。它也会追踪最后一级缓存（第二或第三极）以便追踪内存访问。这样的情况下，使用 **cachegrind** 的应用程序运行起来会比通常慢 20-100 倍。

Cachegrind 会收集应用程序执行期间的统计数据，并且将概要输出至操作台。要在应用程序中运行 **cachegrind** ，请执行以下指令：

```
# valgrind --tool=cachegrind application
```

用户也可以使用以下选项来让 **cachegrind** 的输出集中在一个特定的问题上。

--I1

指定大小、关联性以及第一级指令缓存行大小的方法如下：**--I1=*size*,*associativity*,*line_size***。

--D1

指定大小、关联性以及第一级数据缓存行大小的方法如下：**--D1=*size*,*associativity*,*line_size***.。

--LL

指定大小、关联性以及最后一级缓存行大小的方法如下：**--LL=*size*,*associativity*,*line_size***。

--cache-sim

启用或禁用缓存访问和缺失数量的集合是默认启用的（**--cache-sim=yes**）。禁用此集合以及 **--branch-sim** 来使 **cachegrind** 不收集信息。

--branch-sim

启用或禁用分支指令及错误预测数量的集合是默认启用的（**--branch-sim=yes**）。禁用此集合以及 **--cache-sim** 来使 **cachegrind** 不收集信息。

Cachegrind 写入详细的分析信息至每个进程 **cachegrind.out.*pid*** 文件，其中， *pid* 是进程标识符。这一详细信息可以使用 **cg_annotate** 工具进行进一步处理，方法如下：

```
# cg_annotate cachegrind.out.pid
```

Cachegrind 也提供 cg_diff 工具，可以更为容易地在代码变化前后对程序性能进行记录。要对比输出文件，请执行以下命令：先用原始配置输出文件替代，再用后续配置输出文件替代。

```
# cg_diff first second
```

生成的输出文件可以使用 **cg_annotate** 工具来查看更多详情。

要查看 **cachegrind** 选项的完整列表，请参见 **/usr/share/doc/valgrind-version/valgrind_manual.pdf** 中的文档。

4.2.2.3. 使用 Massif 分析堆栈空间

Massif 测量特定应用程序的堆空间。它测量可用空间和额外用来记录和调准的空间。**massif** 有助于用户了解减少应用程序内存使用的方法，以便提高运行速度，减少应用程序耗尽系统交换空间的可能性。使用 **massif** 执行的应用程序运行起来比平常通常慢 20 倍左右。

要在一个应用程序中运行 **massif**，请执行如下命令：

```
# valgrind --tool=massif application
```

用户也可以使用以下选项来将 **massif** 的输出集中在一个特定的问题上。

--heap

设定 **massif** 是否分析堆。默认值为 **--heap=yes**。要禁用堆分析可设置为 **--heap=no**。

--heap-admin

堆分析启用时要设定每个用于管理的数据块字节数。默认值为 **8** 字节。

--stacks

设定 **massif** 是否分析堆。默认值为 **--stack=no** 是由于堆分析会大大减缓 **massif**。将这一选项设置为 **--stack=yes** 来启用堆分析。要注意的是，**massif** 会假设主要的堆始于零值，这是为了更好地显示与所分析的应用程序相关的堆尺寸的变化。

--time-unit

设定 **massif** 收集分析数据的间隔。默认值为 **i**（执行指令）。用户也可以指定 **ms**（毫秒或实时）和 **B**（分配或收回的堆栈字节数）。检查分配的字节数有利于短期运行的应用程序及测试，因为对于不同的硬件来说，它是最具重复性的。

Massif 将分析数据输出至 **massif.out.*pid*** 文件中，该文件中的 *pid* 是指定应用程序的进程标识符。**ms_print** 工具将此分析数据绘成图表，以此显示执行应用程序的内存消耗，也包括峰值内存分配点负责分配的站点详情。要绘制 **massif.out.*pid*** 文件中的数据，请执行以下指令：

```
# ms_print massif.out.pid
```

要查看 **Massif** 选项的完整列表，请参见 **/usr/share/doc/valgrind-version/valgrind_manual.pdf** 中的文档。

4.3. 配置工具

内存使用量往往是通过设置一个或多个内核的参数值来进行配置的。这些参数可以通过改变在 /proc 文件系统中的文件内容来进行暂时设置，或是通过设置系统核心参数工具来进行永久设置，此工具由 *procps-ng* 数据包提供。

例如，要将 overcommit_memory 参数暂时设置为 1，请运行以下指令：

```
# echo 1 > /proc/sys/vm/overcommit_memory
```

要永久设置这个值，请运行以下指令：

```
# sysctl vm.overcommit_memory=1
```

暂时设置一个参数有利于决定此参数对系统的影响。用户可以在确定了参数值有预期的效果之后再将其设置为永久值。

4.3.1. 配置大页面

大页面依赖于连续的内存区域，因此最好在启动时，也就是在内存变为片段前就定义好大页面。为此，请添加以下参数至内核启动命令行：

hugepages

启动时在内核中定义 2MB 定值大页面的数量。默认值为 0。只有在系统拥有足够的物理持续性空闲页面时才能进行分配（或是收回）大页面。此参数保留的页面不能用于其他目的。

此值可以在启动后通过改变 **/proc/sys/vm/nr_hugepages** 文件值来调节。

更多详情请参阅相关内核文档，默认安装于 **/usr/share/doc/kernel-doc-kernel_version/Documentation/vm/hugetlbpage.txt** 中。

/proc/sys/vm/nr_overcommit_hugepages

通过超量使用内存来定义系统所能创建和使用的最大数量的额外大页面。在此文件中写入任何非零的值，表示系统包含此数目的大页面，在不变的页面池耗尽后，这些大页面便来自于内核的常规页面池。由于这些额外的大页面是未使用过的，因此它们会释放并返回至内核的常规页面池中。

4.3.2. 配置系统内存容量

本章探讨与内存相关的内核参数，它们可能有助于提高用户系统的内存使用率。基于测试目的，可以通过改变 **/proc** 文件系统中相应的文件值来暂时设置这些参数。一旦决定了能提供最佳性能的值，就可以使用 **sysctl** 指令来进行永久设置。

4.3.2.1. 虚拟内存参数

此章中的参数都在 **/proc/sys/vm** 内，除非另有标明。

dirty_ratio

一个百分比值。当整个系统内存的这一百分比值被修改时，系统会通过运行 **pdflush** 将改动写入磁盘。默认值为 **20**%。

dirty_background_ratio

一个百分比值。当整个系统内存的这一百分比值被修改时，系统会在后台将改动写入磁盘。默认值为 **10** %。

overcommit_memory

定义用来决定接受或拒绝一个大内存请求的注意事项。

默认值为 **0**。默认情况下，内核执行探索法内存超量使用，是通过估算可用内存大小和由于太大而失败的请求来进行处理的。但由于内存分配使用的是探索法而不是精确算法，这一设置导致超载内存是可能的。

当这一参数设置成 **1** 时，内核不执行内存超量使用处理。这增加了内存超量的可能性，但提高了内存密集型任务的性能。

当这一参数设置成 **2** 时，内核拒绝请求，即请求的内存等于或大于总的可用交换空间，以及在 ***overcommit_ratio*** 中指定的物理 RAM 的百分比。这减少了超量使用内存的风险，但仅在系统交换空间大于物理内存时推荐此设置。

overcommit_ratio

当 ***overcommit_memory*** 设置为 **2** 时，设定所考虑的物理 RAM 的百分比。默认值为 **50**。

max_map_count

定义一个进程可以使用的最大内存映射区域数量。默认值（**65530**）适用于大部分情况。如果应用程序需要映射超过此数量的文件，可增加此值。

min_free_kbytes

指定千字节的最小数量，使之在整个系统中都保持空闲。这是用来给每一个低内存区决定一个合适的值，每一个低内存区都按照其大小比例分配了大量保留的空闲页面。

警告

极值会损坏用户的系统。将 ***min_free_kbytes*** 设置为一个极小的值以防止系统回收内存，回收内存会导致系统锁死，以及 OOM-killing 进程。但是，将 ***min_free_kbytes*** 设置过高 （例如，整个系统内存的 5–10% ）会使系统立即进入一种内存不足的状态，导致系统花太多时间来回收内存。

oom_adj

在系统内存不足，并且 ***panic_on_oom*** 参数设置成 **0** 的情况下，**oom_killer** 功能会结束进程，直至系统可以恢复，从最高的 **oom_score** 进程开始。

oom_adj 参数有助于确定一个进程的 ***oom_score***。此参数以每一个进程标识符为单位进行设置。值为 **-17** 时会禁用进程的 **oom_killer**。其他有效值是从 **-16** 到 **15**。

注意

由一个调整过的进程而产生的进程会继续该进程的 **oom_score**。

swappiness

一个从 **0** 到 **100** 的值可以控制系统交换的程度。高值优先考虑系统效率，并在进程不活跃时主动交换物理内存耗尽的进程。低值优先考虑响应度，并且尽可能久地避免交换物理内存耗尽的进程。默认值为 **60**。

4.3.2.2. 文件系统参数

此章中的参数都在 **/proc/sys/fs** 内，除非另有标明。

aio-max-nr

定义在异步输入/输出环境中允许的最大事件数量。默认值为 **65536**。修改此值不会预分配或改变任何内核数据结构的大小。

file-max

定义内核分配的最大的文件句柄数量。默认值与内核中的 **files_stat.max_files** 值相匹配，

将此值设为最大值 ***NR_FILE***（8192，在红帽企业版 Linux 中）或是以下结果：

```
(mempages * (PAGE_SIZE / 1024)) / 10
```

增加此值可以解决由于缺少可用的文件句柄而引起的错误。

4.3.2.3. 内核参数

此章中的参数都在 **/proc/sys/kernel** 内，除非另有标明。

msgmax

以字节为单位，定义任何一个在信息队列中的信息可能的最大值。该值不能超过队列的大小（***msgmnb***）。默认值为 **65536**。

msgmnb

以字节为单位，定义每一个信息队列的最大值。默认值为 **65536**。

msgmni

定义信息队列标识符的最大数量（以及队列的最大数量）。在 64 位架构的系统中，默认值为 **1985**。

shmall

定义页面上共享内存的总量，这些内存是系统可以同时使用的。

shmmax

定义页面上内核允许的单个共享内存片段的最大值。

shmmni

定义系统范围内最大的共享内存片段数量。在所有系统中的默认值为 **4096**。

threads-max

定义系统范围内内核能同时使用的最大线程量。默认值与内核参数 ***max_threads*** 相同，或为以下结果：

```
mempages / (8 * THREAD_SIZE / PAGE SIZE )
```

最小值为 **20**。

第 5 章 存储和文件系统

此章节概述支持的文件系统和配置选项，此选项影响在红帽企业版 Linux 7 中 I/O 和文件系统的应用程序性能。〈第 5.1 节 “注意事项” 〉讨论了影响性能的 I/O 和文件系统相关因素。〈第 5.2 节 “性能问题监控和诊断”〉教授如何使用红帽企业版 Linux 7 工具诊断与 I/O 或者文件系统配置细节相关的性能问题。〈第 5.3 节 “配置工具” 〉讨论了为解决红帽企业版 Linux 7中 I/O 和文件系统相关的性能问题，您可使用的工具和策略。

5.1. 注意事项

存储和文件系统性能的合理设置在很大程度上取决于存储目的。 I/O 和文件系统性能会受到下列因素的影响：

- 数据写入或读取模式
- 数据重新排列与底层几何
- 块大小
- 文件系统大小
- 日记大小和位置
- 记录访问次数
- 确保数据可靠性
- 预取数据
- 预先分配磁盘空间
- 文件碎片
- 资源争用

阅读此章节可了解影响文件系统吞吐量、可伸缩性、响应能力、资源使用和可用性的格式和挂载选项。

5.1.1. 固态硬盘

SSD（固态硬盘）使用闪存芯片而非旋转磁盘存储永久数据。它们为逻辑块地址内的全部数据提供恒定访问时间，且不会像它们旋转的对应物那样出现可测量的搜寻成本。每千兆字节的存储空间更昂贵且存储密度更小，但比 HDD 延迟时间短、吞吐量更大。

当在 SSD 上使用的块接近磁盘容量，性能通常会降低。降低的程度因供应商的不同而异，在此情况下，所有设备性能都会降低。启用放弃有助于缓解性能降低；更多细节，请参见〈第 5.1.4.3 节 “维护”〉 。

默认 I/O 调度器和虚拟内存选项适用于 SSD。

SSD 部署建议的更多信息，可从 《*红帽企业版 Linux 7 存储管理指南*》获得， 参见 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

5.1.2. I/O 调度器

I/O 调度器决定 I/O 操作何时运行在存储设备上以及运行多久。它也被称为 I/O elevator（I/O 升降机）。

红帽企业版 Linux 7 提供三种 I/O 调度器。

deadline

除了 SATA 磁盘为所有块设备的默认 I/O 调度器。**Deadline** 尝试为指向到达 I/O 调度器的请求提供有保障的延迟。该调度器适合大多数用例，尤其适用于读取操作比写入操作更频繁的请求。

将排队的 I/O 请求分类为读或者写批处理，并按照 LBA 递增顺序执行。默认设置下，读批处理优先于写批处理，这是因为应用更可能阻止读取 I/O。批处理后，**deadline** 检查写入操作因等待处理器时间而处于多久的“饥饿”状态，并且适当地调度下一个读批处理或写批处理。解决批处理的请求数量、发出写批处理的读批处理数量、以及请求过期前的时间量都是可配置的，更多细节，请参见〈第 5.3.4 节 “调整期限调度器”〉。

cfq

默认调度器只适用于标识为 SATA 硬盘的设备。完全公平队列调度器，**cfq**，将进程分成三个独立类别：实时、尽其所能和空闲。实时类别的进程总是先于尽其所能类别进程执行，而尽其所能类别进程总是在空闲类别进程之前执行。这意味着实时类别的进程可以使尽其所能和空闲进程等待处理器时间而忍受“饥饿”。默认设置下，分配进程到尽其所能类别。

Cfq 使用历史数据来预测应用是否会在不久之后发出更多 I/O 请求。如果将有更多 I/O 请求，**cfq** 空闲则会等待新的 I/O，即使有来自其他进程的 I/O 在等待处理。

因为有空闲的趋势，cfq 调度器不应用于连接不会引起大量搜寻 penalty（惩罚）的硬件，除非它为此目的而被调整。cfq 调度器也不应用于连接其他断续工作型调度器，例如基于主机的硬件 RAID 控制器，因为这些调度器的堆积有造成大量延迟的趋势。

Cfq 行为是可高度配置的，更多细节，请参见〈第 5.3.5 节 “调整 cfq 调度器”〉。

noop

noop I/O 调度器执行简单的 FIFO（先进先出）调度算法。请求通过简单的最后选中的缓存数据在一般块层合并。对于使用最快存储的受 CPU 限制的系统，这是最佳调度器。

设置不同默认 I/O 调度器的细节，或为特定设备指定不同的调度器，请参见〈第 5.3 节 “配置工具”〉。

5.1.3. 文件系统

欲了解红帽企业版 Linux 7 中支持文件系统的细节、推荐用例、通常情况下的文件系统可用的格式和挂载选项，请阅读此章节。为这些文件系统的调整推荐，请参见〈第 5.3.7 节 “为性能配置文件系统”〉。

XFS

XFS 是一个可靠的、且可高度缩放的 64 位文件系统。它是红帽企业版 Linux 7 中默认文件系统。XFS 使用基于分区的分配，具有一些分配方案，包括预先分配和延迟的分配，这两种都会减少碎片和辅助性能。它也支持促进故障恢复的元数据日志。当挂载并激活时，能够对 XFS 进行碎片整理和放大，红帽企业版 Linux 7 支持几种 XFS 特定的备份和还原工具程序。

自红帽企业版 Linux 7.0 GA 起，XFS 支持最大容量可达 500 TB 的文件系统，以及最大容量为 8 EB 的文件偏移量（稀疏文件）。管理 XFS 的细节，请参见《*红帽企业版 Linux 7 存储管理指南*》。可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。如有调整 XFS 的特殊需求而需要协助，请参见〈第 5.3.7.1 节 “调整 XFS”〉。

Ext4

Ext4 是 ext3 文件系统的可缩放扩展。它的默认行为对大部分工作负载是最佳的。然而，它只支持最大容量为 50 TB的文件系统以及最大容量为 16 TB 的文件。管理 ext4 的细节，请参见《*红帽企业版 Linux 7 存储管理指南*》，可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/，如有调整 ex4 的特殊需求而需要协助，请参见〈第 5.3.7.2 节 “调整 ext4”〉。

Btrfs（技术预览）

Btrfs 是提供缩放性、容错和方便管理的 copy-on-write（写时复制）文件系统。它包括内置快照和 RAID 支持，通过数据和元数据校验来提供数据的完整性。它也通过数据压缩提高性能及使用空间的效率。Btrfs 作为一种技术预览，支持最大容量可达 50 TB 的文件系统。

Btrfs 是最适合桌面存储和云存储的。最初格式化设备时，最好按照预期使用而调整 btrfs。

红帽企业版 Linux 7 提供 Brtfs 作为技术预览。技术预览特征的细节，请参见 https://access.redhat.com/site/support/offerings/techpreview/。

管理 brtfs 的细节，请参见《*红帽企业版Linux 7存储管理手册*》，可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。如有调整 btrfs 的特殊需求而需要协助，请参见〈第 5.3.7.3 节 “调整 btrfs”〉。

GFS2

GFS2 是具有极高可用性附加装置的一部分，为红帽企业版 Linux 7 提供簇文件系统支持。GFS2 集群提供所有服务器一致的文件系统图像，允许服务器在一个单独共享文件系统中读取和写入。

GFS2 支持最大容量可达 250 TB 的文件系统。

管理 GFS2 的细节，请参见《*红帽企业版 Linux 7 存储管理指南*》。可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。如有调整 GFS2 的特殊需求而需要协助，请参见〈第 5.3.7.4 节 “调整 GFS2”〉。

5.1.4. 文件系统的一般调整注意事项

此章节涵盖普遍适用于所有文件系统的调整注意事项。特定文件系统的调整推荐，请参见〈第 5.3.7 节 “为性能配置文件系统”〉。

5.1.4.1. 格式时间注意事项

在设备格式化后，文件系统配置的部分决定不能改变。此章节包含格式化存储设备前必须要做的决定的可用选项。

大小

按照工作负载创建合理大小的文件系统。按相应比例，较小的文件系统的备份次数也较少，且文件系统检查所需时间和内存也更少。然而，如果您的文件系统太小，性能将因大量碎片而降低。

块大小

块是文件系统中工作的单位。块大小决定单个块能存储多少数据，也因而决定能够同时读写的数据最小量。

默认块大小适用于大部分用例。然而，如果块大小（或者多个块大小）和通常同时读写的数据数量一样大，或者略大时，文件系统将执行得更好、存储数据更加有效率。小文件仍将使用一个完整的块。文件分布在多个块中，但这会造成额外的运行时间开销。另外，一些文件系统受限于一定数量的块，转而限制文件系统最大尺寸。

使用 **mkfs** 指令格式化设备时，块大小作为文件系统选项的一部分而被指定。指定块大小的参数随文件系统变化，文件系统的细节，请参见 **mkfs** 手册页。例如，查看格式化 XFS 文件系统时可用的选项，执行下列命令：

```
$ man mkfs.xfs
```

几何

文件系统几何与文件系统中数据的分布相关。如果系统使用带状存储器，例如 RAID ，可在格式化设备时，通过重新排列数据和底层存储几何的元数据提高性能。

很多数据导出的推荐几何在使用特定文件系统格式化设备时会被自动设置。如果设备没有导出这些推荐几何，或您想要变更推荐设置，那么您在使用 **mkfs** 格式化设备时，需要手动指定几何 。

指定文件系统几何的参数随文件系统而变化；文件系统细节请参见 **mkfs** 手册页。例如，查看格式化 ext4 系统时可用的选项，执行下列命令：

```
$ man mkfs.ext4
```

外部日记

日志文件系统会在执行写操作之前，将写操作期间发生的变化记录到日志文件中。 这会降低系统发生故障、电源故障时日志的存储设备损坏的可能性，并加速恢复过程。

元数据密集工作负载涉及日志的频繁更新。大型日志使用更多内存，但会减少写操作的频繁性。此外，可通过将设备日志置于和主要存储一样快或者更快的专用存储上，提高带有元数据密集工作负载的设备的寻道时间。

警告

确保外部日志的可靠性。失去外部日志，设备将导致文件系统损坏。

外部日志必须在格式化时便创建，并在挂载期间指定日志设备。细节请参见 **mkfs** 和 **mount** 手册页。

```
$ man mkfs
```

```
$ man mount
```

5.1.4.2. 挂载时间注意事项

此章节包含适用于大部分文件系统的调整决定，且可在挂载设备时指定。

Barrier（屏障）

文件系统 barrier 确保文件系统元数据正确写入到永久存储并排序，使用 **fsync** 传输的数据在断电下得以存留。以前红帽企业版 Linux 版本中，启用文件系统 barrier 会明显放慢严重依赖 **fsync** 的应用程序，或者创建和删除很多小文件。

红帽企业版 Linux 7 中，文件系统 barrier 性能的得到的改善使禁用的文件系统 barrier 的性能影响变得极小（小于3%）。

更多信息，请参见《*红帽企业版 Linux 7 存储管理指南*》，可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

访问时间

每次读取文件，它的元数据随访问时间（**atime**）更新。这涉及额外的写入 I/O。在大多数情况下，这样的开销是最小的，因为在默认设置下，前次访问时间早于上次修改时间（**mtime**） 或者状态变化（**ctime**），红帽企业版 Linux 7 只更新 **atime** 字段。

然而，如果更新元数据耗时，且并不对准确访问时间做要求，您可以使用 **noatime** 挂载选项挂载文件系统。读取文件时会禁用元数据的更新。它也会启用 **nodiratime** 行为，读取目录时，该行

为禁用元数据的更新。

预读

预读行为通过预取可能立即需要的数据，并且将其加载到可比在磁盘上更快检索的页面缓存中加速文件访问。预读值越高，系统预取数据越早。

红帽企业版 Linux 根据对于文件系统的检测，尝试设置一个合适的预读值。然而，检测不可能总是准确的。例如，如果存储数组将自己作为单一 LUN 展示给系统，系统会检测单一 LUN，但不会为数组设置合适的预读值。

涉及大量数据流的量数据流的顺序 I/O 的工作负载常常受益于高预读值。红帽企业版 Linux 7 提供的相关存储调整配置文件提高预读值，和使用 LVM 条带化一样，但这些调整对于所有工作负载而言并不总是足够的。

定义预写行为的参数随着文件系统而变化；请参见手册页。

```
$ man mount
```

5.1.4.3. 维护

定期丢弃文件系统不用的块是对于固态硬盘和精简配置存储的建议做法。有两种丢弃不使用的块的做法：batch discard（批量丢弃）和 online discard（网络丢弃）。

batch discard（批量丢弃）

这种丢弃方式是 **fstrim** 指令的一部分。它丢弃文件系统中与管理员指定的标准相配的所有不使用的块。

红帽企业版 Linux 7 支持 XFS 和 ext4 格式化设备上的 batch discard，这些设备支持实际丢弃操作即（**/sys/block/*devname*/queue/discard_max_bytes** 值不为 0 的 HDD 设备，和 **/sys/block/sda/queue/discard_granularity** 不为 **0** 的 SSD 设备）。

online discard（网络丢弃）

这种方式的丢弃操作在挂载时间使用 **discard** 选项配置，实时运行不受用户干扰。然而，online discard 只丢弃从使用转换到空闲的块。红帽企业版 Linux 7 支持 XFS 和 ext4 格式化设备上的 online discard。

红帽推荐 batch discard，除非要求用 online discard 维持性能，或 batch discard 不可用于系统工作负载。

预先分配

预先分配将硬盘空间标记为已经将磁盘空间分配给一个文件，而未将数据写入该空间。这可用于限制数据碎片和较差的读取性能。红帽企业版 Linux 7 支持挂载时间内 XFS、ext4、 和 GFS2 设备上预先分配空间。对您文件系统的合适参数，请参见 **mount** 手册页。应用程序也可通过使用 **fallocate(2) glibc** 调用从预先分配空间受益。

5.2. 性能问题监控和诊断

红帽企业版 Linux 7 提供一些工具，这些工具有助于监控系统性能和诊断与 I/O、文件系统及其配置相关性能问题。此章节概述可用的工具，并列举如何使用这些工具监控和诊断与 I/O 及文件系统相关的性能问题的例子。

5.2.1. 使用 vmstat 监控系统性能

Vmstat 报告整个系统的进程、内存、分页、阻止 I/O、中断和 CPU 活动。它能帮助管理员决定 I/O 子系统是否对任何性能问题负责。

下列是与 I/O 性能最相关的信息：

si

切换进硬盘，或者以 KB 为单位写入交换空间。

so

从硬盘中切换出，或者以 kb 为单位从交换空间读取。

bi

写入块，或者以 kb 为单位阻止写操作。

bo

读取块，或者以 kb 为单位阻止读操作。

wa

等待 I/O 操作完成的队列部分。

交换空间和数据在同一设备上时，而且是内存使用指示器，切换进硬盘和从硬盘中切换出尤为有用。

另外，空闲、缓存和缓存列能帮助识别回写频率。缓存值突然下降和空闲值的增加表明回写和页面缓存无效。

如果使用 **vmstat** 的分析显示 I/O 子系统对性能下降负责，管理员可用 **iostat** 确定承担责任的 I/O 设备。

vmstat 由软件包 *procps-ng* 提供。更多使用 **vmstat** 的信息，请参见手册页：

```
$ man vmstat
```

5.2.2. 使用 iostat 监控 I/O 性能

Iostat 是由 *sysstat* 软件包提供的。它报告加载在您系统中的 I/O 设备。如果使用 **vmstat** 的分析显示 I/O 子系统对性能下降负责，您可以使用 **iostat** 来确定负责的 I/O 设备。

您可以使用 **iostat** 手册页中定义的参数，将 **iostat** 报告的输出集中在特定设备上。

```
$ man iostat
```

5.2.2.1. 使用 blktrace 详细说明 I/O 分析

Blktrace 提供 I/O 子系统上时间如何消耗的详细信息。配套工具 **blkparse** 从 **blktrace** 读取原始输出，并产生人们可读的输入和输出操作摘要，该摘要由 **blktrace** 记录。

更多此工具的细节请参见手册页：

```
$ man blktrace
```

```
$ man blkparse
```

5.2.2.2. 用 btt 分析 blktrace 输出

Btt 作为 *blktrace* 软件包的一部分而被提供。它分析 **blktrace** 输出，并显示该数据用在每个 I/O 栈区域的时间量，使它更容易在 I/O 子系统中发现瓶颈。

例如，如果 **btt** 显示发送至块层（**Q2Q**）的请求之间的时间比用于块层（**Q2C**）的请求之间的总时间长，那么 I/O 子系统可能不对性能问题负责。如果设备花了很长时间处理请求（**D2C**），该设备可能超载，或者发送给该设备的工作负载可能不是最佳的。如果块 I/O 队列等待很长时间，请求（**Q2G**）才分配给块 I/O 队列，这可能表明正在使用的存储不能够为 I/O 负载提供服务。

更多此工具的细节请参见手册页：

```
$ man btt
```

5.2.2.3. 使用 seekwatcher 分析 blktrace 输出

seekwatcher 工具可使用 **blktrace** 输出随时间绘制 I/O 图表。它集中在硬盘 I/O 的 LBA（逻辑块地址）、每秒以兆计算的吞吐量、每秒搜寻的数量和每秒 I/O 操作。这能帮助您了解何时达到设备操作系统每秒的限制。

更多此工具的细节请参见手册页：

```
$ man seekwatcher
```

5.2.3. 使用 SystemTap 监控存储

《*红帽企业版 Linux 7 SystemTap 入门指南*》 包含几个有助于配置和监控存储性能的示例脚本。

下列的 **SystemTap** 示例脚本与存储性能有关，并可能有助于诊断存储或文件系统性能问题。默认设置下，安装它们至 **/usr/share/doc/systemtap-client/examples/io** 目录下。

disktop.stp

每 5 秒检查读／写硬盘状态并输出在此期间的前十项。

iotime.stp

显示用在读操作和写操作的时间量，以及读和写的字节量。

traceio.stp

根据观察到的累计 I/O 流，显示每秒前十项可执行文件。

traceio2.stp

在特定设备进行读和写操作时，显示可执行的名称和进程标识符。

inodewatch.stp

每当在特定主要／次要设备上的特定 inode 上进行读或者写操作时，显示可执行的名称和进程标识符。

inodewatch2.stp

每当在特定主要／次要设备上的特定 inode 上属性发生变化时，显示将可执行的名称、进程标识符、和属性。

《*红帽企业版 Linux 7 SystemTap 入门指南*》可在网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

5.3. 配置工具

红帽企业版 Linux 提供一些帮助管理员配置存储和文件系统的工具。此章节概述可用的工具并提供示例阐述如何使用这些工具解决红帽企业版 Linux 7 中 I/O 和文件系统相关的性能问题。

5.3.1. 配置存储性能的调整配置文件。

Tuned 和 **tuned-adm** 提供一些旨在为特定用例提高性能的配置文件。下列配置文件对于提高存储性能尤其有用。

- 延迟性能
- 吞吐量性能（默认）

如需配置系统中的配置文件，请运行以下命令，用您想用的配置文件名称替代 *name*。

```
$ tuned-adm profile name
```

tuned-adm recommend 命令为系统推荐合适的配置文件。在安装时它也会为系统设置默认配置文件，因此可用于返回默认配置文件。

配置文件或其他配置选项更多细节，请参见 第 A.6 节 “tuned-adm”。

5.3.2. 设置默认 I/O 调度器

如果设备的挂载选项没有指定调度器，可使用默认 I/O 调度器。

如需设置默认 I/O 调度器，在重启时通过向内核命令行附加 elevator 参数来指定欲使用的调度器，或通过编辑 **/etc/grub2.conf** 文件。

```
elevator=scheduler_name
```

5.3.3. 为设备配置 I/O 调度器

如需设置特定存储设备的调度器或调度器优先顺序，编辑 **/sys/block/*devname*/queue/scheduler** 文件，*devname* 为您欲配置的设备名称。

```
# echo cfq > /sys/block/hda/queue/scheduler
```

5.3.4. 调整期限调度器

使用 **deadline** 时，排队的 I/O 请求将分为读批处理或者写批处理，然后按照 LBA 递增的执行顺序调度。默认设置下，读批处理比写批处理优先处理，这是因为在读 I/O 上应用程序易被阻止。在批处理被处理后，**deadline** 会检查写操作因等待处理器时间而处于多久的“饥饿”状态，并合理调度下一个读或者写批处理。

下列参数影响 **deadline** 调度器行为：

fifo_batch

单个批处理中读操作或写操作发出的数量。默认值为 **16**。值越高，吞吐量也会更多，但也会增加延迟。

front_merges

如果您的工作负载从不产生正面合并，可调整的参数设置为 **0**。然而，除非您已经测试了该检查的开销，红帽推荐 **1** 的默认值。

read_expire

应为服务调度读请求中毫秒的数量。默认值为 **500**（0.5秒）。

write_expire

应为服务调度写请求中毫秒的数量。默认值为 **5000**（5秒）。

writes_starved

先于写批处理而处理的读批处理数量。该值越高，给读批处理的优先更多。

5.3.5. 调整 cfq 调度器

使用 **cfq** 时 ，进程分为三类：实时、尽其所能和空闲。尽其所能进程之前调度所有实时进程，而空闲进程之前调度尽其所能进程。默认设置下，进程归类为尽其所能。可使用 **ionice** 命令手动调整进程分类。

通过使用下列参数进一步调整 **cfq** 调度器的行为。这些参数通过改变 **/sys/block/*devname*/queue/iosched** 目录下的指定文件，基于每个设备设置的。

back_seek_max

cfq 将执行向后搜寻以千字节计算的最大距离。默认值是 **16** KB。 向后搜寻通常会损害性能，因此不推荐大的值。

back_seek_penalty

磁头在决定向前还是向后移动时，乘法器应用于向后搜寻。默认值为 **2**。如果磁头位置是 1024 KB，并且在系统中有等距的请求（例如：1008 KB 和 1040 KB），***back_seek_penalty*** 应用于向后搜寻距离和磁盘向前移动。

fifo_expire_async

异步（缓冲写入）请求以毫秒计算的可能持续无服务的时间长度。在这个时间过期之后，一个单独的“饥饿”的异步请求移动至配送列表。默认值为 **250** 毫秒。

fifo_expire_sync

同步（读取或者**O_DIRECT** 写入）请求以毫秒计算的可能持续无服务的时间长度。在这个时期过期后，一个单独的“饥饿”的同步请求被移动到配送列表。默认值为 **125** 毫秒。

group_idle

默认设置下，参数设为 **0**（禁用）。设置为 **1**（禁用）时，**cfq** 调度器空闲在控制组中发出I/O的最后进程里。如使用成比例的重量 I/O 控制组，或 ***slice_idle*** 设置为 **0** （在快速存储上）会有帮助。

group_isolation

默认设置下，该参数设置为 **0**（禁用）。设置为 **1**（启用）时，它提供组之间更强的隔离，但是吞吐量会减少，这是因为公平性用于随机和顺序工作负载。***group_isolation*** 禁用时（设置为 **0**），公平性只提供给顺序工作负载。更多信息，请参见安装文件 **/usr/share/doc/kernel-doc-*version*/Documentation/cgroups/blkio-controller.txt**。

low_latency

默认设置下，设置参数为 **1** （启用）。启用后，通过为设备上发出 I/O 的每个进程提供最大为 **300** ms 的等待时间，**cfq** 更注重公平性而非吞吐量。设置参数为 **0** 时（禁用），目标延迟被忽略，每个进程接受完整时间片。

quantum

该参数定义 **cfq** 在同一时间发送给一个设备的 I/O 请求的数量，实质上是对队列深度的限制。默认值为 **8** 请求。使用的设备可能支持更大的队列深度，但增加量程值也会导致延迟增加，尤其是大的顺序写工作负载。

slice_async

该参数定义分配给每个发出异步 I/O 请求的进程的时间片（以毫秒计算）长度。默认值为 **40** 毫秒。

slice_idle

该参数指定等待下一步请求时以毫秒计算的 cfq 空闲时间长度。默认值为 **0**（队列无空闲或者 service tree level）。默认值对于外部 raid 存储器的吞吐量是理想的，由于增加了搜寻操作的整体数量，而降低内部 non-RAID 存储器的吞吐量。

slice_sync

该参数定义分配给每个发出同步 I/O 请求的进程的时间片长度（以毫秒计算）。默认值为 **100** ms。

5.3.5.1. 为快速存储调整 cfq

不向无法遭受大搜寻 penalty（惩罚）的硬件推荐 **cfq** 调度器，例如快速外部存储数列或者固态硬盘。如果您需要在此存储上使用 **cfq** ，需要编辑下列配置文件：

- 设置 **/sys/block/*devname*/queue/ionice/slice_idle**为**0**
- 设置 **/sys/block/*devname*/queue/ionice/quantum** 为**64**
- 设置 **/sys/block/*devname*/queue/ionice/group_idle**为 **1**

5.3.6. 调整 noop 调度器

noop I/O 调度器主要对使用快速存储的受 cpu 限制的系统有用。请求在块层合并，因此通过编辑 **/sys/block/sdX/queue/** 目录中的文件中块层参数，修改**noop** 行为。

add_random

一些 I/O 事件会影响 **/dev/random** 的熵池 。如果这些影响的负荷变得可测量，该参数可设置为 **0**。

max_sectors_kb

指定 I/O 请求的最大尺寸（以千字节计算），默认值为 **512** KB。该参数的最小值是由存储设备的逻辑块大小决定的。该参数的最大值是由 ***max_hw_sectors_kb*** 值决定的。

I/O 请求大于内部擦除块大小时，一些固态硬盘会表现不佳。在这种情况下，红帽推荐将 ***max_hw_sectors_kb*** 减少至内部擦除块大小。

nomerges

大多数工作负载受益于请求合并。然而，禁用合并有助于调试目的。可设置参数为 **0** 禁用合并。默认设置下为启用（设置为 **1**）。

nr_requests

限定同一时间排队的读和写请求的最大数量。默认值为 **128**，即在请求读或者写操作的下一个进程进入睡眠模式前有 128 个读请求和 128 个写请求排队。

对于延迟敏感应用程序，降低该参数值，并限制存储上的命令队列深度，这样回写 I/O 便无法填充有写请求的设备队列。设备队列填充时，其他尝试执行 I/O 操作的进程会进入睡眠模式，直到有可用队列空间。随后请求会以 round-robin fashion（循环方式）分配，以防止一个进程持续使用队列所有点。

optimal_io_size

一些存储设备用此参数报告最佳 I/O 大小。如果报告该值，红帽建议您尽可能将应用程序发出 I/O 与最佳 I/O 大小对齐，并是最佳 I/O 大小的倍数。

read_ahead_kb

定义操作系统在顺序读取操作阶段将预先读取的千字节数量，以便存储在页面缓存中可能马上需要的信息。设备映射程序经常受益于高的 ***read_ahead_kb*** 值 128 KB ；对于访问将要被映射的设备这是一个良好的起点。

旋转

一些固态硬盘不能正确公布其固态硬盘状态，并且会如传统旋转磁盘挂载。如果您的固态硬盘不能将它自动设置它为 **0**，那么请您手动设置，禁用调度器上不必要的搜寻减少逻辑。

rq_affinity

默认设置下，I/O 完成能在不同处理器上进行，而不是限定在发出 I/O 请求的处理器上。将***rq_affinity*** 设置为 **1** 以禁用此能力，并只在发出 I/O 请求的处理器上执行完成。这能提高处理器数据缓存的有效性。

5.3.7. 为性能配置文件系统

此章节包含红帽企业版 Linux 7 支持的每个文件系统的调整参数。用户格式化存储设备或者挂载格式化设备时，参数根据其值是否应当配置而分配。

如果文件碎片或者资源争用引起性能损失，性能通常可通过重新配置文件系统而提高性能。然而，在有些用例中，可能需要更改应用程序。这种情况下，红帽建议联系客户支持以获得帮助。

5.3.7.1. 调整 XFS

此章节包含对 XFS 文件系统格式化和挂载时可用的一些调整参数。

XFS 默认格式化和挂载设置适用于大多数工作负载。红帽建议只在更改特定配置会对您的工作负载有益时对它们进行更改。

5.3.7.1.1. 格式化选项

格式化选项的更多信息参见手册页：

```
$ man mkfs.xfs
```

目录块大小

目录块大小影响每个 I/O 操可检索或修改的目录信息数量。目录块大小最小值即文件系统块大小（默认设置下为4 KB）。目录块大小最大值为 **64** KB。

对于指定的目录块大小来说，大的目录比小的目录需要更多 I/O。因为和小目录块的系统相比，大目录块大小的系统每 I/O 操作会使用更多的处理能力。因此，根据您的工作负载，推荐使用尽可能小的目录和目录块大小。

如文件系统比大量写和大量读工作负载的列出项目数量少，红帽推荐使用以下目录块大小，请参见〈表 5.1 “为目录块大小推荐的最大目录项”〉

表 5.1. 为目录块大小推荐的最大目录项

目录块大小	最大项 （大量读操作）	最大项 （大量写操作）
4 KB	100000–200000	1000000–2000000
16 KB	100000–1000000	1000000–10000000
64 KB	>1000000	>10000000

在不同大小文件系统中，目录块大小对读和写工作负载的影响的情况请参见 XFS 文件。

使用 **mkfs.xfs -l** 选项配置目录块大小。请参见 **mkfs.xfs** 手册页。

分配组

分配组是独立的结构，指示自由空间并在文件系统中一节分配 inodes。只要同时操作影响不同分配组，每个分配组能被独立修改，这样 XFS 同时执行分配和解除分配操作。因此文件系统中执行的同时操作数量和分配组数量相等。然而，由于执行同时操作的能力受到能够执行操作的处理器数量的限制，红帽建议分配组数量应多于或者等于系统中处理器的数量。

多个分配组无法同时修改单独目录。因此，红帽推荐大量创建和移除文件的应用程序不要在单个目录中存储所有文件。

使用 **mkfs.xfs -d** 选项配置分配组，更多信息参见 **mkfs.xfs** 手册页。

增长约束

如您在格式化之后（通过增加更多硬件或通过自动精简配置），需要增加文件系统的大小，由于分配组大小在完成格式化之后不能更改，请务必仔细考虑初始文件布局。

必须根据文件系统最终能力，而非根据初始能力调节分配组大小。占据所有使用空间的文件系统中分配组数量不应超过数百，除非分配组处于最大尺寸 （1 TB）。因此，红帽向大部分文件系统推荐最大增长，允许文件系统是初始大小的十倍。

增长 RAID 数组的文件系统时，务必考虑额外护理，由于设备大小必须与固定多个分配组大小对齐，以便新分配组表头在新增加的存储中正确对齐。由于几何在格式化之后不能被更改，因此新存储也必须与已有的存储几何一致，因此，在同一个块设备上，不能优化不同几何的存储。

Inode 大小和内联属性

如果 inode 有足够可用空间，XFS 能直接将属性名称和值写入 inode。由于不需要额外的 I/O，这些内联属性能够被获取和修改，达到比获取单独的属性块更快的量级。

默认 inode 大小为 256 bytes。其中只有约 100 bytes 大小可用于属性存储，取决于 inode 上存储的数据范围指针数量。格式化文件系统时，增加 inode 大小能增加存储属性可用的空间数量。

属性名称和属性值两者都受到最大尺寸 254 bytes 的限制。如果名称或者值超过 254 bytes 长度，该属性会被推送到单独的属性快，而非存储在内联中。

使用 **mkfs.xfs -i** 选项配置 inode 参数，更多信息请参见 **mkfs.xfs** 手册页。

RAID

如果使用软件 RAID ，**mkfs.xfs** 会使用合适的带状单元和宽度自动配置底层的硬件。然而，如果使用硬件 RAID， 带状单元和宽度可能需要手动配置，这是因为不是所有硬件 RAID 设备输出此信息。使用 **mkfs.xfs -d** 选项配置带状单元和宽度。更多信息请参见 **mkfs.xfs** 手册页。

日志大小

直到同步事件被触发，待定的更改在内存中累计，这个时候它们会被写入日志。日志大小决定同时处于进行中的修改数量。它也决定在内存中能够累计的最大更改数量，因此决定记录的数据写入磁盘的频率。与大日志相比，小日志促使数据更频繁地回写入磁盘。然而，大日志使用更多内存来记录待定的修改，因此有限定内存的系统将不会从大日志获益。

日志与底层带状单元对齐时，日志表现更佳；换言之，它们起止于带状单元边界。使用 **mkfs.xfs -d** 选项将日志对齐带状单元，更多信息请参见 **mkfs.xfs** 手册页。

使用下列 **mkfs.xfs** 选项配置日志大小，用日志大小替换 *logsize* ：

```
# mkfs.xfs -l size=logsize
```

更多详细信息参见 **mkfs.xfs** 手册页：

```
$ man mkfs.xfs
```

日志带状单元

日志写操作起止于带状边界时（与底层带状单元对齐），存储设备上使用 RAID5 或 RAID6 布局的日志写操作可能会表现更佳。**mkfs.xfs** 尝试自动设置合适的日志带状单元，但这取决于输出该信息的 RAID 设备。

如果您的工作负载过于频繁地触发同步事件，设置大日志带状单元会降低性能。这是因为小的写操作需要填充至日志带状单元，而这会增加延迟。如果您的工作负载受到日志写操作延迟的约束，红帽推荐将日志带状单元设置为 1 个块，从而尽可能地触发非对齐日志写操作。

支持的最大日志带状单元为最大日志缓存的大小（256 KB）。因此底层存储器可能拥有更大的带状单元，且该带状单元能在日志中配置。在这种情况下，**mkfs.xfs** 会发出警告，并设置一个大小为 32 KB 的日志带状单元。

使用以下选项之一配置日志带状单元，其中 *N* 是被用于带状单元的块的数量，*size* 是以 KB 为单位的带状单元的大小。

```
mkfs.xfs -l sunit=Nb
mkfs.xfs -l su=size
```

更多详细信息参见 **mkfs.xfs** 手册页：

```
$ man mkfs.xfs
```

5.3.7.1.2. 挂载选项

Inode 分配

强烈推荐文件系统大于 1 TB。***inode64*** 参数配置 XFS，从而在文件系统中分配 inode 和数据。这样能保证 inode 不会被大量分配到文件系统的起始位置，数据也不会被大量分配到文件系统的结束位置，从而提高大文件系统的性能表现。

日志缓存和数量

日志缓存越大，将所有变更写入日志的 I/O 操作越少 。大日志缓存能提高有大量 I/O 密集型工作负

载的系统性能表现，而该工作负载没有非易变的写缓存。

通过 ***logbsize*** 挂载选项配置日志缓存大小，并确定日志缓存中信息存储的最大数量。如果未设置日志带状单元，缓存写操作可小于最大值，因此不需要减少大量同步工作负载中的日志缓存大小。默认的日志缓存大小为 32 KB。最大值为 256 KB， 也支持 64 KB、128 KB 或以 2 的倍数为幂的介于 32 KB 和 256 KB 之间的日志带状单元。

日志缓存的数量是由 ***logbufs*** 挂载选项确定的。日志缓存的默认值为 8 （最大值），但能配置的日志缓存最小值为2。通常不需要减少日志缓存的数量，除非内存受限的系统不能为额外的日志缓存分配内存。减少日志缓存的数量会降低日志的性能，尤其在工作负载对 I/O 延迟敏感的时候。

延迟变更日志

内存的变更写入日志前，XFS 的选项能集成这些改变。 ***delaylog*** 参数允许将频繁更改的元数据周期性地写入日志，而非每次改变都要记录到日志中。此选项会增加故障中操作丢失的潜在数量，也会增加用于跟踪元数据的内存大小。但是，它能通过按量级排序增加元数据的更改速度和可扩展性，为确保数据和元数据写入硬盘而使用 **fsync**、**fdatasync** 或**sync** 时，此选项不能减少数据或元数据的完整性。

5.3.7.2. 调整 ext4

本章节涵盖格式化和挂载时ext4 文件系统可用的一些调整参数。

5.3.7.2.1. 格式化选项

inode 表初始化

在很大的文件系统上初始化文件系统中所有 inode 会耗时很久。默认设置下会推迟初始化过程（启用迟缓 inode 表初始化）。但是，如果您的系统没有 ext4 驱动，默认设置下会禁用迟缓 inode 表初始化。可通过设置 ***lazy_itable_init*** 为 **1** 启用。那么在挂载后，内核进程继续初始化文件系统。

本章节仅描述了在格式化时可用的一些选项。更多格式化参数，参见**mkfs.ext4**手册页：

```
$ man mkfs.ext4
```

5.3.7.2.2. 挂载选项

inode表初始化率

启用迟缓 inode 表初始化时，您可通过指定 ***init_itable*** 参数值控制初始化发生的速率。执行后台初始化的时间约等于 1 除以该参数值。默认值为 **10**。

自动文件同步

对现有文件重命名、截断或重写后，一些应用程序无法正确执行 **fsync**。默认设置下，执行这些操作之后，ext4 会自动同步文件。但这样会比较耗时。

如果不需要此级别的同步，您可在挂载时通过指定 **noauto_da_alloc** 选项禁用该行为。如果 **noauto_da_alloc** 已设置，应用程序必须明确使用 fsync 以确保数据的持久化。

日志 I/O 优先级

默认设置下日志 I/O 优先级为 **3**，该值比常规 I/O 的优先级略高。您可在挂载时使用 ***journal_ioprio*** 参数控制日志 I/O 的优先级。***journal_ioprio*** 的有效值范围为从 **0** 到 **7**，其中 **0** 表示具有最高优先级的 I/O。

本章节仅描述挂载时可用的一些选项。更多选项，参见 **mount** 手册页：

```
$ man mount
```

5.3.7.3. 调整 btrfs

自红帽企业版 Linux 7.0 起，btrfs 作为技术预览而为用户提供。如果 btrfs 受到全面支持，本章节将在未来更新。

5.3.7.4. 调整 GFS2

本章节涵盖 GFS2 文件系统在格式化和挂载时可用的调整参数。

目录间距

GFS2 挂载点的顶层目录中创建的所有目录都是自动间隔，以减少目录中的碎片并提高写速度。为像顶层目录间隔其他目录，用 ***T*** 属性标注该目录，如示，用您想间隔该目录的路径替代 *dirname*。

```
# chattr +T dirname
```

chattr 作为 *e2fsprogs* 软件包中的一部份为用户提供。

减少争用

GFS2 使用全域锁机制，该机制需要簇中节点之间的通信。多节点之间的文件和目录争用会降低性能。通过最小化多节点间共享的文件系统区域，您可将缓存失效的风险最小化。

第 6 章 网络

网络子系统由很多敏感连接的不同部分构成。红帽企业版 Linux 7 网络因此旨在为大多数工作负载提供最佳性能，并且自动优化其性能。因此，不需要时常手动调节网络性能。本章探讨了可以对功能网络系统做的进一步优化。

网络性能问题有时是硬件故障或基础结构层故障造成的。解决这些问题超出了本文的范畴。

6.1. 注意事项

要决定调优，用户需要对红帽企业版 Linux 包的接收有充分的认识。本章节解释了如何接收和处理网络数据包，以及潜在瓶颈会出现的地方。

发送至红帽企业版 Linux 系统的数据包是由 NIC（网络接口卡）接收的，数据包置于内核硬件缓冲区或是循环缓冲区中。NIC 之后会发送一个硬件中断请求，促使生成一个软件中断操作来处理该中断请求。

作为软件中断操作的一部分，数据包会由缓冲区转移到网络堆栈。根据数据包及用户的网络配置，数据包之后会被转发、删除或转移到一个应用程序的 socket接收队列，并将从网络堆栈中删除。这一进程会持续进行，直到 NIC 硬件缓冲区中没有数据包或一定数量的数据包（在 **/proc/sys/net/core/dev_weight** 中指定）被转移。

6.1.1. 调节前

网络性能问题最常见的是由硬件故障或基础结构层故障造成的。红帽极力推荐在开始调节网络堆栈前核实硬件及基础结构层是否在按预期运行。

6.1.2. 数据包接收瓶颈

虽然网络堆栈基本上是自我优化的，但是在网络堆栈处理过程中有很多导致瓶颈且降低性能的问题。

NIC 硬件缓冲区或循环缓冲区

如果大量的数据包被弃置，硬件缓冲区就会成为瓶颈。要监控系统传送的数据包，请见 <第 6.2.4 节 “ethtool” >。

硬件或软件中断队列

中断会增加延迟，争用处理器。处理器如何处理中断，请见 <第 3.1.3 节 “中断请求管理” >。如何监控系统中断处理，请见 <第 3.2.3 节 “/proc/ 中断” >。影响中断处理的配置选项，请见 <第 3.3.7 节 “设置中断关联” >。

应用程序的 socket接收队列

应用程序接收队列的瓶颈是大量的数据包没有复制到请求应用程序中，或是 UDP 输入错误（**InErrors**）增加，此错误在 **/proc/net/snmp** 中。监控系统中的这些错误，请见 <第 6.2.1 节 “ss” > 和 <第 6.2.5 节 “/proc/net/snmp” >。

6.2. 监控和诊断性能问题

红帽企业版 Linux 7 提供大量有用的工具来监控系统性能和诊断与网络子系统有关的性能问题。本章对可用工具进行了概述并举例说明如何使用它们来监控和诊断与网络相关的性能问题。

6.2.1. ss

ss 是一个命令行实用程序，显示关于 socket的数据信息，允许管理员随时评估设备性能。**ss** 列表会默认打开没有注意到但已建立了连接的 TCP socket，但是会提供大量有用的选项来给管理员筛选特定的 socket数据。

红帽推荐在红帽企业版 Linux 7 中使用 **ss** 来替代 **netstat**。

ss 由 *iproute* 数据包提供。更多信息请见手册页：

```
$ man ss
```

6.2.2. ip

ip 实用程序允许管理员管理和监控线路、设备、路由策略及通道。**ip monitor** 指令可以持续监控设备、地址和线路的状况。

ip 由 *iproute* 数据包提供。使用 ip 详情请见手册页：

```
$ man ip
```

6.2.3. dropwatch

Dropwatch 是一个交互工具，用来监控和记录内核弃置的数据包。

更多信息，请见 **dropwatch** 手册页：

```
$ man dropwatch
```

6.2.4. ethtool

ethtool 实用程序允许管理员查看和编辑网络接口卡的设置。它有助于观察特定设备的数据，例如该设备弃置的数据包数量。

用户可以使用 **ethtool -S** 查看特定设备的计数状态和想要监控的设备名称。

```
$ ethtool -S devname
```

更多信息，请见手册页：

```
$ man ethtool
```

6.2.5. /proc/net/snmp

/proc/net/snmp 文件显示的数据是 snmp 用来代理监控和管理 IP、ICMP、TCP和 UDP 的。定期检查此文件可以协助管理员识别异常值，从而识别潜在的性能问题。例如，UDP 输入错误（**InErrors**）增加，且错误在**/proc/net/snmp** 中，就意味着 socket接收队列中出现了瓶颈。

6.2.6. 使用 SystemTap 监控网络

《*红帽企业版 Linux 7 SystemTap 初学者指南*》包含很多有用的示例脚本以分析和监控网络性能。

以下 **SystemTap** 示例脚本与网络有关，可能有助于诊断网络性能问题。默认安装在 **/usr/share/doc/systemtap-client/examples/network** 目录下。

nettop.stp

每 5 秒显示进程列表（进程标识符和指令），包括发送和接收的数据包，以及间隔时间里进程发送和接收的数据量。

socket-trace.stp

在 Linux 内核 **net/socket.c** 文件中检测每个功能，并显示跟踪数据。

tcp_connections.stp

显示每个系统接受的新传入的 TCP 连接信息。信息包括 UID 、接受该连接的指令、指令的进程标识符、连接端口和发起请求的 IP 地址。

dropwatch.stp

每 5 秒显示内核释放的 socket缓冲区数量。

《*红帽企业版 Linux 7 SystemTap 初学者指南*》可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

6.3. 配置工具

红帽企业版 Linux 提供大量工具来协助管理员配置系统。本章概述了可用的工具并举例说明如何使用它们来解决在红帽企业版 Linux 7 中与网络相关的性能问题。

但要记住的重要的一点是，网络性能问题有时是硬件故障或基础结构层故障造成的。红帽极力推荐在开始调节网络堆栈前先核实硬件及基础结构层在按预期运作。

此外，比起重新配置网络子系统来说，通过改变应用程序来解决一些网络性能问题显得较为容易。通常，配置应用程序来运行频繁的 posix 调用是个好办法，即使这意味着在应用程序空间排列数据，但这使得数据存储灵活且能按要求换入或换出内存。

6.3.1. 网络性能 Tuned-adm 配置文件

tuned-adm 对很多特定用例提供大量不同的配置文件以提高性能。以下配置文件有助于提高网络性能。

- 延迟性能
- 网络延迟
- 网络吞吐量

关于这些配置文件的更多信息，请见 <第 A.6 节 “tuned-adm” >。

6.3.2. 配置硬件缓冲区

如果硬件缓冲区弃置了大量的数据包，那么有很多可能的解决方法。

减缓输入流量

筛选传入的流量，减少加入的多播组数量或减少广播流量，以降低队列填充率。筛选传入流量的方法，详情请见 *Red Hat Enterprise Linux 7 Security Guide*。关于多播组详情，请见红帽企业版 Linux 7 聚类分析文档。关于广播流量详情，请见《*红帽企业版 Linux 7 系统管理员参考指南*》，或用户想要配置的设备相关文档。所有红帽企业版 Linux 7 的文档可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

调整硬件缓冲队列

通过增加队列的大小以减少传送的数据包数量，使其不易过剩。用户可以使用 ethtool 指令来更改网络设备的 rx/tx 参数：

```
# ethtool --set-ring devname value
```

改变队列的排出率

设备重量是指设备一次可以接收的数据包数量（单个预定处理器访问）。用户可以通过提高设备重量以增加队列的排出比率，这由 ***dev_weight*** 参数控制。此参数可以通过改变 **/proc/sys/net/core/dev_weight** 文件的内容来做临时更改，或使用 **sysctl** 进行永久更改，这由 *procps-ng* 数据包提供。

改变队列排出率通常是缓解不良网络性能最简单的方法。但是，增加设备一次可以接收的数据包数量会消耗处理器额外的时间，在此期间不可调度其他进程，因此可能会导致其他性能问题。

6.3.3. 配置中断队列

如果分析显示高延迟，系统可能受益于基于轮询的包接收，而不是基于中断的包接收。

6.3.3.1. 配置繁忙轮询

繁忙轮询有助于减少网络接收路径中的延迟， 使 socket 层代码查询网络设备的接收队列并禁用网络中断，这可以消除由于中断和由此产生的环境切换所造成的延误。但是，它会增加 CPU 的使用率。繁忙轮询可以防止 CPU 进入睡眠状态，睡眠状态会造成额外的功耗。

繁忙轮询是默认禁用的。要在特定 socket 中启用繁忙轮询，请按以下指示：

- 将 ***sysctl.net.core.busy_poll*** 设置为除 **0** 以外的值。这一参数控制的是 socket 轮询和选择位于等待设备队列中数据包的微秒数。红帽推荐值为 **50**。
- 添加 **SO_BUSY_POLL** socket 选项至 socket。

要全局启用繁忙轮询， 须将 ***sysctl.net.core.busy_read*** 设置为除了 **0** 以外的值。这一参数控制了 socket 读取位于等待设备队列中数据包的微秒数，且设置了 **SO_BUSY_POLL** 选项的默认值。红帽推荐在 socket 数量少时将值设置为 **50** ， socket 数量多时将值设置为 **100**。对于 socket 数量极大时（超过几百），请使用 **epoll**。

繁忙轮询由以下驱动程序支持。红帽企业版 Linux 7 支持这些驱动程序。

- bnx2x
- be2net
- ixgbe
- mlx4
- myri10ge

6.3.4. 配置 socket 接收队列

如果分析数据显示，数据包由于 socket 队列排出率太慢而被弃置，有几种方法来解决由此产生的性能问题。

减少传入流量的速度

减少队列填充速率可以通过筛选或弃置进入队列前的数据包，或是减少设备重量。

增加应用程序的 socket 队列深度

如果一个 socket 队列接收数量有限的涌入流量，增加 socket 的队列深度以便与涌入的流量大小相配，从而防止数据包被弃置。

6.3.4.1. 减少传入流量的速度

筛选传入流量或降低网络接口卡的设备重量来减少传入的流量。关于筛选传入流量的方法，详情请见 《*红帽企业版 Linux 7 安全指南*》，可从 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 中获取。

设备重量是指设备一次可以接收的数据包数量（单个预定处理器访问）。设备重量由 ***dev_weight*** 参数控制。此参数可以通过改变 **/proc/sys/net/core/dev_weight** 文件的内容来做临时更改，或是使用 **sysctl** 来做永久更改，这由 *procps-ng* 数据包提供。

6.3.4.2. 增加队列深度

增加应用程序的 socket 队列深度往往是提高 socket 队列排出率最简单的方法，但不可能是一个长期的解决方法。

要增加队列深度就要增加 socket 接收缓冲区的大小，可以做如下变化：

增加 /proc/sys/net/core/rmem_default 值

这一参数控制 socket 使用的接收缓冲区的默认大小，这个值必须小于 **/proc/sys/net/core/rmem_max** 的值。

使用 setsockopt 配置较大的 SO_RCVBUF 值

这一参数控制的是以字节为单位的 socket 接收缓冲区的最大值。使用 **getsockopt** 系统调用来确定当前缓冲区的值。此参数的更多信息，请见手册页：

```
$ man 7 socket
```

6.3.5. 配置 RSS

RSS（接收端调整），也叫多队列接收，是通过一些基于硬件的接收队列来分配网络接收进程，从而使入站网络流量可以由多个 CPU 进行处理。RSS 可以用来缓解接收中断进程中由于单个 CPU 过载而出现的瓶颈，并减少网络延迟。

要确定您的网络接口卡是否支持 RSS，须查看多个中断请求队列是否在 **/proc/interrupts** 中有相关的接口。例如，如果用户对 **p1p1** 接口有兴趣：

```
# egrep 'CPU|p1p1' /proc/interrupts
   CPU0    CPU1    CPU2    CPU3    CPU4    CPU5
89:   40187       0       0       0       0       0   IR-PCI-MSI-edge
p1p1-0
90:       0     790       0       0       0       0   IR-PCI-MSI-edge
p1p1-1
91:       0       0     959       0       0       0   IR-PCI-MSI-edge
p1p1-2
92:       0       0       0    3310       0       0   IR-PCI-MSI-edge
p1p1-3
93:       0       0       0       0     622       0   IR-PCI-MSI-edge
p1p1-4
94:       0       0       0       0       0    2475   IR-PCI-MSI-edge
p1p1-5
```

之前的输出显示 NIC 驱动程序为 **p1p1** 接口创建了 6 个接收队列（**p1p1-0** 至 **p1p1-5**）。也显示出每个队列处理的中断数量以及处理中断的 CPU。在这种情况下，由于有 6 个默认队列，这一特殊的 NIC 驱动程序就为每个 CPU 创建一个队列，这个系统一共有 6 个 CPU。这是 NIC 驱动程序中很常见的模式。

或者用户可以在网络驱动程序加载后查看 **ls -1 /sys/devices/*/*/*device_pci_address*/msi_irqs** 的输出。例如，如果用户对 PCI 地址为 **0000:01:00.0** 的设备感兴趣，可以通过以下指令来列出该设备中断请求队列：

```
# ls -1 /sys/devices/*/*/0000:01:00.0/msi_irqs
101
102
103
104
105
106
107
108
109
```

RSS 是默认启用的。RSS 的队列数（或是需要运行网络活动的 CPU ）会由适当的网络驱动程序来进行配置。 **bnx2x** 驱动程序是在 ***num_queues*** 中进行配置。**sfc** 驱动程序是用 ***rss_cpus*** 参数进行配置。通常，这都是在 **/sys/class/net/*device*/queues/*rx-queue*/** 中进行配置，其中 *device* 是网络设备的名称（比如 **eth1**），*rx-queue* 是适当的接收队列名称。

配置 RSS 时，红帽推荐限制每一个物理 CPU 内核的队列数量。超线程在分析工具中通常代表独立的内核，但是所有内核的配置队列，包括如超线程这样的逻辑内核尚未被证实对网络性能有益。

启用时，基于每个队列的 CPU 进程数量，RSS 在 CPU 间平等地分配网络进程。但是，用户可以使用 **ethtool *--show-rxfh-indir*** 和 ***--set-rxfh-indir*** 参数来更改网络活动的分配方式，并权衡哪种类型的网络活动更为重要。

irqbalance 后台程序可与 RSS 相结合，以减少跨节点内存及高速缓存行反弹的可能性。这降低了处理网络数据包的延迟。

6.3.6. 配置 RPS

RPS（接收端包控制）与 RSS类似，用于将数据包指派至特定的 CPU 进行处理。但是，RPS 是在软件级别上执行的，这有助于防止单个网络接口卡的软件队列成为网络流量中的瓶颈。

较之于基于硬件的 RSS ，RPS 有几个优点：

- RPS 可以用于任何网络接口卡。
- 易于添加软件过滤器至 RPS 来处理新的协议。
- RPS 不会增加网络设备的硬件中断率。但是会引起内处理器间的中断。

每个网络设备和接收队列都要配置 RPS，在 **/sys/class/net/*device*/queues/*rx-queue*/rps_cpus** 文件中，*device* 是网络设备的名称（比如 **eth0**），*rx-queue* 是适当的接收队列名称（例如 **rx-0**）。

rps_cpus 文件的默认值为 **0**。这会禁用 RPS，以便处理网络中断的 CPU 也能处理数据包。

要启用 RPS，配置适当的 **rps_cpus** 文件以及特定网络设备和接收队列中须处理数据包的 CPU 。

rps_cpus 文件使用以逗号隔开的 CPU 位图。因此，要让 CPU 在一个接口为接收队列处理中断，请将它们在位图里的位置值设为 1。例如，用 CPU 0、1、2 和 3 处理中断，将 **rps_cpus** 的值设为 **00001111**（1+2+4+8），或 **f**（十六进制的值为 15）。

对于单一传输队列的网络设备，配置 RPS 以在同一内存区使用 CPU 可获得最佳性能。在非 NUMA 的系统中，这意味着可以使用所有空闲的 CPU。如果网络中断率极高，排除处理网络中断的 CPU 也可以提高性能。

对于多队列的网络设备，配置 RPS 和 RSS 通常都不会有好处，因为 RSS 配置是默认将 CPU 映射至每个接收队列。但是，如果硬件队列比 CPU 少，RPS依然有用，并且配置 RPS 是来在同一内存区使用 CPU。

6.3.7. 配置 RFS

RFS（接收端流的控制）扩展了 RPS 的性能以增加 CPU 缓存命中率，以此减少网络延迟。RPS 仅基于队列长度转发数据包，RFS 使用 RPS 后端预测最合适的 CPU，之后会根据应用程序处理数据的位置来转发数据包。这增加了 CPU 的缓存效率。

RFS 是默认禁用的。要启用 RFS，用户须编辑两个文件：

/proc/sys/net/core/rps_sock_flow_entries

设置此文件至同时活跃连接数的最大预期值。对于中等服务器负载，推荐值为 **32768** 。所有输入的值四舍五入至最接近的2的幂。

/sys/class/net/*device*/queues/*rx-queue*/rps_flow_cnt

将 *device* 改为想要配置的网络设备名称（例如，**eth0**），将 *rx-queue* 改为想要配置的接收队列名称（例如，**rx-0**）。

将此文件的值设为 **rps_sock_flow_entries** 除以 **N**，其中 **N** 是设备中接收队列的数量。例如，如果 **rps_flow_entries** 设为 **32768**，并且有 16 个配置接收队列，那么 **rps_flow_cnt** 就应设为 **2048**。对于单一队列的设备，**rps_flow_cnt** 的值和 **rps_sock_flow_entries** 的值是一样的。

从单个发送程序接收的数据不会发送至多个 CPU。如果从单个发送程序接收的数据多过单个 CPU 可以处理的数量，须配置更大的帧数以减少中断数量，并以此减少 CPU 的处理工作量。或是考虑 NIC 卸载选项来获得更快的 CPU。

考虑使用 **numactl** 或 **taskset** 与 RFS 相结合，以将应用程序固定至特定的内核、 socket 或 NUMA 节点。这可以有助于防止数据处理紊乱。

6.3.8. 配置加速 RFS

加速 RFS 是通过添加硬件协助来增速的。如同 RFS，数据转发是基于应用程序处理数据包的位置。但不同于传统 RFS 的是，数据是直接发送至处理数据线程的本地 CPU：即运行应用程序的 CPU，或是对于在缓存层次结构中的 CPU 来说的一个本地 CPU。

加速 RFS 只有满足以下条件才可以使用：

- 网络接口卡须支持加速 RFS。加速 RFS 是由输出 **ndo_rx_flow_steer()** **netdevice** 功能的接口卡支持。
- **ntuple** 筛选必须启用。

一旦满足了这些条件，队列映射 CPU 就会基于传统 RFS 配置自动导出。即队列映射 CPU 会基于由每个接收队列的驱动程序配置的 IRQ 关联而自动导出。从 <第 6.3.7 节 “配置 RFS”> 中来获取配置传统 RFS 的信息。

红帽推荐在可以使用 RFS 以及网络接口卡支持硬件加速时使用加速 RFS 。

工具参考

此附录为红帽企业版 Linux 7 中多种工具提供快速参考，这些工具可用于调整性能。 工具完整、最新、详细参考资料请参见相关手册页。

A.1. irqbalance（中断平衡）

irqbalance 是一个命令行工具，在处理器中分配硬件中断以提高系统性能。默认设置下在后台程序运行，但只可通过 **--oneshot** 选项运行一次。

以下参数可用于提高性能。

--powerthresh

CPU 进入节能模式之前，设定可空闲的 CPU 数量。如果有大于阀值数量的 CPU 是大于一个标准的偏差，该差值低于平均软中断工作负载，以及没有 CPU 是大于一个标准偏差，且该偏差高出平均，并有多于一个的 irq 分配给它们，一个 CPU 将处于节能模式。在节能模式中，CPU 不是 irqbalance 的一部分，所以它在有必要时才会被唤醒。

--hintpolicy

决定如何解决 irq 内核关联提示。有效值为 **exact**（总是应用 irq 关联提示）、**subset**（irq 是平衡的，但分配的对象是关联提示的子集）、或者 **ignore**（irq 完全被忽略）。

--policyscript

通过设备路径、当作参数的irq号码以及 **irqbalance** 预期的零退出代码，定义脚本位置以执行每个中断请求。定义的脚本能指定零或多键值对来指导管理传递的 irq 中 **irqbalance**。

下列是为效键值对：

ban

有效值为 **true**（从平衡中排除传递的 irq）或 **false**（该 irq 表现平衡）。

balance_level

允许用户重写传递的 irq 平衡度。默认设置下，平衡度基于拥有 irq 设备的 PCI 设备种类。有效值为 **none**、**package**、**cache**、或 **core**。

numa_node

允许用户重写视作为本地传送 irq 的 NUMA 节点。如果本地节点的信息没有限定于 ACPI，则设备被视作与所有节点距离相等。有效值为识别特定 NUMA 节点的整数（从0开始）和 **-1**，规定 irq 应被视作与所有节点距离相等。

--banirq

将带有指定中断请求号码的中断添加至禁止中断的列表。

您也可以使用 ***IRQBALANCE_BANNED_CPUS*** 环境变量来指定被 **irqbalance** 忽略的 CPU 掩码。

更多信息，请参见手册页。

```
$ man irqbalance
```

A.2. Tuna

Tuna 使您能够控制处理器和调度关联。此章节包含命令行界面，但是也可使用有相同功能范围的图形界面。运行命令行 **tuna** 启动图形工具。

Tuna 接受多种按顺序处理的命令行参数。下列命令将负载分配到四个 socket中。

```
tuna --socket 0 --isolate \n   --thread my_real_time_app --move \n   --
irq serial --socket 1 --move \n   --irq eth* --socket 2 --spread \n   --
show_threads --show_irqs
```

--gui

打开图形用户界面。

--cpu

取用由 **Tuna** 控制的 CPU 逗号分隔列表。直到指定新列表前此列表均有效。

--config_file_apply

将配置文件名称应用于系统。

--config_file_list

列出预加载配置文件。

--cgroup

用于连接 ***--show_threads***。如果启用控制组，显示控制组类型，该控制组处理显示带有 ***--show_threads*** 所属于的控制组类型。

--affect_children

指定后，**Tuna** 影响子线程以及父线程。

--filter

过滤显示，只显示受影响的实体。

--isolate

取用 CPU 的逗号分隔列表。**Tuna** 从指定的 CPU 中迁移线程。

--include

取用 CPU 的逗号分隔列表，Tuna 允许所有线程在指定的 CPU 上运行。

--no_kthreads

指定此参数后，Tuna 不影响内核线程。

--move

将选择的实体移至指定的 CPU 中。

--priority

指定线程调度器策略和优先级。有效调度器策略为 **OTHER**、**FIFO**、 **RR**、**BATCH**、或者 **IDLE**。

当策略为 **FIFO** 或者 **RR**，有效的优先级值为从 1（最低）到 99（最高）的整数。默认值是 **1**。例如，**tuna--threads 7861 --priority=RR:40** 为线程 **7861** 设定了 **RR**（轮循）的策略和 **40** 的优先级。

当策略是 **OTHER**、**BATCH**、或者 **IDLE**，唯一有效优先级值为 **0**，它也是默认值。

--show_threads

显示线程列表。

--show_irqs

显示 irq 列表。

--irqs

取用受 **Tuna** 影响的 IRQ 逗号分隔列表 。直到指定新列表之前此列表均有效。使用 **+** 可将 IRQ 添加至列表，使用 **-** 可从列表中移除。

--save

将内核线程调度保存至指定文件。

--sockets

取用受 **Tuna** 控制的 CPU socket逗号分隔列表。该选项考虑了系统的拓扑结构，例如共享单一处理器缓存，且在同一个物理芯片上的核心。

--threads

取用受 **Tuna** 控制的线程逗号分隔列表。直到指定新列表之前此列表均有效。使用 **+** 可将线程添加至列表，**-** 可从列表中移除。

--no_uthreads

禁止影响用户线程的操作。

--what_is

更多帮助，请参见选定的实体。

--spread

平均分配 **--threads** 指定的线程至 **--cpus** 指定的 CPU。

A.3. ethtool

ethtool 工具允许管理员查看和编辑网络接口卡设置。这有助于观察某些设备的统计信息，比如被设备丢弃的数据包的数量。

手册页全面记录了 **ethtool** 的选项和使用。

```
$ man ethtool
```

A.4. ss

ss 是一个命令行工具，显示 socket的统计信息，允许管理员超时访问设备性能。默认设置下，**ss** 列出打开的非监听且已建立联系的 TCP socket，但向管理员提供一些为筛选掉特定 socket数据的有用选项。

ss -tmpie 是一个常用命令，显示所有 TCP socket（**t**、内部 TCP 信息（**i**）、 socket内存使用 （**m**）、使用 socket的进程 （**p**）、和详细的 socket信息（**i**）。

红帽企业版 Linux 7 中，与 **netstat** 相比， 红帽更推荐使用 **ss**。

ss 是由 *iproute* 软件包提供的。更多信息，请参见手册页。

```
$ man ss
```

A.5. tuned

Tuned 是一个调整的后台程序，在某种工作负载量下通过设置调整配置文件使操作系统有更好的性能表现。对其进行配置，使其对 CPU 和网络使用的变化做出响应，调整设置以提高激活设备的性能，并减少在未激活设备中的能耗。

在 **/etc/tuned/tuned-main.conf** 文件中编辑 ***dynamic_tuning*** 参数以配置动态调整行为。您也能在调整检查使用和更新调整细节之间，使用 ***update_interval*** 参数以秒为单位配置时间。

更多信息，请参见手册页。

```
$ man tuned
```

A.6. tuned-adm

tuned-adm 是一个命令行工具，提供一些不同配置文件以提高一些特定用例性能。它也提供一个评估系统和输出推荐的调整配置文件的子命令（**tuned-adm recommend**）。在您系统安装时它也能设置默认配置文件，以便能用于返回默认配置文件。

自红帽企业版 Linux 7 起，**tuned-adm** 有能力运行所有命令，这些命令是启用和禁用调整配置文件的一部分。这允许您添加 **tuned-adm** 中不可用的环境特定检测。例如在选择应用何种调整配置文件之前，检测系统是否是主数据库节点。

红帽企业版 Linux 7 在配置定义文件中提供 ***include*** 参数，允许您将自己的 **tuned-adm** 配置文件建立在存在的配置文件基础上。

以下调整配置文件是随 **tuned-adm** 一起提供的，并由红帽企业版 Linux 7 支持。

吞吐量性能

服务器配置文件的重点在于提高吞吐量。这是默认配置文件，并是为大多数系统推荐的。

通过设置 **intel_pstate** 和 **max_perf_pct=100**，与节约能耗相比，该配置文件更注重性能表现。它能启用透明大页面，使用 **cpupower** 来设置 **performance** CPU 频率管理器，并将输入/输出调度器设置为 **deadline**。它同样将 ***kernel.sched_min_granularity_ns*** 设置为 **10** μs，将 ***kernel.sched_wakeup_granularity_ns*** 设置为 **15** μs，以及将 ***vm.dirty_ratio*** 设置 **40**%。

延迟性能

服务器配置文件的重点在于降低延迟。该配置文件是为延迟敏感的工作负载所推荐的，其中工作负载会从 c- 状态调整和透明大页面增加的 TLB 高效性中获益。

通过设置 **intel_pstate** 和 **max_perf_pct=100**，与节约能耗相比，该配置文件更注重性能表现。它能启用透明大页面，使用 **cpupower** 来设置 **performance** CPU 频率管理器，并请求值为 **1** 的 ***cpu_dma_latency***。

网络延迟

服务器配置文件的重点在于降低网络延迟。

通过设置 **intel_pstate** 和 **max_perf_pct=100**，与节约能耗相比，该配置文件更注重性能表现。它禁用透明大页面以及自动 NUMA 平衡 。它使用 **cpupower** 来设置 **performance** CPU 频率管理器，并请求值为 **1** 的 ***cpu_dma_latency***。它同样将 ***busy_read*** 和 ***busy_poll*** 的时间设置为 **50** μ s，并将 ***tcp_fastopen*** 设置为 **3**。

网络吞吐量

服务器配置文件的重点在于提高网络吞吐量。

通过设置 **intel_pstate** 和 **max_perf_pct=100**，与节约能耗，该配置文件更注重性能表现。它能启用透明大页面，使用 **cpupower** 来设置 **performance** CPU 频率管理器，它同样将 ***kernel.sched_min_granularity_ns*** 设置为 **10** μ s，***kernel.sched_wakeup_granularity_ns*** 设置为 15 μ s，以及 ***vm.dirty_ratio*** 设置为 **40**%。

虚拟来宾

虚拟来宾是一个重点在于优化红帽企业版 Linux 7 虚拟机器性能的配置文件。

通过设置 **intel_pstate** 和 **max_perf_pct=100**，与节约能耗相比，该配置文件更注重性能表现。它降低了虚拟内存的交换。启用透明大页面，使用 **cpupower** 来设置 **performance** CPU 频率管理器。它也能将 ***kernel.sched_min_granularity_ns*** 设置为 **10** μ s，***kernel.sched_wakeup_granularity_ns*** 设置为 15 μ s，以及将 ***vm.dirty_ratio*** 设置为 **40**%。

虚拟-主机

虚拟主机是一个重点在于优化红帽企业版Linux 7虚拟主机的性能的配置文件。

通过设置 **intel_pstate** 和 **max_perf_pct=100**，相比节约能耗，该配置文件更注重性能表现。它降低了虚拟内存的交换。它能启用透明大页面，更频繁地重写脏页到磁盘。使用 **cpupower** 来设置 **performance** CPU 频率管理器，它将 ***kernel.sched_min_granularity_ns*** 设置为 **10** μ 秒，***kernel.sched_wakeup_granularity_ns*** 设置为 15 μ 秒，***kernel.sched_migration_cost*** 设置为 **5** μ 秒，以及 ***vm.dirty_ratio*** 设置为 **40**%。

配有 tuned-adm 的节能配置文件更多信息请参见《*红帽企业版 Linux 7 能耗管理指南*》，可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

使用 **tuned-adm** 的更多信息，请参见手册页。

```
$ man tuned-adm
```

A.7. perf

perf 提供一些有用的指令，此章节列出了其中一些指令。**perf** 的更多信息请参见《*红帽企业版 7 开发者指南*》，可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 或者参见手册页。

perf stat

此命令为常见性能事件提供整体数据，包括执行步骤和消耗所用的时间周期。您可使用选项标志来收集事件数据，而非默认测量事件。自红帽企业版 Linux 6.4 起，根据一个或多个特定控制组（c 组），可使用 **perf stat** 筛选监控。

更多信息请参见手册页：

```
$ man perf-stat
```

perf record

此命令将性能数据记录到随后可使用 **perf report** 分析的文件中。更多信息，请参见手册页。

```
$ man perf-record
```

perf report

此命令从文件中读取性能数据并分析记录数据，更多信息，请参见手册页。

```
$ man perf-report
```

perf list

此命令列出特定机器上有效事件。这些事件因系统性能监控硬件和软件配置而异。更多信息，请参见手册页。

```
$ man perf-list
```

perf top

此命令执行与 **top** 工具相似的功能。它实时生成并显示性能计数器配置文件。更多信息，请参见手册页。

```
$ man perf-top
```

perf trace

此命令执行与 **strace** 工具相似的功能。它监控特定线程或进程使用的系统调用以及该应用程序接收的所有信号。可获得其他的跟踪目标。请参见手册页以查看完整列表：

```
$ man perf-trace
```

A.8. PCP（性能协驾）

PCP 提供大量命令行工具、图形工具和库文件。此类工具更多信息请参见手册页：在命令行输入 **man *toolname***，用工具名称替代 *toolname*。

默认设置下，*pcp-doc* 软件包将详细文件安装至 **/usr/share/doc/pcp-doc** 目录。

A.9. vmstat

Vmstat 输出系统进程、内存、分页、块输入／输出、中断和 CPU 活动的报告。它提供自最近一次重启机器或者自前一次报告以来，这些事件平均值的瞬时报告。

-a

显示激活和未激活的内存。

-f

显示自重启以来 fork 数量。这包括 **fork**、**vfork**、和 **clone** 系统调用，且和创建的任务总数量

相同。根据线程使用，每个进程是由一个或多个任务代表的。显示不重复。

-m

显示 slab 信息。

-n

指定只会出现一次而非周期性出现的标题。

-s

显示多种事件计数器和内存数据的表格。显示不重复。

delay

报告之间有几秒的延迟。如果没有指定延迟，只显示一个报告，包括机器自上一次重启以来的平均值。

count

系统中报告的次数。如果没有指定 count（计数器），且定义了延迟，**vmstat** 无限期报告。

-d

显示磁盘统计。

-p

以分区名为值，为该分区报告详细统计。

-s

报告定义单位输出。有效值为 **k**（1000 字节）、**K**（1024 字节）、 **m**（1000000 字节）、或者 **M**（1048576 字节）。

每个输出模式提供的输出信息请参见手册页。

```
$ man vmstat
```

A.10. x86_energy_perf_policy

x86_energy_perf_policy 工具允许管理员定义性能和能效的相对重要性。由 *kernel-tools* 软件包提供。

查看当前策略，运行以下命令：

```
# x86_energy_perf_policy -r
```

设置新策略，运行以下命令：

```
# x86_energy_perf_policy profile_name
```

用以下配置文件的其中之一替代配置文件名。

performance

处理器不为节能而降低性能。这是默认值。

normal

处理器能容忍由潜在的显著节能所造成的微小性能下降。对于大多数服务器和桌面而言这是合理的节省。

powersave

处理器接受潜在的显著性能下降，以便充分利用能效。

如何使用 **x86_energy_perf_policy** 的更多信息，请参见手册页。

```
$ man x86_energy_perf_policy
```

A.11. turbostat

turbostat 工具提供系统处于不同状态所用时间的详细信息。 **Turbostat** 由 *kernel-tools* 软件包提供。

默认设置下，**turbostat** 为整个系统显示计数器结果的摘要，随后每隔五秒出现计数器结果，以下列标头：

pkg

处理器软件包编号。

core

处理器内核编号。

CPU

LinuxCPU（逻辑处理器）编号。

%c0

cpu 执行完毕的指令间隔百分比。

GHz

当 CPU 处于 c0 状态时，平均时钟速度。

TSC

整个间隔进程中的平均时钟速度。

%c1、 %c3、和 %c6

处理器分别在 c1、c3 或者 c6 状态下间隔百分比。

%pc3 或者 %pc6

处理器分别在 pc3 或者 pc6 状态下的间隔百分比。

使用 **-i** 选项指定计数器结果间的不同周期，例如：运行 **turbostat -i 10** 改为每10秒显示结果。

注意

即将推出的 Intel 处理器可能会添加其他 c 状态。自红帽企业版 Linux 7 起，**turbostat** 支持 c7、c8、c9 和 c10 状态。

A.12. numastat

numastat 由 *numactl* 软件包提供，并以每个 NUMA 节点为基础，为处理器和操作系统显示内存统计数据（例如分配时断时续）。 **numastat** 命令的默认跟踪类别如下所示：

numa_hit

成功分配至该节点的页面数量。

numa_miss

因预期节内存不足分配至该节点的页面数量。每个 **numa_miss** 事件在另一个节点上都有相应的 **numa_foreign** 事件。

numa_foreign

原本预期分配至此节点，而改为分配至其他节点的页面数量。**numa_foreign** 事件在另外节点上有一个相应的 **numa_miss** 事件。

interleave_hit

成功分配至该节点、交叉存取策略页面数量。

local_node

由节点上进程成功分配至该节点的页面数量。

other_node

由其他节点的进程分配至该节点的页面数量。

提供以下任一选项会改变按兆字节内存计算的显示单元（约两个小数位），且其他指定的 **numastat** 行为也会改变，描述如下：

-c

水平浓缩信息的显示表。这有助于含有大量 NUMA 节点的系统，在某种程度上列宽度和列内空间不可预测。使用该选项时，内存的数量四舍五入到最近的兆。

-m

根据单位节点，显示系统范围的内存使用信息，与 **/proc/meminfo** 中的信息类似。

-n

使用更新的格式、兆为度量单位，显示和如下原始 numastat 命令相同信息：（**numa_hit**、**numa_miss**、**numa_foreign**、**interleave_hit**、**local_node** 和 **other_node**）。

-p pattern

为指定模式显示单位节点内存信息。如果模式的值是由数字组成的，**numastat** 假定它是数字进程标识符。否则，**numastat** 从进程命令行查找指定的模式。

假定 **-p** 选项值后输入的命令行参数是附加模式，目的是将其过滤 。附加模式扩展，而非缩减过滤器。

-s

将显示的数据按降序排列，以便将最大的内存消耗者（根据所有列）列在首位。

您也可指定节点，这样表格将根据节点列分类。使用该选项时，节点值必须马上采用 **-s** 选项，具体如下：

```
numastat -s2
```

不要在选项和其值之间使用空格符。

-v

显示更多冗长的信息。即多进程的进程信息将显示每个进程的细节信息。

-V

显示 numastat 版本信息。

-z

从显示的信息中省略表中只为 0 值的行和列。请注意为了便于显示，一些四舍五入后接近 0 的值不会从显示输出中被省略。

A.13. numactl

Numactl 允许管理员使用指定的调度或内存放置策略来运行进程。**Numactl** 也能为共享的内存段或文件设置永久策略，以及进程的处理器关联和内存关联。

Numactl 提供许多实用的选项。此附录概述一部分选项，也为用户提供了一些使用建议，但并不详细。

--hardware

显示系统中的可用节点，且包含节点间的相对距离的详细目录。

--membind

确保内存只由指定节点分配。如果指定地点没有足够的可用内存，分配会失败。

--cpunodebind

确保指定命令及其子进程只在指定的节点上执行。

--phycpubind

确保指定的命令及其子进程只在指定的处理器上执行。

--localalloc

指明内存应当始终从本地节点分配。

--preferred

指定分配内存的优选节点。如果内存不能从指定的节点分配，其他的节点将被用于回退。

有关以上内容和其他参数的细节，参见手册页：

```
$ man numactl
```

A.14. numad

numad 是一个自动 NUMA 关联管理后台程序。为了动态提高 NUMA 资源分配和管理，它在系统内监控

NUMA 的拓扑结构和资源使用。

注意启用 **numad** 时 ，其行为将替代默认的自动 NUMA 平衡的行为。

A.14.1. 在命令行使用 numad

将 **numad** 作为执行表使用，只需运行：

```
# numad
```

numad 运行的时候，其活动被记录在 **/var/log/numad.log**。它会持续运行直到被以下命令终止：

```
# numad -i 0
```

终止 **numad** 不会移除它所做的提高 NUMA 关联的变更。如果系统使用有显著的变化，再次运行 **numad** 能调整关联来提高新条件下的性能。

如需将 **numad** 管理限制为特定进程，用下列选项来启动它：

```
# numad -S 0 -p pid
```

-p *pid*

该选项将指定的 *pid* 添加到显式的包含列表中。当指定的进程达到 **numad** 进程的显著门限值，指定的进程才会被管理。

-S 0

它将进程扫描的类型设置为 **0**，这会将 **numad** 管理限制到显式包含的进程。

有关可用的 **numad** 选项的详细信息，参见**numad**手册页：

```
$ man numad
```

A.14.2. 作为一个服务来使用 numad

将 **numad** 作为服务运行时，它尝试基于当前系统的工作负载来动态调整系统。其活动记录在 **/var/log/numad.log**。

如需启动服务，运行：

```
# systemctl start numad.service
```

如需重启后服务持久，运行：

```
# chkconfig numad on
```

有关可用的 **numad** 选项的详细信息，参见**numad**手册页：

```
$ man numad
```

A.14.3. 预安排建议

numad 提供多任务管理系统能够查询的预安排建议服务，从而为进程提供 CPU 的初始绑定和内存资源的协助。无论 **numad** 是否作为可执行或服务来运行，该预安排建议均可用。

A.14.4. 使用带 KSM 的 numad

如果 KSM 在 NUMA 系统中有所使用，为避免跨 NUMA 节点合并页，把 **/sys/kernel/mm/ksm/merge_nodes** 参数值改为 **0**。否则，由于跨 NUMA 节点页合并，KSM 会增加远程内存接入。此外，在大量的跨 NUMA 节点页合并后，内核的内存会计统计会最终否定彼此。因此，KSM 后台程序合并了很多内存页后，**numad** 会对可用内存正确数量和位置产生困惑。只有过度使用系统内存时，KSM 是有益的。如果系统有足够的空闲内存，可关闭或禁用 KSM 后台程序以提高性能表现。

A.15. OProfile

由 *oprofile* 软件包提供的 OProfile 是低开销的、系统范围的性能监控工具。它使用处理器上的性能监控硬件检索有关系统内核和执行表的信息。例如引用的内存、二级缓存请求的编号以及接收的硬件中断的编号。OProfile 也能配置运行在 Java 虚拟机（JVM）里的应用程序。

OProfile 提供以下工具。注意旧有的 **opcontrol** 工具和新的 **operf** 工具是互斥的。

ophelp

显示系统的处理器可用的事件，及其简要描述。

opimport

将样本数据库文件从异质的二进制格式转换为系统的本机格式。仅从不同的结构中分析样本数据库时使用该选项。

opannotate

如果应用和调试符号一起被编译，为执行表创建源码。

opcontrol

配置在分析运行中收集的数据。

operf

打算替换 **opcontrol**。**operf** 工具使用 Linux 性能事件子系统，它使您能够更准确地将配置文件作为单进程或系统范围设定目标，以及使得 OProfile 能与其他在您系统中使用性能监控的硬件工具更好的共存。不同于 **opcontrol**，无需初始设置，并且它能在没有 root 权限的情况下使用，除非使用了 **--system-wide** 选项。

opreport

检索配置文件数据。

oprofiled

以后台方式运行样本数据，将其周期性地写入硬盘。

旧有模式（**opcontrol**、**oprofiled** 和后期处理工具）仍然可用，但它不再是推荐的分析模式。

上述命令的更多信息，参见 OProfile 手册页：

```
$ man oprofile
```

A.16. taskset

A.16. taskset

taskset 工具是由 *util-linux* 软件包提供的。它允许管理员检索和设置正在运行的进程的处理器关联，或通过指定的处理器关联运行进程。

重要

taskset 不保证本地内存分配。如果您需要本地内存分配的额外性能收益，红帽推荐使用 **numactl** 来替代 taskset。

设置正在运行的进程的 CPU 关联，运行下列命令：

```
# taskset -c processors pid
```

使用处理器的逗号分隔列表或处理器的范围（例如，**1**、 **3**、 **5**－**7**）替换 *processors*。使用欲重新配置的进程的进程标识替换 *pid* 。

用指定的关联运行进程，运行下列命令：

```
# taskset -c processors -- application
```

用处理器的逗号分隔列表或处理器的范围替换 *processors*。使用您欲运行的应用程序的命令、选项和参数替换 *application*。

有关 **taskset** 的更多信息，请参见手册页：

```
$ man taskset
```

A.17. SystemTap

SystemTap 指南中有关于它的大量记载。红帽企业版 Linux 7 的《*SystemTap 初学者指南*》和《*SystemTap TapSet 参考*》版本都可在下列网站中查找 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

修订历史

修订 0.3-7.1 **Thu Nov 20 2014** **Chester Cheng**

说明：翻译完成。
翻译、校对：付莹莹、龚心星。
校对、编辑：任浩。
校对、责任编辑：鄭中。
附注：本简体中文版来自「**红帽工程部翻译中心**」与「**澳大利亚昆士兰大学笔译暨口译研究所**」之合作计划。若有疏漏之处，盼各方先进透过以下网址，给予支持指正：https://bugzilla.redhat.com/。

修订 0.3-7 **Thu Jun 26 2014** **劳拉·贝利 (Laura Bailey)**

更正 CPU 章节中的排版错误；感谢伊里·哈拉德奇。
移除 I/O 调度器的调整变更的参考文献；感谢伊里·哈拉德奇。

修订 0.3-5 **Wed Jun 11 2014** **劳拉·贝利 (Laura Bailey)**

为不能重新定向的 access.redhat.com 链接添加了尾斜杠。

修订 0.3-4 **Tue Jun 10 2014** **劳拉·贝利 (Laura Bailey)**

将中断和 CPU 禁止细节添加至 irqbalance 附录 BZ852981。

修订 0.3-3 **Mon Apr 07 2014** **劳拉·贝利 (Laura Bailey)**

为 RHEL7.0GA 重建。

修订 0.3-2 **Mon Apr 07 2014** **劳拉·贝利 (Laura Bailey)**

为 RT#294949 更新书的构架。

修订 0.2-38 **Mon Apr 07 2014** **劳拉·贝利 (Laura Bailey)**

添加更新的 OProfile 数据， BZ955882。
移除过期的评价。

修订 0.2-34 **Fri Apr 04 2014** **劳拉·贝利 (Laura Bailey)**

修正 lstopo 输出图像格式，BZ1042800。
添加 irqbalance 后台程序细节， BZ955890。
添加并修正控制组的细节，BZ794624。
添加 PCP 的细节，BZ955883。
更新 XFS 调整细节，BZ794616。
添加更新的 OProfile 数据， BZ955882。

修订 0.2-27 **Fri Mar 28 2014** **劳拉·贝利 (Laura Bailey)**

根据杰里米·埃德尔的反馈，修正 busy_poll 章节，RT276607。
根据杰里米·埃德尔的反馈，修正 nohz_full 章节并添加细节，RT284423。
在 SystemTap 章节中添加更多细节，BZ955884。
在 SSD 章节中添加更多细节，BZ955900。
添加更多 tuned-adm 推荐命令的细节，BZ794623。
修正在特性章节中自动 NUMA 平衡的注释，BZ794612。
修正一些术语问题和 NUMA 的示例输出问题，包括新图像，BZ1042800。
根据杰里米·埃德尔的反馈，修正 irqbalance 与 RSS 连接的细节。

修订 0.2-19 **Fri Mar 21 2014** **劳拉·贝利 (Laura Bailey)**

在内存章节中添加透明大页面的细节，BZ794621。
修正与 NUMA 节点有关的术语用法，BZ1042800。
更新系统内核限制，BZ955894。
起草无滴答系统内核章节，RT284423。
起草繁忙轮询章节，RT276607。
更新文件系统 barrier 的信息。
移除每个节点大页面分配的不清楚信息，为将来添加更有用的信息建立 BZ1079079。
添加固态硬盘的细节，BZ955900。
移除审阅标记。

修订 0.2-14 Thu Mar 13 2014 劳拉·贝利 (Laura Bailey)
应用杰里米·埃德尔和乔·马里奥的反馈。
记录从 BZ955872 到 Tuna GUI 的更新。
在网络章节和工具参考附录中添加 SystemTap 的细节，BZ955884。

修订 0.2-12 Fri Mar 07 2014 劳拉·贝利 (Laura Bailey)
记录为自动NUMA迁移的支持，BZ794612。
应用杰里米·埃德尔其他的反馈。

修订 0.2-11 Fri Mar 07 2014 劳拉·贝利 (Laura Bailey)
应用杰里米·埃德尔的反馈。

修订 0.2-10 Mon Feb 24 2014 劳拉·贝利 (Laura Bailey)
根据来自卢卡斯·切尔内 （BZ#794607）的反馈，修正 Ext4 信息。

修订 0.2-9 Mon Feb 17 2014 劳拉·贝利 (Laura Bailey)
根据比尔·盖伊的反馈修正 CPU 章节。
根据比尔·盖伊的反馈，在内存章节和工具参考附录中做出修正和添加。

修订 0.2-8 Mon Feb 10 2014 劳拉·贝利 (Laura Bailey)
在 CPU 章节中添加 isocpus 启动参数（RT276607）。
SME 反馈：修正参数描述和添加新参数（BZ#970844）。
在网络章节中添加推荐的 tuned-adm 配置文件。
添加评论来标记章节以审阅。

修订 0.2-4 Mon Feb 03 2014 劳拉·贝利 (Laura Bailey)
确认 **numactl --membind** 参数的文档有所描述（BZ#922070）。
在工具介绍、CPU 章节和工具参考附中中添加 Tuna 的细节，（BZ#970844）。
修正存储和文件系统章节中的结构错误。
添加缺失的交叉引用。

修订 0.2-2 Fri Jan 31 2014 劳拉·贝利 (Laura Bailey)
完成重新撰写和架构。
确保在此指南中提到的所有的工具列入了提供它们的软件包。

修订 0.1-11 Thu Dec 05 2013 劳拉·贝利 (Laura Bailey)
为 RHEL 7.0 Beta 建立重构的指南。

修订 0.1-10 Wed Nov 27 2013 劳拉·贝利 (Laura Bailey)
预测试客户版本。

修订 0.1-9 Tue Oct 15 2013 劳拉·贝利 (Laura Bailey)

基于客户反馈的细微修正（BZ#1011676）。

修订 0.1-7 **Mon Sep 09 2013** **劳拉 · 贝利 （Laura Bailey）**

合并 RHEL 6.5 的新内容。
应用编辑器反馈。

修订 0.1-6 **Wed May 29 2013** **劳拉 · 贝利 （Laura Bailey）**

更新 ext4 文件系统限制（BZ#794607）。
修正 64 位文件系统理论上限。
添加新特性部分来追踪性能相关的变化。
将默认 I/O 调度器从 **cfq** 变更到 **deadline**（BZ#794602）。
为 BTRFS 调整添加草拟内容（BZ#794604）。
更新 XFS 章节以提供目录块大小更清楚的推荐，并更新 XFS 支持的限制（BZ#794616）。

修订 0.1-2 **Thurs Jan 31 2013** **塔利亚 理查德森**

根据 RHEL7 草拟内容更新和发布。

修订 0.1-1 **Wed Jan 16 2013** **劳拉 · 贝利 （Laura Bailey）**

是此文件中 RHEL6.4 版本的分支。