Red Hat Enterprise Linux 7

性能调优指南

监控并优化 RHEL 7 中的子系统吞吐量

Last Updated: 2024-01-16

Red Hat Enterprise Linux 7 性能调优指南

监控并优化 RHEL 7 中的子系统吞吐量

Milan Navrátil
Red Hat Customer Content Services

Laura Bailey
Red Hat Customer Content Services

Charlie Boyle
Red Hat Customer Content Services

编辑

Marek Suchánek
Red Hat Customer Content Services
msuchane@redhat.com

法律通告

摘要

Red Hat Enterprise Linux 7 性能调优指南 解释了如何优化 Red Hat Enterprise Linux 7 性能。它还记录了 Red Hat Enterprise Linux 7 中与性能相关的升级。 性能调优指南 仅显示经现场测试和验证的步骤。然而，所有潜在配置都应该在测试环境中设置和测试，然后再应用到生产系统。也建议在调优前备份所有数据和配置设置。 为了扩展您的专业知识，您可能还对红帽企业 Linux 性能调优(RH442) 培训课程感兴趣。

目录

第 1 章 简介

要了解在每个 Red Hat Enterprise Linux 7 次发行版本中介绍的功能，请参阅相应次版本的发行说明。

性能调优指南是优化组成红帽企业 Linux 7 以满足特定用途的各种子系统的综合指南。本指南还概述了红帽企业 Linux 7 中提供的性能监控和调优工具。

在开始调整前，红帽有以下重要建议：

配置前进行备份

Red Hat Enterprise Linux 7 中的默认设置适用于在中等负载下运行的大多数服务。提高特定子系统的性能可能会给其他系统带来负面影响。在开始调整系统前，备份所有数据和配置信息。

从生产中测试配置

性能调优指南中记录的步骤由红帽工程师在实验室和字段中广泛测试。但是，红帽建议在安全测试环境中测试所有计划配置，然后再将这些配置应用到生产系统。

谁应该阅读此书

性能调优指南主要针对两个不同但重叠的受众编写：

系统管理员

性能调优指南详细介绍了每个配置选项的影响，以便系统管理员能够针对其特定目的优化红帽企业 Linux 7。本指南中的步骤适用于拥有红帽认证工程师(RHCE)证书或同等经验（3-5 年部署和管理基于 Linux 的系统）的系统管理员。

系统和业务分析师

本指南简要介绍了红帽企业 Linux 7 性能功能。它提供关于子系统在特定工作负载下如何执行的信息，使分析师能够确定红帽企业 Linux 7 是否适合其用例。

如有可能，《性能调优指南》还为读者参考更详细的功能文档。这让读者能够培养必要的深度知识，以建立基础架构和部署调整所需的详细部署和优化策略。

第 2 章 性能监控工具

本章简要描述了适用于 Red Hat Enterprise Linux 7 的一些性能监控和配置工具。在可能的情况下，本章指导读者进一步了解有关如何使用该工具的信息，以及工具可用于解决的实际生命周期情况示例。

以下知识库文章提供了适合与 Red Hat Enterprise Linux 一起使用的更加全面的性能监控工具列表：

2.1. /PROC

/proc "file system" 是一个目录，其中包含代表 Linux 内核当前状态的文件层次结构。它允许用户和应用程序查看系统的内核视图。

/proc 目录还包含有关系统硬件和任何当前运行的进程的信息。**/proc** 文件系统中的大多数文件都是只读的，但某些文件（通常 /proc/sys 中的这些文件）可以被用户和应用程序处理，以与内核通信配置更改。

有关在 **/proc** 目录中查看和编辑文件的详情，请参考 Red Hat Enterprise Linux 7 系统管理员指南 。

2.2. GNOME 系统监控器

GNOME 桌面环境包含一个图形工具 System Monitor，可帮助您监控和修改系统的行为。系统监控器显示基本系统信息，并允许您监控系统进程和资源或文件系统使用情况。

系统监控器有四个选项卡，各自显示系统的不同信息。

系统

此选项卡显示有关系统硬件和软件的基本信息。

进程

此选项卡显示活动进程的详细信息，以及这些进程之间的关系。可以过滤显示的进程，以便更轻松地查找某些进程。此选项卡还允许您对显示的进程执行一些操作，如启动、停止、终止和更改优先级。

资源

此选项卡显示当前的 CPU 时间使用情况、内存和交换空间使用情况以及网络使用情况。

文件系统

此选项卡列出所有已挂载文件系统，并提供有关每个文件系统的一些基本信息，如文件系统类型、挂载点和内存使用情况。

要启动系统监控器，请按 Super *键进入"活动概览"，键入"系统监控器"*，然后按 Enter 键。

有关系统监控器的更多信息，请参阅应用程序中的帮助菜单或 Red Hat Enterprise Linux 7 系统管理员指南。

2.3. 内置命令行工具

Red Hat Enterprise Linux 7 提供了多个工具，这些工具可用于从命令行监控您的系统，允许您在运行级别 5 之外监控您的系统。本章简要讨论各个工具，并提供有关每个工具应在何处使用及其使用方法的进一步信息的链接。

2.3.1. top

顶级工具由 procps-ng 软件包提供，可为正在运行的系统中的进程提供动态视图。它可以显示各种信息，包括系统摘要和当前由 Linux 内核管理的任务列表。它还具有操控进程以及使配置更改在系统重启后保留的有限能力。

默认情况下，显示的进程的排序会根据 CPU 使用率百分比进行排序，以便您可以轻松查看消耗最多资源的进程。top 显示的信息及其操作都高度可配置，因此您可以根据需要专注于不同的使用量统计。

有关使用 top 的详情请参考 man page：

```
$ man top
```

2.3.2. ps

ps 工具由 procps-ng 软件包提供，为所选活跃进程组生成快照。默认情况下，检查的组仅限于由当前用户拥有且与运行 ps 的终端相关联的进程。

PS 可以提供比 top 更详细的信息，但默认情况下，它提供了此数据的单个快照，按进程标识符排序。

有关使用 ps 的详情请参考 man page：

```
$ man ps
```

2.3.3. 虚拟内存统计信息(vmstat)

虚拟内存统计信息工具 vmstat 提供了有关系统进程、内存、分页、块输入/输出、中断和 CPU 活动的即时报告。vmstat 允许您设置一个抽样间隔，以便您可以近乎实时地观察系统活动。

vmstat 由 procps-ng 软件包提供。有关使用 vmstat 的详情，请查看 man page：

```
$ man vmstat
```

2.3.4. 系统活动报告器(sar)

系统活动报告器 sar 收集并报告有关目前为止发生的系统活动的信息。默认输出显示当天的 CPU 使用量从一天开始起 10 分钟（根据您的系统时钟划分为00:00:00）。

您还可以使用 **-i** 选项设置间隔时间（以秒为单位），例如： **sar -i 60** 告知 sar 每分钟检查 CPU 用量。

SAR 是手动创建使用 top 的系统活动的定期报告的有效替代方案。它由 sysstat 软件包提供。有关使用 sar 的详情请参考 man page:

```
$ man sar
```

2.4. PERF

perf 工具使用硬件性能计数器和内核追踪点来跟踪其他命令和应用程序对您的系统的影响。各种 perf 子命令显示和记录常见性能事件的统计信息，并对记录的数据进行分析和报告。

有关 perf 及其子命令的详细信息，请参阅 第 A.6 节 “perf”。

另外，红帽企业 Linux 7 开发人员指南中提供了更多信息。

2.5. TURBOSTAT

turbostat 由 kernel-tools 软件包提供。它在 Intel® 64 处理器中报告处理器拓扑、频率、空闲电源状态统计、温度和电源使用情况。

Turbostat 可用于识别在功耗或空闲时间方面效率低下的服务器。它还有助于确定系统上发生的系统管理中断(SMI)的速度。 它还可用于验证电源管理调整的影响。

turbostat 需要 root 特权才能运行。它还需要以下处理器支持：

- invariant 时间戳计数器
- APERF 模型特定寄存器
- MPERF 模型特定寄存器

有关 **turbostat** 输出以及如何读取它的详情，请参考 第 A.10 节 “turbostat”。

有关 **turbostat** 的更多信息，请参阅 man page：

```
$ man turbostat
```

2.6. IOSTAT

iostat 工具由 sysstat 软件包提供、监控和报告系统输入/输出设备加载，以帮助管理员决定如何在物理磁盘之间平衡输入/输出负载。**iostat** 工具报告自 **iostat** 上次运行或自启动以来对处理器或设备利用率进行报告。您可以使用 iostat(1) 手册页中定义的参数集中这些报告在特定设备上的输出。有关 **await** 值以及可能导致其值高的详细信息，请参阅以下红帽知识库文章： iostat 报告的值"await"的含义是什么？

2.7. IRQBALANCE

irqbalance 是一种命令行工具，可在处理器之间分发硬件中断以提高系统性能。有关 **irqbalance** 的详情，请查看 第 A.1 节 “irqbalance” 或 man page：

```
$ man irqbalance
```

2.8. SS

SS 是一个命令行实用程序，可打印有关套接字的统计信息，允许管理员评估设备性能。默认情况下，ss 列出已建立连接的打开的非侦听 TCP 套接字，但提供了多个有用的选项来帮助管理员过滤有关特定套接字的统计信息。

红帽建议在 Red Hat Enterprise Linux 7 中使用 ss over netstat。

一个常见的用法是 **ss -tmpie**，它显示使用套接字的 TCP 套接字、内存用量和进程的详细信息（包括内部信息）。

SS 由 iproute 软件包提供。如需更多信息，请参阅 man page:

```
$ man ss
```

2.9. NUMASTAT

numastat 工具根据每个 NUMA 节点显示进程和操作系统的内存统计信息。

默认情况下，**numastat** 显示每个节点的 NUMA 命中内核内存分配器中的系统统计信息。最佳性能由高 **numa_hit** 值和低 **numa_miss** 值表示。**numastat** 也提供了很多命令行选项，它可显示系统和进程内存如何在系统中的 NUMA 节点之间分布。

使用每个 CPU 顶**部** 输出跨节点 **numastat** 输出的交叉引用很有用，以验证进程线程是否在分配给内存的同一节点上运行。

numastat 由 numactl 软件包提供。有关如何使用 numastat 的详情请参考 第 A.11 节 “numastat”。有关 numastat 的详情请参考 man page:

```
$ man numastat
```

2.10. NUMAD

numad 是自动 NUMA 关联性管理守护进程。它监控系统中 NUMA 拓扑和资源使用情况，以便动态改进 NUMA 资源的分配和管理（以及系统性能）。根据系统工作负载，numad 可以提供高达 50% 的性能基准改进。它还提供预置建议服务，可供各种作业管理系统查询，为进程初始绑定 CPU 和内存资源提供帮助。

numad 通过定期访问 **/proc** 文件系统中的信息来基于每个节点监控可用的系统资源。它尝试维护指定的资源使用量水平，并通过在 NUMA 节点之间移动进程来重新平衡资源分配。numad 会尝试通过本地化并隔离系统 NUMA 节点子集上的重要进程来实现最佳 NUMA 性能。

枚举的主要优势是系统具有长时间运行的进程，占用了大量资源，并包含在总系统资源的子集中。它也可能有益于消耗多个 NUMA 节点需要的资源的应用程序；但是，当系统资源的消耗百分比增加时，数字提供的好处会降低。

当进程仅运行几分钟或者不消耗许多资源时，numad 可能无法提高性能。具有持续、不可预测的内存访问模式的系统（如大型内存数据库）也不太可能受益于使用 numad。

有关使用 numad 的详情，请查看 第 6.3.5 节 “使用 numad 自动 NUMA 关联性管理” 或 第 A.13 节 “numad” 或者参考 man page:

```
$ man numad
```

2.11. SYSTEMTAP

SystemTap 是一款跟踪和探测工具，可让您详细监控和分析操作系统活动，尤其是内核活动。它提供的信息类似于 top、ps、netstat 和 iostat 等工具的输出，但包含用于过滤和分析所收集数据的附加选项。

SystemTap 提供更深入、更精确的分析系统活动和应用行为，允许您查明系统和应用瓶颈。

有关 SystemTap 的详细信息，请参阅红帽企业 Linux 7 SystemTap 开头指南和 红帽企业 Linux 7 SystemTap Tapset 参考。

2.12. OPROFILE

OProfile 是一个整个系统的性能监控工具。它使用处理器的专用性能监控硬件来检索有关内核和系统可执行文件的信息，以确定某些事件的频率，如引用内存、二级缓存请求数以及收到的硬件请求数。OProfile 还可用于确定处理器使用情况，以及确定最常使用哪些应用程序和服务。

但是，OProfile 确实有一些限制：

- 性能监控示例可能不准确。由于处理器可能会不按顺序执行指令，因此可以从近邻指令（而非触发中断的指令）记录样本。
- OProfile 需要进程多次启动和停止。因此，来自多个运行的示例可以积累。您可能需要清除之前运行中的示例数据。
- OProfile 着重识别 CPU 访问权限限制的进程存在的问题。因此，它不适用于识别他们在其他事件上等待锁定时处于睡眠状态的进程。

有关 OProfile 的详情，请查看 第 A.14 节 “oprofile” 或 Red Hat Enterprise Linux 7 系统管理员指南 。或者，请参阅您的系统文档，位于 **/usr/share/doc/oprofile-*版本*** 中。

2.13. VALGRIND

Valgrind 提供大量检测和性能分析工具，以帮助提高应用程序的性能。这些工具可以检测内存和线程相关错误，以及堆、堆栈和阵列超量运行，使您可以轻松查找并更正应用代码中的错误。它们还可以对缓存、堆和分支预测进行性能分析，以识别可能提高应用速度并最大程度减少内存使用量的因素。

Valgrind 通过在综合 CPU 上运行并检测执行时的现有应用程序代码来分析您的应用。然后它会打印清楚应用程序执行涉及的每个进程到用户指定的文件、文件描述符或网络套接字的信息。请注意，执行检测代码的时间可能比正常执行的时间要长 4 到 50 倍。

Valgrind 可以按原样使用，无需重新编译。但是，由于 Valgrind 使用调试信息来查明代码中的问题，如果您的应用程序和支持库没有在启用了调试信息的情况下编译，红帽建议重新编译来包括此信息。

Valgrind 还与 GNU Project Debugger(gdb)集成以提高调试效率。

Valgrind 及其从属工具可用于内存分析。有关使用 Valgrind 分析系统内存的详情请参考 第 7.2.2 节 “使用 Valgrind 分析应用程序内存使用情况”。

有关 Valgrind 的详细信息，请参阅红帽企业 Linux 7 开发人员指南 。

有关使用 Valgrind 的详情，请查看 man page：

```
$ man valgrind
```

安装 valgrind 软件包时，也可在 **/usr/share/doc/valgrind-*版本*** 中找到附带的文档。

2.14. PQOS

pqos 工具包括在 intel-cmt-cat 软件包中，可让您监控和控制当前 Intel 处理器上的 CPU 缓存和内存带宽。您可以使用它来隔离工作负载，并提高多租户部署的性能确定性。

它从 Resource Director Technology(RDT)功能集中公开以下处理器功能：

监控

- 使用缓存监控技术(CMT)的最后一个缓存(LLC)使用情况和争用监控.
- 使用内存带宽监控(MBM)技术进行每线程内存带宽监控

分配

- 使用缓存分配技术(CAT)控制可用于特定线程和进程的 LLC 空间量.
- 使用代码和数据优先级(CDP)技术控制 LLC 中的代码和数据放置

使用以下命令列出系统上支持的 RDT 功能并显示当前的 RDT

```
# pqos --show --verbose
```

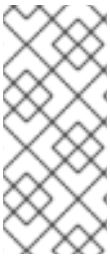
其它资源

- 有关使用 **pqos** 的更多信息，请参阅 pqos(8) man page。
- 有关 CMT、MBM、CAT 和 CDP 处理器功能的详细信息，请参阅官方 Intel 文档：Intel® 资源总监技术(Intel® RDT)。

第 3 章 TUNED

3.1. TUNED 概述

TuneD 是一个使用 **udev** 监控连接的设备和静态调整系统设置的守护进程，并根据所选的配置集动态调整系统设置。**TuneD 由很多预定义的配置集分发，用于常见用例，如高吞吐量、低延迟或节能。可以修改为每个配置集定义的规则，并自定义如何调优特定的设备。要恢复特定配置集对系统设置进行的所有更改，您可以切换到另一个配置集或取消激活 tuned 服务。**



注意

从 Red Hat Enterprise Linux 7.2 开始，您可以在 **no-daemon 模式下运行 Tuned**，这不需要任何常量内存。在这个模式中，**tuned** 应用设置并退出。在默认情况下，**no-daemon 模式** 被禁用，因为在这个模式中缺少很多 **tuned** 功能，包括 D-Bus 支持、热插支持或对设置进行回滚支持。要启用 **no-daemon 模式**，请在 **/etc/tuned/tuned-main.conf** 文件中设置以下内容： **daemon = 0**。

*静态*调优主要由预定义的 **sysctl** 和 **sysfs** 设置的应用程序组成，以及多个配置工具（如 **ethtool** ）的一次性激活。**TuneD** 还监控系统组件的使用，并根据监控信息动态调整系统设置。

动态调优考虑了在任何给定系统的正常运行时间内使用不同系统组件的方式。例如，硬盘驱动器在启动和登录期间大量使用，但在用户可能主要使用 Web 浏览器或电子邮件客户端等应用时很少使用。同样，CPU 和网络设备在不同的时间使用不同。**TuneD** 监控这些组件的活动，并对使用中的更改做出反应。

作为实际示例，请考虑典型的办公室工作站。大多数时候，以太网网络接口非常不活跃。只需几封电子邮件会随时进入和传出一次，或者可能会加载一些网页。对于这些类型的负载，网络接口不必像默认那样全速运行。TuneD 为网络设备有一个监控和调优插件，可检测此低活动，然后自动降低该接口的速度，通常是降低功耗。**如果接口的活动在较长时间内增加，例如，因为下载了 DVD 镜像或打开了带有大附件的电子邮件，tuned 会检测到这个信息，并设置接口速度的最大速度，以便在活动级别高时提供最佳性能。此原则也用于 CPU 和硬盘的其他插件。**

在 Red Hat Enterprise Linux 中全局禁用动态性能优化，可以通过编辑 **/etc/tuned/tuned-main.conf** 文件并将 **dynamic_tuning** 标志改为 **1** 来启用。

3.1.1. 插件

TuneD 使用两种类型的插件： *监控插件和 调优插件*。监控插件用于从正在运行的系统获取信息。目前，以下监控插件已被实现：

disk

获取每个设备的磁盘负载（IO 操作数）和测量间隔。

net

获取每个网卡的网络负载（传输数据包数）和测量间隔。

load

获得每个 CPU 的 CPU 负载和测量间隔。

监控插件的输出可通过调优插件以进行动态调优。目前实施了动态调优算法，尝试平衡性能和节能，因此在性能配置集中禁用（单个插件的动态性能优化可以在 **tuned** 配置集中启用或禁用）。每当启用的调优插件需要其指标时，监控插件会自动实例化。如果两个调优插件需要相同的数据，则仅创建一个监控插件实

例并共享数据。

每个调优插件对单个子系统进行调优，并获取从 **调优配置集** 填充的多个参数。每个子系统可以有多个设备（如多个 CPU 或网卡），它们由调优插件的各个实例处理。也支持单个设备的特定设置。提供的配置集使用通配符来匹配单个子系统的所有设备（有关如何更改此子系统的详情，请参阅 第 3.1.3 节 “自定义配置集”），它允许插件根据所需的目标（选择配置文件）以及用户需要做的唯一任务来调优这些子系统。

目前，以下调优插件已被实施（仅其中一部分实现动态调优，插件支持的参数也会列出）：

cpu

将 CPU 调控器设置为 ***governor*** 参数指定的值，并根据 CPU 负载动态更改 PM QoS CPU DMA 延迟。如果 CPU 负载小于 ***load_threshold*** 参数指定的值，则延迟被设置为 ***latency_high*** 参数指定的值，否则它会被设置为 ***latency_low*** 指定的值。此外，延迟也可以强制使用特定值而无需进一步动态更改。这可以通过将 ***force_latency*** 参数设置为所需的延迟值来实现。

eeepc_she

根据 CPU 负载动态设置 FSB 速度；此功能可在一些笔记本中找到，也称为 Asus Super 混合引擎。如果 CPU 负载较低或等于 ***load_threshold_powersave*** 参数指定的值，插件会将 FSB 速度设置为 ***she_powersave*** 参数指定的值（有关 FSB frequencies 和对应值的详情，请参阅内核文档，提供的默认值应该适用于大多数用户）。如果 CPU 负载较高或等于 ***load_threshold_normal*** 参数指定的值，它会将 FSB 速度设置为 ***she_normal*** 参数指定的值。不支持静态调优，如果未检测到对此功能的硬件支持，则插件将被透明地禁用。

net

将 wake-on-lan 配置为 ***wake_on_lan*** 参数指定的值（使用与 **ethtool** 工具相同的语法）。它还会根据接口使用率动态更改接口速度。

sysctl

设置由插件参数指定的各种 **sysctl** 设置。语法是 **name=value**，其中 **name** 与 **sysctl** 工具提供的名称相同。如果您需要更改其他插件未涵盖的设置（但在设置涵盖时首选特定插件），请使用此插件。

usb

将 USB 设备的自动暂停超时设置为 ***autosuspend*** 参数指定的值。值 0 表示 autosuspend 被禁用。

vm

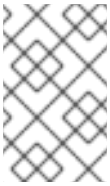
根据 ***transparent_hugepages*** 参数的布尔值，启用或禁用透明大内存页。

audio

将音频解码器的 autosuspend timeout 设置为 ***timeout*** 参数指定的值。目前支持 **snd_hda_intel** 和 **snd_ac97_codec**。值 **0** 表示自动暂停已被禁用。您还可以通过将布尔值参数 ***reset_controller*** 设置为 **true** 来强制实施控制器重置。

disk

将 elevator 设置为 ***elevator*** 参数指定的值。它还将 ALPM 设置为 ***alpm*** 参数 指定的值，将 ASPM 设置为 ***aspm*** 参数指定的值， 调度程序 quantum 到 ***scheduler_quantum*** 参数指定的值，磁盘 spindown 超时为 ***spindown*** 参数指定的值，磁盘 readahead 设置为 ***readahead*** 参数指定的值，并可以乘以 ***readahead_multiply*** 参数指定的当前磁盘 readahead 值。此外，这个插件会根据当前的驱动器使用率动态更改驱动器的高级电源管理和递减超时设置。动态调优可由布尔值参数 ***dynamic*** 控制，并默认启用。



注意

如果应用一个使用 **udev** 规则进行了配置的，则应用不同磁盘 readahead 值的 **tuned** 配置集会覆盖磁盘 readahead 值设置。红帽建议使用 **tuned** 工具来调整磁盘 readahead 值。

mounts

根据 ***disable_barriers*** 参数的布尔值启用或禁用挂载障碍。

script

此插件可用于执行在加载或卸载配置集时所运行的外部脚本。脚本由一个参数调用，该参数可以是 **start** 或 **stop**（它取决于在配置集加载或卸载过程中调用该脚本）。脚本文件名可以通过 ***script*** 参数指定。请注意，您需要在脚本中正确实施 stop 操作，并恢复您在启动操作过程中更改的所有设置，否则回滚将无法正常工作。为方便起见，默认安装 **功能** Bash 帮助程序脚本，并允许您导入和使用它中定义的各种功能。请注意，这个功能主要用于向后兼容，建议您将它用作最后的手段，如果它们涵盖所需的设置，首选使用其他插件。

sysfs

设置由插件参数指定的各种 **sysfs** 设置。语法是 **name=value**，其中 **name** 是要使用的 **sysfs** 路径。如果需要更改未被其他插件覆盖的一些设置，请使用此插件（如果它们涵盖所需的设置，则首选使用特定的插件）。

video

在显卡上设置不同的节能级别（目前只支持 Radeon 卡）。可以使用 ***radeon_powersave*** 参数指定节能级别。支持的值有： **default**,**auto**,**low**,**mid**,**high**, 和 **dynpm**。有关详细信息，请参阅：请注意，此插件是实验性的，将来的版本中可能会更改 参数。

bootloader

在内核引导命令行中添加参数。此插件支持传统的 GRUB 1、GRUB 2 以及具有可扩展固件接口(EFI)的 GRUB。grub2 配置文件的自定义非标准位置可以通过 ***grub2_cfg_file*** 选项指定。这些参数添加到当前的 grub 配置及其模板中。需要重新引导计算机才能使内核参数生效。

这些参数可使用以下语法指定：

```
cmdline=arg1 arg2 ... argn.
```

3.1.2. 安装和使用

要安装 tuned 软件包，以 root 用户身份运行以下命令：

```
yum install tuned
```

安装 tuned 软件包时，还会预设最适合您的系统的配置集。目前，默认配置集会根据以下可自定义规则选择：

throughput-performance

这在充当计算节点的红帽企业 Linux 7 操作系统上预先选中。此类系统上的目标就是最佳吞吐量性能。

virtual-guest

这在虚拟机上预先选择。目标是获得最佳的性能。如果您对最佳性能不感兴趣，您可能想将其更改为 **balanced** 或 **powersave** 配置集（请参阅 bellow）。

balanced

所有其他情况下均预先选择此项。目标是平衡性能和能耗。

要启动 **tuned**，请以 root 身份运行，使用以下命令：

```
systemctl start tuned
```

要使 **tuned** 在每次机器启动时都启动，请输入以下命令：

```
systemctl enable tuned
```

对于其他 **tuned** 控制，如选择配置集和其他，请使用：

```
tuned-adm
```

此命令需要运行 **tuned** 服务。

要查看可用的安装配置集，请运行：

```
tuned-adm list
```

要查看当前激活的配置集，请运行：

```
tuned-adm active
```

要选择或激活配置集，请运行：

```
tuned-adm profile profile
```

例如：

```
tuned-adm profile powersave
```

作为实验性功能，可以同时选择更多配置集。**tuned** 应用程序将尝试在负载期间合并它们。如果存在冲突，则来自最后一个指定的配置集的设置将优先。这会自动完成，且不会检查生成的参数组合是否有用。如果使用时不考虑，则该功能可能会以相反的方式调整某些参数，这可能会是反击的方法。例如，通过使用 **throughput-performance** 配置集为 **高吞吐量** 设置磁盘，并通过 **spindown-disk** 配置集 **同时将** 磁盘 spindown 设置为低值。以下示例优化了系统，使其在虚拟机中运行以获得最佳性能，并同时针对低功耗进行调整，而低功耗是优先级：

```
tuned-adm profile virtual-guest powersave
```

要让 **tuned** 为您系统的最佳配置集，而不更改任何现有配置集并使用在安装过程中使用的逻辑，请运行以下命令：

```
tuned-adm recommend
```

TuneD 本身具有额外的选项，您可以在手动运行时使用。不过，我们不建议这样做，主要用于调试目的。使用以下命令可以查看可用选项：

```
tuned --help
```

3.1.3. 自定义配置集

特定于分发的配置文件存储在 **/usr/lib/tuned/** 目录中。每个配置集都有自己的目录。该配置集由名为 **tuned.conf** 的主配置文件以及其他文件（如帮助程序脚本）组成。

如果您需要自定义配置集，请将配置集目录复制到用于自定义配置集的 **/etc/tuned/** 目录中。如果同一名称有两个配置集，则会使用 **/etc/tuned/** 中包含的配置集。

已被取消处理，您也可以在 **/etc/tuned/** 目录中创建自己的配置集，以使用 **/usr/lib/tuned/** 中包含的配置集，并只调整或覆盖某些参数。

tuned.conf 文件包含多个部分。有一个 **[main]** 部分。其他部分是插件实例的配置。所有部分都是可选的，包括 **[main]** 部分。以 hash 符号(#)开头的行是注释。

[main] 部分有以下选项：

include=*profile*

将包含指定的配置集，例如 **include=powersave** 将包含 **powersave** 配置集。

描述插件实例的部分以以下方式格式化：

```
[NAME]
type=TYPE
devices=DEVICES
```

NAME 是插件实例的名称，因为它在日志中使用。它可以是任意字符串。*TYPE* 是调优插件的类型。有关调优插件的列表和描述，请参阅 第 3.1.1 节 “插件”。*DEVICES* 是此插件实例将处理的设备的列表。**devices** 行可以包含列表、通配符 driver 和负效果(!)。您还可以组合规则。如果没有 **devices** 行，则插件实例将处理所有在 **TYPE** 系统中附加的所有设备。这与使用 **devices to**.如果没有指定插件的实例，则不会启用插件。如果插件支持更多选项，也可以在 plugin 部分中指定它们。如果未指定 选项，则将使用默认值（如果之前未在包含的插件中指定）。有关插件选项列表，请参考 第 3.1.1 节 “插件”。

例 3.1. 描述插件实例

以下示例将与从 **sd** 开始的所有内容（如 **sda** 或 **sdb** ）匹配，且不会禁用它们中的障碍：

```
[data_disk]
type=disk
devices=sd*
disable_barriers=false
```

以下示例将匹配除 **sda1** 和 **sda2** 以外的一切：

```
[data_disk]
type=disk
devices=!sda1, !sda2
disable_barriers=false
```

如果您不需要插件实例的自定义名称，并且配置文件中只有一个实例定义，Tuned 支持以下简短语法：

```
[TYPE]
devices=DEVICES
```

在这种情况下，可以省略 **type** 行。然后，实例将通过名称来引用，与类型相同。以上示例可改写为：

```
[disk]
devices=sdb*
disable_barriers=false
```

，如果使用相同的部分被指定多次，则设置将被合并。**如果由于冲突而无法合并，最后一个冲突的定义会覆盖之前在冲突中的设置。有时，您不知道之前定义的内容。在这种情况下，您可以使用 replace 布尔值选项并将其设置为 true。这将导致之前所有具有相同名称的定义被覆盖，合并也不会发生。**

您还可以通过指定 enabled=false 选项来禁用插件。这具有与从未定义实例相同的效果。如果您从 include 选项重新定义之前定义，且不想在自定义配置集中激活插件，则禁用插件会很有用。

以下是基于 balanced 配置集的自定义配置集，并将其扩展为 ALPM 的所有设备的 ALPM 设置为 maximal powersaving。

```
[main]
include=balanced

[disk]
alpm=min_power
```

以下是在内核引导命令行中添加 isolcpus=2 的自定义配置集示例：

```
[bootloader]
cmdline=isolcpus=2
```

应用配置集后，需要重新启动计算机才能使更改生效。

3.1.4. tuned-adm

对系统的详细分析可能非常耗时。Red Hat Enterprise Linux 7 包括了很多预定义配置集，用于典型的用例，您可以使用 tuned-adm 工具轻松激活。您还可以创建、修改和删除配置文件。

要列出所有可用的配置集并识别当前活跃的配置集，请运行：

```
tuned-adm list
```

要只显示当前活跃的配置集，请运行：

```
tuned-adm active
```

要切换到其中一个可用的配置集，请运行：

```
tuned-adm profile profile_name
```

例如：

```
tuned-adm profile latency-performance
```

禁用所有调整：

```
tuned-adm off
```

以下是典型用例的预定义配置集列表：

备注

下列配置文件可以通过基础软件包安装，也可能不会安装，具体取决于所使用的存储库文件：要查看系统上安装的 tuned 配置集，请以 root 用户身份运行以下命令：

```
tuned-adm list
```

要查看要安装的可用 调优配置集 列表，请以 root 用户身份运行以下命令：

```
yum search tuned-profiles
```

要在您的系统中安装 tuned 配置集，以 root 用户身份运行以下命令：

```
yum install tuned-profiles-profile-name
```

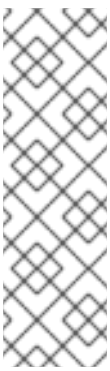
使用您要安装的配置集替换 ***profile-name***。

balanced

默认节能配置文件.它旨在成为性能和能耗之间的妥协。它尝试尽可能使用自动扩展和自动调整。大部分负载都有很好的结果。唯一缺点是增加了延迟。在当前的 tuned 发行版本中，它启用了 CPU、磁盘、音频和视频插件，并激活了 conservative 调控器。*radeon_powersave* 设置为 auto。

powersave

用于最大节能性能的配置文件。它可以限制性能，从而最大程度减少实际的功耗。在当前的 tuned 发行版本中，它为 SATA 主机适配器启用 USB 自动挂起、WiFi 节能和 ALPM 节能。它还为使用低折率的系统调度多核功耗，并激活 ondemand 监管器。它可节省 AC97 音频功率，或者根据您的系统节省 HDA-Intel 功耗，而超时为 10 秒。如果您的系统包含支持 Radeon 图形卡，并启用了 KMS，它会将其配置为自动省电。在 Asus Eee PC 上启用了动态 Super 混合引擎。



备注

powersave 配置集可能并非始终是最有效的。请考虑需要执行一系列明确的工作，例如需要转码的视频文件。如果转码在全功率上完成，则计算机可以消耗较少的能源，因为任务很快就会完成，计算机会开始闲置，并自动步入到非常高效的节能模式。另一方面，如果您用限流机器对文件进行转换，则计算机在转码期间消耗的功率会更少，但这个过程需要更长的时间，并且消耗的总消耗能源可能会更高。这就是为什么 balanced 配置文件通常是一个更好的选择。

throughput-performance

针对高吞吐量优化的服务器配置文件。它禁用节能机制并启用 sysctl 设置，以提高磁盘吞吐量性能、网络 IO 并切换到 截止时间调度程序。CPU 调控器设置为 performance。

latency-performance

针对低延迟而优化的服务器配置文件。它禁用节能机制，并启用可改善延迟的 sysctl 设置。CPU 调控器被设置为 performance，CPU 被锁定到低 C 状态（按 PM QoS）。

network-latency

低延迟网络调优配置文件。它基于 latency-performance 配置集。它还禁用透明大页、NUMA 平衡和调优其他几个网络相关的 sysctl 参数。

network-throughput

用于吞吐量网络调优的配置文件.它基于 throughput-performance 配置集。它还会增加内核网络缓冲区。

virtual-guest

专为红帽企业 Linux 7 虚拟机和 VMware 客户机设计的配置集，它基于企业存储配置文件，除了其他任务外，可减少虚拟内存交换性并增加磁盘预读值。它不禁用磁盘障碍。

virtual-host

基于 enterprise-storage 配置集（除其他任务）为虚拟主机设计的配置集降低了虚拟内存交换，增加磁盘预读值，并启用更积极的脏页面值。

oracle

根据 throughput-performance 配置集，为 Oracle 数据库负载进行了优化。它还禁用透明大内存页并修改其他一些性能相关的内核参数。这个配置集由 tuned-profiles-oracle 软件包提供。它可用于红帽企业 Linux 6.8 及更高版本。

desktop

根据 balanced 配置文件，为桌面进行了优化的配置集。它还支持调度程序自动组，以更好地响应交互式应用程序。

cpu-partitioning

cpu-partitioning 配置集将系统 CPU 划分为隔离和内务 CPU。为减少隔离 CPU 上的 jitter 和中断，配置集清除了与用户空间进程、可移动内核线程、中断处理程序和内核计时器隔离的 CPU。

内务 CPU 可以运行所有服务、shell 进程和内核线程。

您可以在 /etc/tuned/ cpu-partitioning -variables.conf 文件中配置 cpu-partitioning 配置集。配置选项为：

isolated_cores=*cpu-list*

列出要隔离的 CPU。隔离 CPU 的列表用逗号分开，用户可以指定范围。您可以使用短划线（如 3-5 ）指定范围。此选项是必需的。此列表中缺少的任何 CPU 都会自动被视为内务 CPU。

no_balance_cores=*cpu-list*

列出内核在系统范围范围的进程负载均衡期间没有考虑的 CPU。此选项是可选的。这通常与 isolated_cores 相同。

有关 cpu-partitioning 的详情，请查看 tuned-profiles-cpu-partitioning(7) man page。

注意

可能提供更多特定于产品或第三方调优配置集。此类配置文件通常由单独的 RPM 软件包提供。

可以使用 Optional 频道中提供的 tuned-profiles-compat 软件包安装其他预定义的配置集。这些配置集旨在向后兼容，不再开发。基本包中的常规配置文件大部分性能相同或更好。如果您没有使用它们的具体原因，请参阅基础包中上述配置文件。compat 配置集如下：

default

这对可用配置集的节能具有最低影响，仅启用 tuned 的 CPU 和磁盘插件。

desktop-powersave

面向桌面系统的省电配置文件.为 SATA 主机适配器 以及 tuned 的 CPU、以太网和磁盘插件启用 ALPM 节能。

laptop-ac-powersave

面向在 AC 上运行的笔记本电脑使用的中型节能配置文件.为 SATA 主机适配器、Wi-Fi 节能以及 tuned 的 CPU、以太网和磁盘插件启用 ALPM 节能。

laptop-battery-powersave

面向运行在电池上的笔记本电脑的高影响力节能配置文件.在当前的 tuned 实现中，它是 powersave 配置集的别名。

spindown-disk

适用于带有经典 HDD 的机器的节能配置文件，以最大化递减时间。它禁用 tuned 节能机制、禁用 USB 自动暂停、禁用蓝牙、启用 Wi-Fi 节能、禁用日志同步、增加磁盘回写时间并降低磁盘交换。使用 noatime 选项重新挂载所有分区。

enterprise-storage

面向企业级存储的服务器配置文件，最大化 I/O 吞吐量。它激活与 throughput-performance 配置集相同的设置，乘以 readahead 设置，并禁用非 root 和非引导分区的障碍。

备注

在物理机上使用 atomic-host 配置文件，以及虚拟机上的 atomic-guest 配置文件。

要为 Red Hat Enterprise Linux Atomic Host 启用 tuned 配置集，请安装 tuned-profiles-atomic 软件包。以 root 用户身份运行以下命令：

```
yum install tuned-profiles-atomic
```

Red Hat Enterprise Linux Atomic Host 的两个 tuned 配置文件是：

atomic-host

在用作裸机服务器上的主机系统时使用 throughput-performance 配置集时，针对红帽企业 Linux Atomic 主机进行优化的配置集。它还会增加 SELinux AVC 缓存、PID 限制和 netfilter 连接跟踪。

atomic-guest

在用作基于虚拟机配置文件的虚拟客户机系统时，针对红帽企业 Linux Atomic 主机优化的配置集。它还会增加 SELinux AVC 缓存、PID 限制和 netfilter 连接跟踪。

备注

可能还会有更多特定于产品的或第三方 调优配置集。这些配置文件通常由单独的 RPM 软件包提供。有三个 tuned 配置集可用于编辑内核命令行： realtime、realtime-virtual-host 和 realtime-virtual-guest。

要启用 realtime 配置集，请安装 tuned-profiles-realtime 软件包。以 root 用户身份运行以下命令：

```
yum install tuned-profiles-realtime
```

要启用 realtime-virtual-host 和 realtime-virtual-guest 配置集，请安装 tuned-profiles-nfv 软件包。以 root 用户身份运行以下命令：

```
yum install tuned-profiles-nfv
```

3.1.5. powertop2tuned

powertop2 tuned 工具是一个工具，可让您从 PowerTOP 建议创建自定义调优配置集。

要安装 powertop2tuned 应用程序，请以 root 用户身份运行以下命令：

```
yum install tuned-utils
```

要创建自定义配置集，以 root 用户身份运行以下命令：

```
powertop2tuned new_profile_name
```

默认情况下，它会在 /etc/tuned 目录中创建配置集，它基于当前选择的 tuned 配置集。为安全起见，所有 PowerTOP 调优最初在新配置集中禁用。使其取消注释 / etc/tuned/*配置集*/tuned.conf 中感兴趣的调整。您可以使用 --enable 或 -e 选项来生成新配置集，并启用了 PowerTOP 建议的大部分调整。USB 自动暂停等一些危险调优仍将被禁用。如果您真的需要它们，您必须手动取消注释。默认情况下，新配置集不会被激活。要激活它，请运行以下命令：

```
tuned-adm profile new_profile_name
```

如需选项 powertop2tuned 支持的完整列表，请输入以下命令：

```
powertop2tuned --help
```

3.2. 使用 TUNED 和 TUNED-ADM 进行性能调优

tuned 调优服务可以通过设置调优配置文件来调整操作系统，以便在特定工作负载下更好地执行。tuned-adm 命令行工具允许用户在不同的调优配置文件间切换。

tuned Profiles 概述

一些预定义的配置集包括在常见用例中，但 tuned 还允许您定义自定义配置集，该配置集可以基于预定义的配置集之一，或者从头开始定义。在 Red Hat Enterprise Linux 7 中，默认配置集是 throughput-performance。

tuned 提供的配置集分为两个类别：节能配置集和性能提升配置集。性能提升配置集包括侧重于以下方面的配置集：

- 存储和网络的低延迟
- 存储和网络的高吞吐量
- 虚拟机性能
- 虚拟化主机性能

调优的 Boot Loader 插件

您可以使用 tuned Bootloader 插件在内核 (boot 或 dracut)命令行中添加参数。请注意，仅支持 GRUB 2 引导装载程序，并且需要重新启动才能应用配置集更改。例如，要将 *quiet* 参数添加到 tuned 配置集中，请在 tuned.conf 文件中包括以下行：

```
[bootloader]
cmdline=quiet
```

切换到另一个配置集或手动停止 tuned 服务会删除附加参数。如果您关闭或重启系统，则内核参数会

在 grub.cfg 文件中保留。

环境变量和扩展调整的内置功能

如果您在更新 GRUB 2 配置后运行 tuned-adm profile *profile_name*，然后 grub2-mkconfig -o *profile_path*，您可以使用 Bash 环境变量（在运行 grub2-mkconfig 后扩展）。例如，以下环境变量被扩展到 nfsroot=/root ：

```
[bootloader]
cmdline="nfsroot=$HOME"
```

您可以使用 tuned 变量作为环境变量的替代选择。在以下示例中，${isolated_cores} 扩展至 1,2，因此内核使用 isolcpus=1,2 参数引导：

```
[variables]
isolated_cores=1,2

[bootloader]
cmdline=isolcpus=${isolated_cores}
```

在以下示例中，${non_isolated_cores} 扩展至 0,3-5，使用 0,3-5 参数调用 cpulist_invert 内置功能：

```
[variables]
non_isolated_cores=0,3-5

[bootloader]
cmdline=isolcpus=${f:cpulist_invert:${non_isolated_cores}}
```

cpulist_invert 函数反转 CPU 列表。对于 6-CPU 机器，inversion 为 1,2，内核使用 isolcpus=1,2 命令行参数引导。

使用 tuned 环境变量可减少所需输入的数量。您还可以将各种内置功能与 tuned 变量一起使用。如果内置功能无法满足您的需要，您可以在 Python 中创建自定义功能，并以插件的形式将它们添加到 tuned 中。在激活调优配置集时，变量和内置功能会在运行时扩展。

这些变量可以在单独的文件中指定。例如，您可以在 tuned.conf 中添加以下行：

```
[variables]
include=/etc/tuned/my-variables.conf
```

```
[bootloader]
cmdline=isolcpus=${isolated_cores}
```

如果您将 **isolated_cores=1,2** 添加到 **/etc/tuned/*my-variables.conf*** 文件中，则内核使用 **isolcpus=1,2** 参数引导。

修改默认系统调优配置集

修改默认系统 调优配置文件 的方法有两种。您可以创建新的调优配置集目录，或者复制系统配置文件的目录，并根据需要编辑配置集。

过程 3.1. 创建新调优配置集目录

1. **在 /etc/tuned/ 中，创建名为与您要创建的配置文件相同的新目录：/etc/tuned/*my_profile_name*/。**

2. **在新目录中，创建一个名为 tuned.conf 的文件，并在顶部包括以下行：**

```
[main]
include=profile_name
```

3. **包括您的配置集修改。例如，要使用 throughput-performance 配置集中的设置，其值为 vm.swappiness 设置为 5，而不是默认值 10，请包含以下行：**

```
[main]
include=throughput-performance

[sysctl]
vm.swappiness=5
```

4. **要激活配置集，请运行：**

```
# tuned-adm profile my_profile_name
```

使用新的 tuned.conf 文件创建目录后，您可以在系统 调优配置集 更新后保留所有配置集修改。

或者，使用系统配置文件将目录从 /user/lib/tuned/ 复制到 /etc/tuned/。例如：

```
# cp -r /usr/lib/tuned/throughput-performance /etc/tuned
```

-

然后，根据您的需要编辑 /etc/tuned 中的配置集。请注意，如果同一名称有两个配置集，则会加载位于 /etc/tuned/ 中的配置集。这种方法的缺点是，如果在 tuned 升级后更新系统配置文件，则更改不会反映在现已修改的版本中。

资源

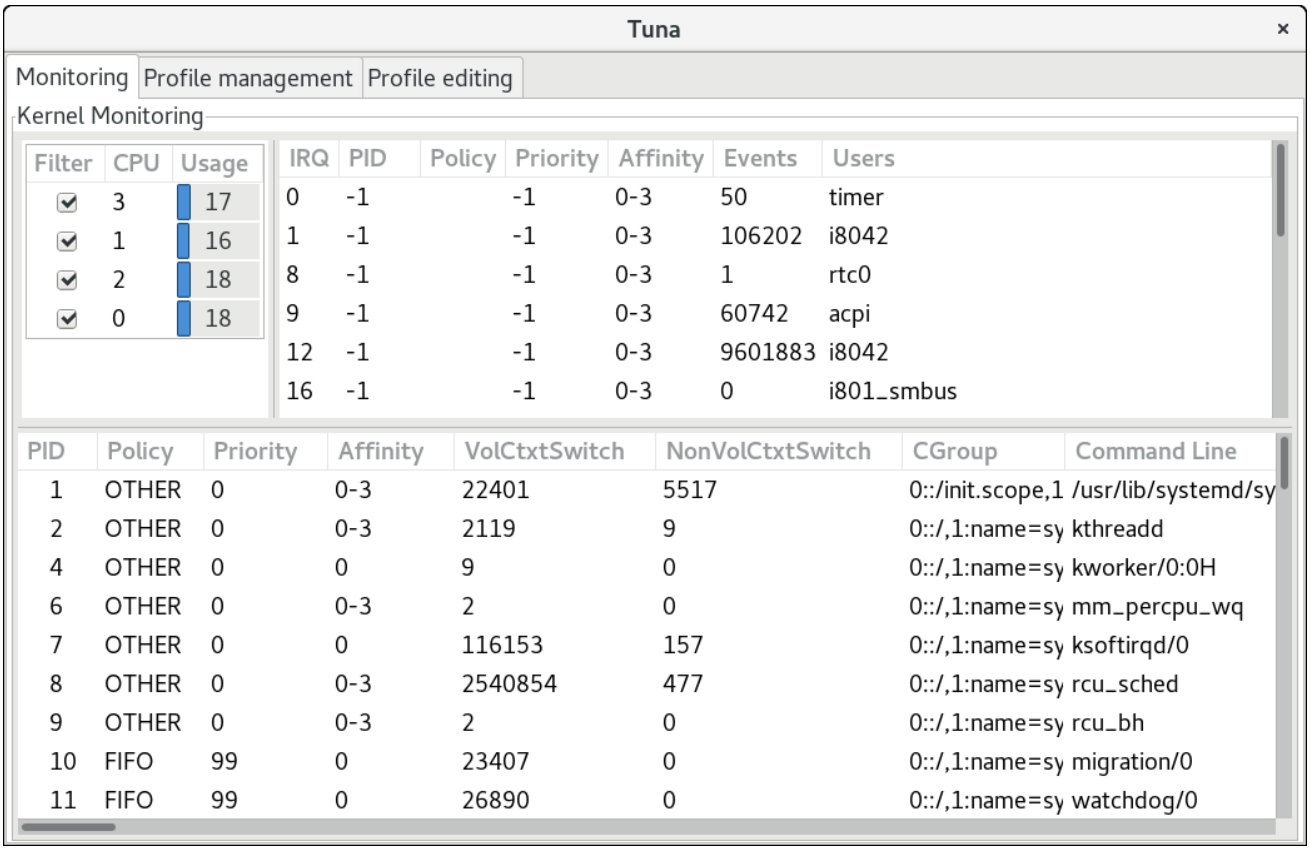
如需更多信息，请参阅 第 A.4 节 “tuned” 和 第 A.5 节 “tuned-adm”。有关使用 tuned 和 tuned-adm 的详情，请查看 tuned(8) 和 tuned-adm(1) 手册页。

第 4 章 TUNA

您可以使用 Tuna 工具调整调度程序可调项，调整线程优先级、RRQ 处理程序，以及隔离 CPU 内核和套接字。Tuna 旨在降低执行调优任务的复杂性。

安装 tuna 软件包后，使用 tuna 命令启动 Tuna 图形用户界面(GUI)。使用 tuna -h 命令显示可用的命令行界面(CLI)选项。请注意，tuna(8) 手册页区分操作和修饰符选项。

Tuna GUI 和 CLI 提供了等效的功能。GUI 在一个屏幕中显示 CPU 拓扑，以帮助您识别问题。Tuna GUI 还允许您更改正在运行的线程，并立即查看这些更改的结果。在 CLI 中，Tuna 接受多个命令行参数，并按顺序处理它们。您可以在应用程序初始化脚本中使用此类命令作为配置命令。



Tuna GUI 的监控选项卡

重要

使用 tuna --save=*filename* 命令和描述性文件名来保存当前配置。*请注意，这个命令不会保存* Tuna 可以更改的每个选项，而是只保存内核线程更改。任何在更改时当前未运行的进程都不会保存。

4.1. 使用 TUNA 查看系统

在进行任何更改之前，您可以使用 Tuna 显示系统上当前发生的情况。

要查看当前的策略和优先级，请使用 tuna --show_threads 命令：

```
# tuna --show_threads
      thread
pid   SCHED_ rtpri affinity             cmd
1     OTHER     0      0,1            init
2      FIFO    99        0     migration/0
3     OTHER     0        0     ksoftirqd/0
4      FIFO    99        0      watchdog/0
```

要只显示与 PID 对应的特定线程或与命令名称匹配，请在 --show_threads 前面添加 --threads 选项：

```
# tuna --threads=pid_or_cmd_list --show_threads
```

Thepid_or_cmd_list 参数是一个逗号分隔的 PID 或命令行模式的列表。

要查看当前的中断请求(IRQ)及其关联性，请使用 tuna --show_irqs 命令：

```
# tuna --show_irqs
# users           affinity
0 timer                 0
1 i8042                 0
7 parport0              0
```

要只显示与 IRQ 编号或与 IRQ 用户名对应的特定中断请求，请在 --show_irqs 前面添加 --irqs 选项：

```
# tuna --irqs=number_or_user_list --show_irqs
```

number_or_user_list 参数是一个以逗号分隔的 IRQ 编号或用户名模式的列表。

4.2. 使用 TUNA 调优 CPU

tuna 命令可以针对单个 CPU。要列出系统上的 CPU，请参阅 Tuna GUI 或 /proc/cpuinfo 文件中的

Monitoring 选项卡。

要指定要受您的命令影响的 CPU 列表，请使用：

```
# tuna --cpus=cpu_list --run=COMMAND
```

隔离 CPU 会导致当前在该 CPU 上运行的所有任务都移到下一个可用的 CPU。要隔离 CPU，请使用：

```
# tuna --cpus=cpu_list --isolate
```

包括 CPU 允许线程在指定的 CPU 上运行。要包括 CPU，请使用：

```
# tuna --cpus=cpu_list --include
```

***cpu_list* 参数是一个逗号分隔的 CPU 编号列表。例如，- -cpus=0,2。**

4.3. 使用 TUNA 调优 IRQ

要查看系统中当前运行的 IRQ 列表，请参阅 Tuna GUI 中的 Monitoring 选项卡或 /proc/interrpupts 文件。您还可以使用 tuna --show_irqs 命令。

要指定要受您的命令影响的 IRQ 列表，请使用 --irqs 参数：

```
# tuna --irqs=irq_list --run=COMMAND
```

要将中断移动到指定的 CPU 中，请使用 --move 参数：

```
# tuna --irqs=irq_list --cpus=cpu_list --move
```

***irq_list* 参数是一个逗号分隔的 IRQ 编号或用户名模式的列表。**

***cpu_list* 参数是一个逗号分隔的 CPU 编号列表。例如，- -cpus=0,2。**

例如，要以 sfc1 开头的所有中断为目标，并将它们分散到两个 CPU 上：

```
# tuna --irqs=sfc1\* --cpus=7,8 --move --spread
```

要验证您设置的更改，请使用 --show_irqs 参数，使用 --move 参数修改 IRQ：

```
# tuna --irqs=128 --show_irqs

  # users          affinity
128 iwlwifi        0,1,2,3

# tuna --irqs=128 --cpus=3 --move

# tuna --irqs=128 --show_irqs

  # users          affinity
128 iwlwifi              3
```

这样，您可以比较更改前后所选 IRQ 的状态。

注意

在某些情况下，使用 Tuna GUI 可能更为方便。通过指定要在其上运行的 CPU 来移动 IRQ 和线程可能会非常耗时且困难，因为它涉及创建 CPU 掩码的多个步骤。Tuna GUI 可自动化该进程.在 Tuna GUI 中，您还可以选择线程和 IRQ，并将它们拖到预期 CPU 上，这样可以更容易更改拓扑。

4.4. 使用 TUNA 调优任务

要更改线程的策略和优先级信息，请使用 --priority 参数：

```
# tuna --threads=pid_or_cmd_list --priority=[policy:]rt_priority
```

- *Thepid_or_cmd_list* 参数是一个逗号分隔的 PID 或命令行模式的列表。
- 将默认策略设置为 RR （用于轮询）、FIFO （第一个为 FIFO），首先为默认策略，或 OTHER 设置为 OTHER。

有关调度策略的概述请查看 第 6.3.6 节 “调优调度策略”。

- 在 1-99.1 范围中设置rt_priority，优先级最低，99 是最高优先级。

例如：

```
# tuna --threads=7861 --priority=RR:40
```

要验证您设置的更改，请在修改 **--priority** 参数前后使用 **--show_threads** 参数：

```
# tuna --threads=sshd --show_threads --priority=RR:40 --show_threads

              thread       ctxt_switches
  pid SCHED_ rtpri affinity voluntary nonvoluntary            cmd
 1034   OTHER     0  0,1,2,3         12            17            sshd
              thread       ctxt_switches
  pid SCHED_ rtpri affinity voluntary nonvoluntary            cmd
 1034      RR    40  0,1,2,3         12            17            sshd
```

这可让您比较更改前后所选线程的状态。

4.5. 使用 TUNA 的示例

例 4.1. 为特定 CPU 分配任务

以下示例使用有四个或更多处理器的系统，并演示了如何使所有 **ssh** 线程在 **CPU 0** 和 **1** 上运行，以及 **CPU 2** 和 **3** 中的所有 **http** 线程。

```
# tuna --cpus=0,1 --threads=ssh\* --move --cpus=2,3 --threads=http\* --move
```

以上示例命令按顺序执行以下操作：

1. 选择 **CPU 0** 和 **1.**
2. 选择以 **ssh** 开头的所有线程。

3. *将所选线程移到所选 CPU。tuna 设置线程的关联掩码，从 ssh 开始到适当的 CPU。CPU 数字化为 0 和 1，十六进制掩码为 0x3，或者在二进制中以 11 表示。*

4. *将 CPU 列表重置为 2 和 3。*

5. *选择以 http 开头的所有线程。*

6. *将所选线程移到所选 CPU。tuna 将以 http 开始的线程的关联掩码设置为适当的 CPU。CPU 数字化为 2 和 3，十六进制掩码为 0xC，或者在二进制 1100 中表示。*

例 4.2. 查看当前配置

以下示例使用 --show_threads(-P)参数显示当前的配置，然后测试请求的更改是否已如预期执行。

```
# tuna --threads=gnome-sc\* \
       --show_threads \
       --cpus=0 \
       --move \
       --show_threads \
       --cpus=1 \
       --move \
       --show_threads \
       --cpus=+0 \
       --move \
       --show_threads

                  thread       ctxt_switches
    pid SCHED_ rtpri affinity voluntary nonvoluntary            cmd
   3861   OTHER     0      0,1     33997           58 gnome-screensav
                  thread       ctxt_switches
    pid SCHED_ rtpri affinity voluntary nonvoluntary            cmd
   3861   OTHER     0        0     33997           58 gnome-screensav
                  thread       ctxt_switches
    pid SCHED_ rtpri affinity voluntary nonvoluntary            cmd
   3861   OTHER     0        1     33997           58 gnome-screensav
                  thread       ctxt_switches
    pid SCHED_ rtpri affinity voluntary nonvoluntary            cmd
   3861   OTHER     0      0,1     33997           58 gnome-screensav
```

以上示例命令按顺序执行以下操作：

1. *选择以 gnome-sc 开头的所有线程。*

2. *显示所选线程以启用用户验证其关联性掩码和 RT 优先级。*

3. *选择 CPU 0。*

4. *将 gnome-sc 线程移到所选 CPU (CPU 0)。*

5. *显示移动的结果。*

6. *将 CPU 列表重置为 CPU 1。*

7. *将 gnome-sc 线程移到所选 CPU (CPU 1)。*

8. *显示移动的结果。*

9. *将 CPU 0 添加到 CPU 列表中。*

10. *将 gnome-sc 线程移到所选 CPU (CPU 0 和 1)。*

11. *显示移动的结果。*

第 5 章 PERFORMANCE CO-PILOT(PCP)

5.1. PCP 概述和资源

Red Hat Enterprise Linux 7 支持 Performance Co-Pilot (PCP)、用于监控、可视化、存储和分析系统级性能测量的工具、服务和库。其轻量级分布式架构使其特别适合集中分析复杂系统。性能指标可以使用 Python、Perl、C++ 和 C 接口添加。分析工具可以直接使用客户端 API（Python、C++、C）和丰富的 Web 应用程序，使用 JSON 接口探索所有可用的性能数据。

pcp 允许：

- 监控和管理实时数据
- 日志记录和检索历史数据

您可以通过比较实时结果和存档的数据，使用历史数据分析模式与问题。

Performance Metric Collection Daemon (pmcd)负责收集主机系统上的性能数据，以及 pminfo 或 pmstat 等各种客户端工具可用于检索、显示、归档和通过网络处理此数据。pcp 软件包提供命令行工具和底层功能。图形工具需要 pcp-gui 软件包。

有关使用 PCP 发布的系统服务和工具列表，请参考 表 A.1 “在红帽企业 Linux 7 中与 Performance Co-Pilot 发布的系统服务” 和 表 A.2 “在红帽企业 Linux 7 中与 Performance Co-Pilot 发布的工具”。

资源

- 名为 PCPIntro 的 man page 充当 Performance Co-Pilot 的介绍。它提供了可用工具的列表，以及可用配置选项的描述以及相关 man page 的列表。默认情况下，全面的文档安装在 /usr/share/doc/pcp-doc/ 目录中，特别是 Performance Co-Pilot 用户和管理员指南 和 Performance Co-Pilot 程序指南。
- 有关 PCP 的详情，请查看红帽客户门户网站中的 Performance Co-Pilot(PCP)文章、解决方案、教程和白皮书。
- 如果您需要确定哪些 PCP 工具具有您熟悉的较旧工具的功能，请参阅 PCP 工具与旧工具红帽知识库文章的并排比较。

- 有关 Performance Co-Pilot 及其用法的深入描述，请参阅官方 PCP 文档。如果要在 Red Hat Enterprise Linux 上快速开始使用 PCP，请参阅 PCP 快速参考指南。PCP 官方网站还包含常见问题的列表。

5.2. 使用 PERFORMANCE CO-PILOT 对 XFS 文件系统性能分析

这部分论述了 PCP XFS 性能指标和使用方法。启动后，性能指标守护进程(PMCD)开始从已安装的 Performance Metric Domain Agents(PMDA)收集性能数据。PMDA 可以在系统上单独加载或卸载，并由同一主机上的 PMCD 控制。XFS PMDA 是默认 PCP 安装的一部分，用于在 PCP 中收集 XFS 文件系统的性能指标数据。

有关使用 PCP 发布的系统服务和工具列表，请参考 表 A.1 “在红帽企业 Linux 7 中与 Performance Co-Pilot 发布的系统服务” 和 表 A.2 “在红帽企业 Linux 7 中与 Performance Co-Pilot 发布的工具”。

5.2.1. 使用 PCP 安装 XFS PMDA 来收集 XFS 数据

XFS PMDA 作为 pcp 软件包的一部分提供，并在安装过程中默认启用。要安装 PCP，请输入：

```
# yum install pcp
```

要在安装 pcp 和 pcp-gui 软件包后在主机机器上启用并启动 PMDA 服务，请使用以下命令：

```
# systemctl enable pmcd.service
```

```
# systemctl start pmcd.service
```

要查询 PCP 环境以验证 PMCD 进程是否在主机上运行，且 XFS PMDA 在配置中被列为 enabled，请输入：

```
# pcp

Performance Co-Pilot configuration on workstation:

platform: Linux workstation 3.10.0-123.20.1.el7.x86_64 #1 SMP Thu Jan
29 18:05:33 UTC 2015 x86_64
hardware: 2 cpus, 2 disks, 1 node, 2048MB RAM
timezone: BST-1
```

```
services pmcd
pmcd: Version 3.10.6-1, 7 agents
pmda: root pmcd proc xfs linux mmv jbd2
```

手动安装 XFS PMDA

如果 PCP 配置读取中没有列出 XFS PMDA，请手动安装 PMDA 代理。PMDA 安装脚本会提示您指定 PMDA 角色：收集器、监控器或两者。

- *collector 角色允许收集当前系统上的性能指标*
- *monitor 角色允许系统监控本地系统、远程系统或两者。*

默认选项是 collector 和 monitor，它允许 XFS PMDA 在大多数场景中正常工作。

要手动安装 XFS PMDA，请切换到 xfs 目录：

```
# cd /var/lib/pcp/pmdas/xfs/
```

在 xfs 目录中，输入：

```
xfs]# ./Install

You will need to choose an appropriate configuration for install of
the "xfs" Performance Metrics Domain Agent (PMDA).

  collector     collect performance statistics on this system
  monitor       allow this system to monitor local and/or remote systems
  both          collector and monitor configuration for this system

Please enter c(ollector) or m(onitor) or (both) [b]
Updating the Performance Metrics Name Space (PMNS) ...
Terminate PMDA if already installed ...
Updating the PMCD control file, and notifying PMCD ...
Waiting for pmcd to terminate ...
Starting pmcd ...
Check xfs metrics have appeared ... 149 metrics and 149 values
```

5.2.2. 配置和检查 XFS 性能指标

使用 pminfo 检查指标

安装 PCP 并启用了 XFS PMDA 后，在 第 5.2.1 节 “使用 PCP 安装 XFS PMDA 来收集 XFS 数据” 中提供了相关的说明，开始查看 PCP 和 XFS 的性能指标是使用 pminfo 工具，该工具显示有关可用性能指标的信息。命令显示 XFS PMDA 提供的所有可用指标的列表。

显示 XFS PMDA 提供的所有可用指标列表：

```
# pminfo xfs
```

使用以下选项显示所选指标的信息：

-t 指标

显示描述所选指标的一行帮助信息。

-t 指标

显示描述所选指标的更详细帮助文本。

-f 指标

显示与指标对应的性能值的当前读取。

您可以将 -t、-T 和 -f 选项用于一组指标或单个指标。探测时，系统中的每个挂载的 XFS 文件系统都提供了大多数指标数据。

XFS 指标具有不同的组，使每个不同的组都是来自 root XFS 指标的新的叶节点，使用点(.)作为分隔符。leaf 节点语义(dots)适用于所有 PCP 指标。有关每个组中可用的指标类型的概述，请参阅 表 A.3 “XFS 的 PCP 指标组”。

另外，XFS 文档还包含有关监控 XFS 文件系统的部分： Chapter 13。XFS 监控.

例 5.1. 使用 pminfo 工具检查 XFS 读和写指标

显示描述 **xfs.write_bytes** 指标的一行帮助信息：

```
# pminfo -t xfs.write_bytes

xfs.write_bytes [number of bytes written in XFS file system write operations]
```

显示描述 **xfs.read_bytes** 指标的更多详细帮助文本：

```
# pminfo -T xfs.read_bytes

xfs.read_bytes
Help:
This is the number of bytes read via read(2) system calls to files in
XFS file systems. It can be used in conjunction with the read_calls
count to calculate the average size of the read operations to file in
XFS file systems.
```

获取与 **xfs.read_bytes** 指标对应的性能值的当前读取：

```
# pminfo -f xfs.read_bytes

xfs.read_bytes
    value 4891346238
```

使用 pmstore 配置指标

使用 **PCP**，您可以修改特定指标的值，特别是当指标充当控制变量时，如 **xfs.control.reset** 指标。要修改指标值，请使用 **pmstore** 工具。

例 5.2. 使用 pmstore 重置 xfs.control.reset Metric

本例演示了如何使用带有 **xfs.control.reset** 指标的 **pmstore**，将 **XFS PMDA** 的记录的计数器值重置为零。

```
$ pminfo -f xfs.write

xfs.write
    value 325262
```

```
# pmstore xfs.control.reset 1

xfs.control.reset old value=0 new value=1
```

```
$ pminfo -f xfs.write

xfs.write
    value 0
```

5.2.3. 检查每个文件系统中可用的 XFS 指标

从红帽企业 Linux 7.3 开始，PCP 可让 XFS PMDA 允许为每个挂载的 XFS 文件系统报告特定的 XFS 指标。这样更易于查明特定挂载的文件系统问题并评估性能。有关每个组群中每个文件系统中每个文件系统可用的指标类型的概述，请参阅 表 A.4 “每个设备的用于 XFS 的 PCP 指标组”。

例 5.3. 使用 pminfo 获取每个Device XFS 指标

pminfo 命令提供每个设备 XFS 指标，为每个挂载的 XFS 文件系统提供实例值。

```
# pminfo -f -t xfs.perdev.read xfs.perdev.write

xfs.perdev.read [number of XFS file system read operations]
inst [0 or "loop1"] value 0
inst [0 or "loop2"] value 0

xfs.perdev.write [number of XFS file system write operations]
inst [0 or "loop1"] value 86
inst [0 or "loop2"] value 0
```

5.2.4. 使用 pmlogger 记录性能数据

PCP 允许您记录性能指标值，这些值可以在以后重播，用于回顾性性能分析。使用 pmlogger 工具在系统上创建所选指标的存档日志。

使用 pmlogger 时，您可以指定在系统上记录哪些指标，以及记录的频率。默认 pmlogger 配置文件为 /var/lib/pcp/config/pmlogger/config.default。配置文件指定主日志记录实例记录哪些指标。

要使用 pmlogger 在本地机器上记录指标值，请启动一个主日志记录实例：

```
# systemctl start pmlogger.service
```

```
# systemctl enable pmlogger.service
```

当启用 pmlogger 并设置了默认配置文件时，PCP 配置中会包含 pmlogger 行：

```
# pcp

Performance Co-Pilot configuration on workstation:

platform: Linux workstation 3.10.0-123.20.1.el7.x86_64 #1 SMP Thu Jan
[...]
pmlogger: primary logger:/var/log/pcp/pmlogger/workstation/20160820.10.15
```

使用 pmlogconf 修改 pmlogger 配置文件

当 pmlogger 服务运行时，PCP 会记录主机上一组默认指标。您可以使用 pmlogconf 工具检查默认配置，并根据需要启用 XFS 日志记录组。启用的重要 XFS 组包括 XFS 信息、XFS 数据和 日志 I/O 流量组。

按照 pmlogconf 提示启用或禁用相关性能指标组，并控制每个启用的组的日志间隔。通过按 y （是）或 n (no)对提示符做出组选择。要使用 pmlogconf 创建或修改通用 PCP 归档日志记录器配置文件，请输入：

```
# pmlogconf -r /var/lib/pcp/config/pmlogger/config.default
```

手动修改 pmlogger 配置文件

您可以手动编辑 pmlogger 配置文件，并添加带有给定间隔的特定指标，以创建定制的日志记录配置。

例 5.4. 带有 XFS 指标的 pmlogger 配置文件

以下示例显示了 pmlogger config.default 文件的提取，并添加了一些特定的 XFS 指标。

```
# It is safe to make additions from here on ...
#

log mandatory on every 5 seconds {
    xfs.write
    xfs.write_bytes
    xfs.read
    xfs.read_bytes
}

log mandatory on every 10 seconds {
    xfs.allocs
    xfs.block_map
    xfs.transactions
    xfs.log

}

[access]
disallow * : all;
allow localhost : enquire;
```

取消 PCP 日志归档

记录指标数据后，您可以使用以下方法在系统中重放 PCP 日志存档：

- 您可以将日志导出到文本文件，并使用 pmdumptext、pmrep 或 pmlogsummary 等 PCP 实用程序将它们导入到电子表格中。

- 您可以重播 PCP Charts 应用程序中的数据，并使用图表来视觉化重新观察数据以及系统的实时数据。请参阅 第 5.2.5 节 “使用 PCP Charts 进行视觉追踪”。

您可以使用 pmdumptext 工具查看日志文件。使用 pmdumptext，您可以解析所选 PCP 日志归档并将值导出到 ASCII 表。pmdumptext 工具允许您转储整个归档日志，或者仅在命令行中指定单个指标从日志中选择指标值。

例 5.5. 显示特定的 XFS 指标日志信息

例如，显示存档中以 5 秒间隔收集的 xfs.perdev.log 指标的数据，并显示所有标头：

```
$ pmdumptext -t 5seconds -H -a 20170605 xfs.perdev.log.writes

Time local::xfs.perdev.log.writes["/dev/mapper/fedora-home"]
local::xfs.perdev.log.writes["/dev/mapper/fedora-root"]
? 0.000 0.000
none count / second count / second
Mon Jun 5 12:28:45 ? ?
Mon Jun 5 12:28:50 0.000 0.000
Mon Jun 5 12:28:55 0.200 0.200
Mon Jun 5 12:29:00 6.800 1.000
```

如需更多信息，请参阅 pmdumptext(1) man page，它包括在 pcp-doc 软件包中。

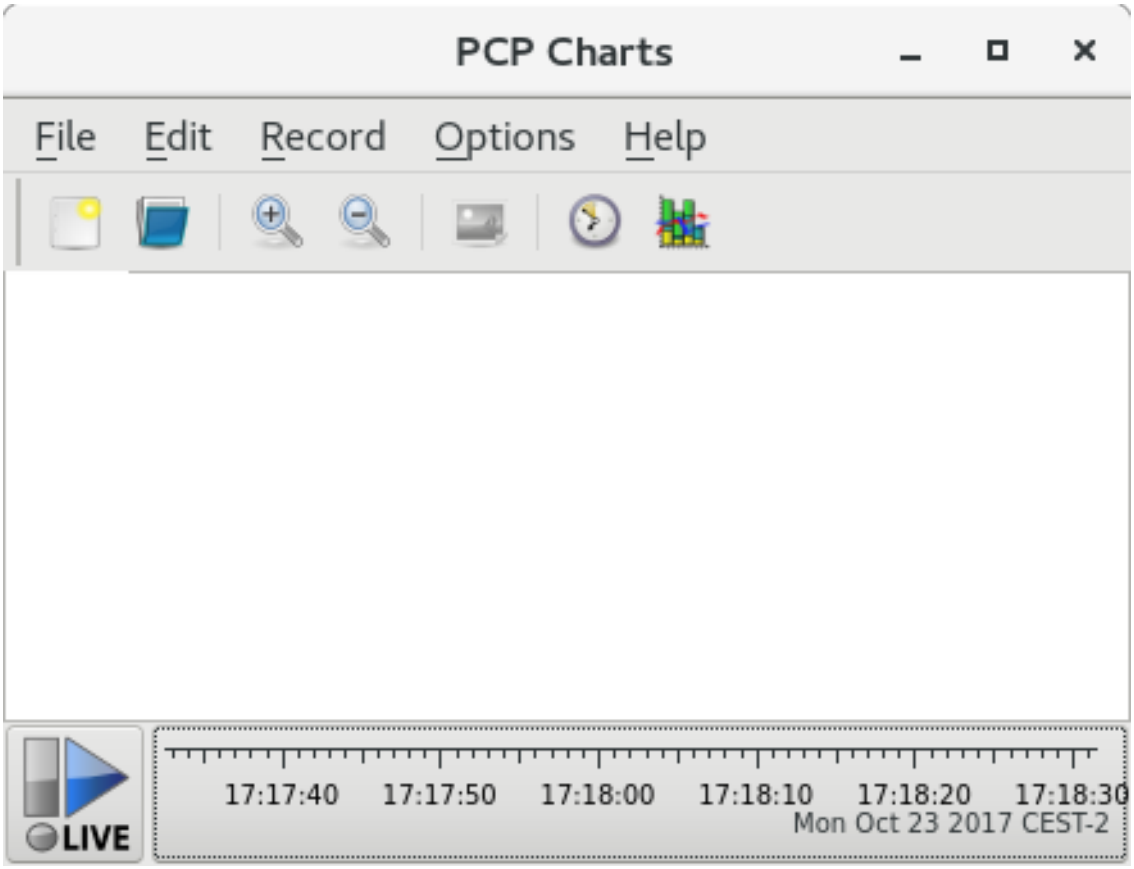
5.2.5. 使用 PCP Charts 进行视觉追踪

要使用图形 PCP Charts 应用程序，安装 pcp-gui 软件包：

```
# yum install pcp-gui
```

您可以使用 PCP Charts 应用将性能指标值绘制为图形。PCP Charts 应用允许同时显示多个图表。指标来自一个或多个实时主机，提供备选选项，以将 PCP 日志存档的指标数据用作历史数据源。要从命令行启动 PCP 图表，请使用 pmchart 命令。

启动 PCP 图表后会出现 GUI：



PCP Charts 应用程序

pmtime 服务器设置位于底部。可以使用 start 和 pause 按钮控制：

- *PCP 轮询指标数据的时间间隔*
- *历史数据指标的日期和时间*

前往 File → New Chart，通过指定主机名或地址来选择本地机器和远程机器的指标。然后，从远程主机选择性能指标。高级配置选项包括手动设置 chart 的轴值以及手动选择图表颜色的功能。

有多个选项可以获取镜像或记录 PCP Charts 中创建的视图：

- *点 File → Export 以保存当前视图的镜像。*
- *点 Record → Start 启动记录。点 Record → Stop 停止记录。停止记录后，记录的指标会*

被存档以供稍后查看。

您可以自定义 PCP Charts 接口，以多种方式显示性能指标中的数据，包括：

- 行图表
- 条形图
- 利用率图

在 PCP Charts 中，主配置文件称为 视图，允许保存与一个或多个 chart 关联的元数据。这个元数据描述了所有 Chart 方面，包括使用的指标和 chart 列。您可以创建自定义 视图 配置，点 File → Save View 保存它，并稍后加载 视图 配置。有关查看 配置文件 及其语法的详情请参考 pmchart(1) 手册页。

例 5.6. PCP Charts View 配置中的堆栈图表图

PCP Charts 视图配置文件示例描述了一个堆栈图，显示读取和写入到给定 XFS 文件系统 loop1 的字节数。

```
#kmchart
version 1

chart title "Filesystem Throughput /loop1" style stacking antialiasing off
    plot legend "Read rate"   metric xfs.read_bytes   instance  "loop1"
    plot legend "Write rate"  metric xfs.write_bytes  instance  "loop1"
```

5.3. 执行最小 PCP 设置以收集文件系统数据

以下流程提供了如何安装最小 PCP 设置以收集 Red Hat Enterprise Linux 中的统计信息的说明。最小

设置涉及在生产系统上添加收集数据以便进一步分析所需的最少软件包数量。

可以使用各种 PCP 工具（如 PCP Charts）分析 pmlogger 输出生成的 tar.gz 存档，并与其他性能信息源进行比较。

1. *安装 pcp 软件包：*

```
# yum install pcp
```

2. *启动 pmcd 服务：*

```
# systemctl start pmcd.service
```

3. *运行 pmlogconf 工具来更新 pmlogger 配置并启用 XFS 信息、XFS 数据和日志 I/O 流量组：*

```
# pmlogconf -r /var/lib/pcp/config/pmlogger/config.default
```

4. *启动 pmlogger 服务：*

```
# systemctl start pmlogger.service
```

5. *对 XFS 文件系统执行操作。*

6. *停止 pmlogger 服务：*

```
# systemctl stop pmcd.service
```

```
# systemctl stop pmlogger.service
```

7. **收集输出并将其保存到根据主机名和当前日期和时间命名的 tar.gz 文件中：**

```
# cd /var/log/pcp/pmlogger/
```

```
# tar -czf $(hostname).$(date +%F-%Hh%M).pcp.tar.gz $(hostname)
```

第 6 章 CPU

本章概述了影响 Red Hat Enterprise Linux 7 中的应用程序性能的 CPU 硬件详情和配置选项。第 6.1 节 “注意事项” 讨论影响性能的 CPU 相关因素。第 6.2 节 “监控和诊断性能问题” 教您如何使用红帽企业 Linux 7 工具诊断与 CPU 硬件或配置详情相关的性能问题。第 6.3 节 “配置建议” 讨论可用于解决 Red Hat Enterprise Linux 7 中 CPU 相关性能问题的工具和策略。

6.1. 注意事项

阅读本节以了解系统和应用程序性能如何受到以下因素的影响：

- 处理器互相连接的方式以及内存等相关资源。
- 处理器如何调度线程来执行.
- 处理器如何在 Red Hat Enterprise Linux 7 中处理中断.

6.1.1. 系统拓扑

在现代计算中，中央处理器的理念令人困惑，因为大多数现代系统都有多个处理器。这些处理器如何互相连接和其他系统资源（系统的拓扑），可能会对系统和应用程序性能以及系统调优注意事项产生巨大影响。

现代计算中使用两种主要拓扑类型：

对称多处理器(SMP)拓扑

SMP 拓扑允许所有处理器在相同的时间里访问内存。但是，由于共享和相等内存访问本质上会强制从所有 CPU 进行序列化内存访问，因此 SMP 系统扩展限制现在通常被视为不可接受。因此，实际上，所有现代服务器系统都是 NUMA 计算机。

非一致性内存访问(NUMA)拓扑

NUMA 拓扑的开发时间比 SMP 拓扑更近。在 NUMA 系统中，一个套接字上对多个处理器进行物理分组。每个套接字都有一个专用的内存区域，以及对该内存具有本地访问权限的处理器将统称为节点。

位于同一节点上的处理器能够快速访问该节点的内存数据库，而且对不在节点的内存库的访问速度也较慢。因此，访问非本地内存会降低性能。

鉴于这种性能损失，具有 NUMA 拓扑的系统上对性能敏感的应用程序应该访问与执行应用程序的处理器位于同一节点上的内存，并尽可能避免访问远程内存。

在具有 NUMA 拓扑的系统上调优应用程序性能时，务必要考虑应用的执行位置，以及哪一个内存银行最接近执行点。

在具有 NUMA 拓扑的系统上，/sys 文件系统包含有关处理器、内存和外设设备如何连接的信息。/sys/devices/system/cpu 目录包含有关系统中处理器如何相互连接的详细信息。/sys/devices/system/node 目录包含有关系统中 NUMA 节点的信息，以及这些节点之间的相对距离。

6.1.1.1. 确定系统拓扑

有许多命令可帮助您了解系统的拓扑。numactl --hardware 命令提供了系统拓扑的概述。

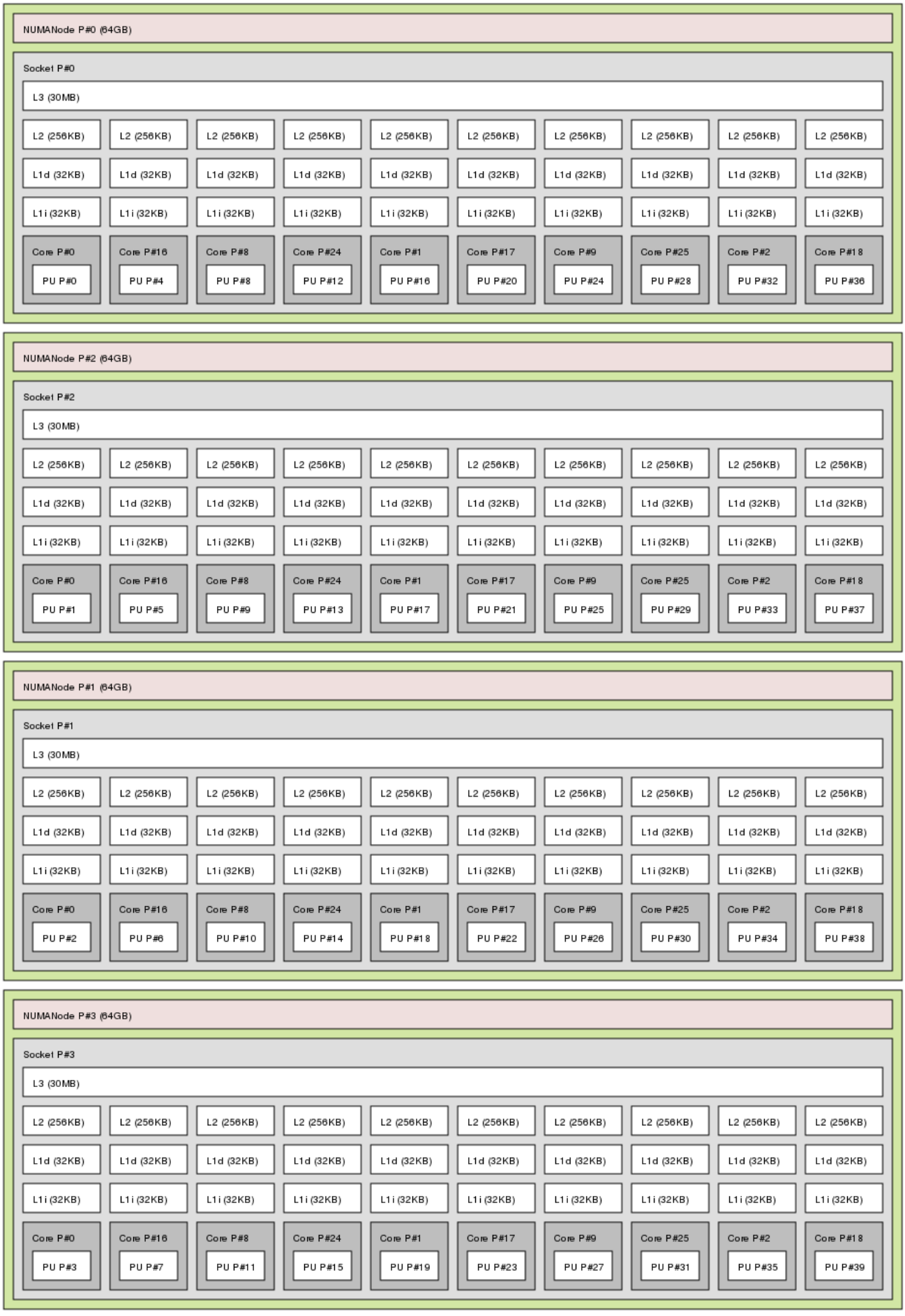
```
$ numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 4 8 12 16 20 24 28 32 36
node 0 size: 65415 MB
node 0 free: 43971 MB
node 1 cpus: 2 6 10 14 18 22 26 30 34 38
node 1 size: 65536 MB
node 1 free: 44321 MB
node 2 cpus: 1 5 9 13 17 21 25 29 33 37
node 2 size: 65536 MB
node 2 free: 44304 MB
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
```

由 util-linux 软件包提供的 lscpu 命令收集有关 CPU 架构的信息，如 CPU、线程、内核、插槽和 NUMA 节点的数量。

```
$ 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
```

由 hwloc 软件包提供的 lstopo 命令会创建您系统的图形表示。lstopo-no-graphics 命令提供了详细的文本输出。



lstopo 命令的输出

6.1.2. 调度

在 Red Hat Enterprise Linux 中，最小的进程执行单元称为线程。系统调度程序决定哪个处理器运行线程，以及线程运行的时间。但是，由于调度程序的主要顾虑是保持系统处于忙碌状态，因此可能无法最佳地为应用程序性能调度线程。

例如，当 Node B 上的处理器可用时，NUMA 系统上的应用程序在节点 A 上运行。为了让节点 B 处于忙碌状态，调度程序将一个应用程序的线程移到 Node B。但是，应用程序线程仍然需要访问节点 A 上的内存。因为线程现在在 Node B 上运行，并且 Node A 内存不再是线程的本地内存，所以访问它所需的时间会更长。线程在节点 B 上运行可能需要更长的时间，需要等待节点 A 上的处理器变得可用，并在原始节点上执行具有本地内存访问权限的线程。

对性能敏感的应用通常得益于设计人员或管理员确定线程的运行位置。有关如何根据性能敏感应用程序需求正确调度线程的详情请参考 第 6.3.6 节 “调优调度策略”。

6.1.2.1. 内核刻录

在以前的 Red Hat Enterprise Linux 版本中，Linux 内核会定期中断每个 CPU，以检查需要执行哪些工作。它利用结果对流程调度和负载平衡做出决策。这种常规中断称为内核环。

无论核心是否起作用，都会出现这一偏差。这意味着，即使空闲内核也经常被迫进入更高的功率状态（最多每秒 1000 倍）来响应中断。这导致系统无法有效地使用几代 x86 处理器中包含的深度睡眠状态。

在 Red Hat Enterprise Linux 6 和 7 中，默认情况下，内核不再中断空闲 CPU，而这往往处于低功率状态。此行为称为无空循环内核。当一个或多个任务正在运行时，定期中断被按需中断替代，从而使 CPU 保持空闲或更低的电源状态更长，并降低功耗。

红帽企业 Linux 7 提供了一个动态无循环选项(nohz_full)，通过减少内核对用户空间任务的干扰来进一步增强确定性。这个选项可在带有 nohz_full 内核参数的指定内核中启用。当在核心上启用这个选项时，所有计时活动都会移到非延迟敏感内核中。这对高性能计算和实时计算工作负载非常有用，其中用户空间任务对与内核定时器环关联的微秒级延迟特别敏感。

有关如何在 Red Hat Enterprise Linux 7 中启用动态无勾号行为的详情请参考 第 6.3.1 节 “配置内核调整时间”。

6.1.3. 中断请求(IRQ)处理

中断请求或 IRQ 是从硬件部分向处理器发送立即关注的信号。系统中的每个设备都会被分配一个或多个 IRQ 号，以允许它发送唯一中断。启用中断后，接收中断请求的处理器将立即暂停当前应用线程的执行，以解决中断请求。

由于中断正常操作，高中断率可能会严重降低系统性能。可以通过配置中断关联或在批处理中发送多个较低优先级中断（联合多个中断）来减少中断所花费的时间。

有关调整中断请求的详情请参考 第 6.3.7 节 “在 AMD64 和 Intel 64 中设置 Interrupt Affinity” 或 第 6.3.8 节 “使用 Tuna 配置 CPU、线程和中断关联性”。有关网络中断的详情请参考 第 9 章 网络。

6.2. 监控和诊断性能问题

红帽企业 Linux 7 提供了很多工具，这些工具可用于监控系统性能并诊断与处理器及其配置相关的性能问题。本节概述了可用的工具，并提供了如何使用它们监控和诊断处理器相关性能问题的示例。

6.2.1. turbostat

turbostat 以指定间隔打印计数器结果，以帮助管理员识别服务器中的意外行为，如过量电源使用、无法进入深度睡眠状态或系统管理中断(SMI)创建不必要。

turbostat 工具是 kernel-tools 软件包的一部分。它支持在带有 AMD64 和 Intel® 64 处理器的系统中使用。它需要 root 权限才能运行，并且处理器支持 invariant 时间戳计数器以及 APERF 和 MPERF 模型特定的寄存器。

有关用法示例，请参阅 man page:

```
$ man turbostat
```

6.2.2. numastat

重要

此工具在 Red Hat Enterprise Linux 6 生命周期中收到大量更新。虽然默认输出仍与 Andi Kleen 编写的原始工具兼容，但向 numastat 提供任何选项或参数都会显著更改其输出格式。

numastat 工具显示进程和操作系统的每个 NUMA 节点内存统计信息，并显示进程内存是否在整个系统中分散或集中到特定的节点上。

交叉引用每个处理器 顶部 输出的 numastat 输出，以确认进程线程正在分配进程的同一节点上运行。

numastat 由 numactl 软件包提供。有关 numastat 输出的详情，请查看 man page：

```
$ man numastat
```

6.2.3. /proc/interrupts

/proc/interrupts 文件列出了从特定 I/O 设备发送到每个处理器的中断数量。它显示中断请求(IRQ)编号、系统中各个处理器处理的中断请求数、发送的中断类型，以及以逗号分隔的设备列表来响应列出的中断请求。

如果特定应用或设备生成大量中断请求，由远程处理器处理，其性能可能会受到影响。在这种情况下，通过让处理器与应用程序或设备在同一节点上处理中断请求，可以降低性能。有关如何将中断处理分配给特定处理器的详情请参考 第 6.3.7 节 “在 AMD64 和 Intel 64 中设置 Interrupt Affinity”。

6.2.4. 使用 pqos 进行缓存和内存带宽监控

pqos 工具可从 intel-cmt-cat 软件包获得，可让您监控最近 Intel 处理器上的 CPU 缓存和内存带宽。

pqos 实用程序提供类似于 top 实用程序的缓存和内存监控工具。它监控：

- 每个周期(IPC)的说明。
- 最后一个级别缓存 MISSES 的数量。
- 程序在 LLC 中给定 CPU 占用空间中执行的 KB 大小。
- 到本地内存的带宽(MBL)。

- 到远程内存的带宽(MBR)。

使用以下命令启动监控工具：

```
# pqos --mon-top
```

输出中的项目按最高 LLC 排序。

其它资源

- 有关 pqos 实用程序和相关处理器功能的一般概述，请参阅 第 2.14 节 “pqos”。
- 有关如何使用 CAT 的示例，可以最大程度降低 noisy 邻居虚拟机对数据平面开发套件 (DPDK) 的网络性能的影响，请参阅《数据平面开发套件》Intel 白皮书的平台确定服务增加。

6.3. 配置建议

Red Hat Enterprise Linux 提供了很多工具来协助管理员配置系统。本节概述了可用的工具，并提供了如何使用它们来解决 Red Hat Enterprise Linux 7 中的处理器相关性能问题的示例。

6.3.1. 配置内核调整时间

默认情况下，Red Hat Enterprise Linux 7 使用无空循环内核，它不会中断空闲 CPU 以降低功耗并允许较新的处理器利用深奥睡眠状态。

红帽企业 Linux 7 还提供了动态无循环选项（默认为禁用），这对于非常敏感延迟的工作负载（如高性能计算或实时计算）非常有用。

要在特定内核中启用动态无空行为，使用 nohz_full 参数在内核命令行中指定这些内核。在 16 个内核系统中，指定 nohz_full=1-15 在内核 1 到 15 中启用动态无勾号行为，将所有时间记录到唯一未指定的内核（核心 0）。此行为可以在引导时临时启用，或者通过 /etc/default/grub 文件中的 GRUB_CMDLINE_LINUX 选项永久启用。对于永久行为，请运行 grub2-mkconfig -o /boot/grub2/grub.cfg 命令来保存您的配置。

启用动态无循环行为确实需要一些手动管理。

- 系统引导时，您必须手动将 rcu 线程移到非延迟敏感的内核中，本例中为 core 0。

```
# for i in `pgrep rcu[^c]` ; do taskset -pc 0 $i ; done
```

- 使用内核命令行中的 isolcpus 参数将特定内核与用户空间任务隔离。

- 另外，还可将内核的回写 bdi-flush 线程的 CPU 关联性设置为 housekeeping 内核：

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

通过执行以下命令验证动态无循环配置是否正常工作，其中 stress 是在 CPU 上启动 1 秒的程序。

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

压力的一种可能的替换是运行类似 while :; do d=1; done 的脚本。

默认内核计时器配置在忙碌的 CPU 上显示 1000 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
```

配置动态无循环内核后，您应该会看到 1 tick：

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

6.3.2. 设置硬件性能策略(x86_energy_perf_policy)

x86_energy_perf_policy 工具允许管理员定义性能和能源效率相对的重要性。当处理器选择在性能和能源效率之间取舍的选择时，可以利用这些信息来影响支持此功能的处理器。

默认情况下，它在 性能 模式的所有处理器上运行。它需要处理器支持，这通过存在 CPUID.06H.ECX.bit3 来指示，且必须以 root 权限运行。

x86_energy_perf_policy 由 kernel-tools 软件包提供。有关如何使用 x86_energy_perf_policy 的详情，请查看 第 A.9 节 “x86_energy_perf_policy” 或参考手册页：

```
$ man x86_energy_perf_policy
```

6.3.3. 使用 taskset 设置进程相关性

taskset 工具由 util-linux 软件包提供。taskset 允许管理员检索和设置正在运行的进程的处理器关联，或者启动具有指定处理器关联性的进程。

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优势，红帽建议使用 numactl 而不是 taskset。

有关 taskset 的详情，请参考 第 A.15 节 “taskset” 或 man page：

```
$ man taskset
```

6.3.4. 使用 numactl 管理 NUMA 关联性

管理员可以使用 numactl 来运行具有指定调度或内存放置策略的进程。numactl 也可以为共享内存段或文件设置持久策略，并设置进程的处理器关联和内存关联性。

在 NUMA 拓扑的系统中，处理器的内存访问速度随着处理器和内存数据库之间的距离增加而减慢。因此，务必要配置对性能敏感的应用，以便它们从最接近的内存库中分配内存。最好使用位于同一 NUMA 节点中的内存和 CPU。

对性能敏感的多线程应用或许能够配置为在特定 NUMA 节点上执行，而非特定的处理器。这是否适合取决于您的系统以及应用程序的要求。如果多个应用程序线程访问相同的缓存数据，则可能适合将这些线程配置为在同一处理器上执行。但是，如果访问和缓存同一处理器中执行的不同数据的多个线程可能会驱除以前的线程访问的缓存数据。这意味着每个线程"缺少"缓存会浪费执行时间从内存中获取数据并在缓存中替换数据。您可以使用 perf 工具（如 第 A.6 节 “perf” 所述）检查大量缓存未命中。

numactl 提供了多个选项，可帮助您管理处理器和内存关联性。详情请查看 第 A.11 节 “numastat” 或 man page:

```
$ man numactl
```

注意

numactl 软件包包含 libnuma 库。此库为内核支持的 NUMA 策略提供简单的编程接口，可用于比 numactl 应用更精细的调优。如需更多信息，请参阅 man page:

```
$ man numa
```

6.3.5. 使用 numad 自动 NUMA 关联性管理

numad 是一个自动 NUMA 关联性管理守护进程。它监控系统中 NUMA 拓扑和资源使用情况，以便动态改进 NUMA 资源的分配和管理。

numad 还提供了一个预放置建议服务，可由各种作业管理系统查询，为进程提供 CPU 和内存资源的初始绑定。无论 numad 是以可执行还是服务方式运行，此预置建议都可用。

有关如何使用 numad 的详情，请参考 第 A.13 节 “numad” 或参考 man page：

```
$ man numad
```

6.3.6. 调优调度策略

Linux 调度程序实施许多调度策略，这些策略决定了线程运行的位置和运行时间。调度策略主要分为两类：常规策略和实时策略。普通线程用于普通优先级的任务。实时策略用于必须在不中断的情况下完成的时间敏感任务。

实时线程不受时间分片的影响。这意味着他们将运行，直到它们阻断、退出、自愿给予，或者被更高优先级线程抢占。最低优先级实时线程排在采用正常策略的任何线程之前调度。

6.3.6.1. 调度策略

6.3.6.1.1. 使用 SCHED_FIFO 进行静态优先级调度

SCHED_FIFO （也称为静态优先级调度）是一个实时策略，为每个线程定义固定优先级。此策略允许管理员改进事件响应时间并缩短延迟，建议针对在较长时间段内不运行的对时间敏感的任务。

当使用 SCHED_FIFO 时，调度程序会按照优先级顺序扫描所有 SCHED_FIFO 线程的列表，并调度可供运行的最高优先级线程。SCHED_FIFO 线程的优先级级别可以是从 1 到 99 的任何整数，其中 99 被

视为最高优先级。红帽建议从较低的数量开始，只有在您发现延迟问题时才会增加优先级。

警告

因为实时线程不受时间片的限制，因此红帽不推荐设置 99 的优先级。这会将您的进程置于与迁移和 watchdog 线程相同的优先级级别；如果您的线程进入计算循环，并且这些线程已被阻止，它们将无法运行。具有单一处理器的系统最终会挂起。

管理员可以限制 SCHED_FIFO 带宽，以防止实时应用程序程序员启动对处理器进行单调执行的实时任务。

/proc/sys/kernel/sched_rt_period_us

此参数定义被认为是处理器带宽的 10% 的时间周期（以微秒为单位）。默认值为 1000000 crius，或 1 秒。

/proc/sys/kernel/sched_rt_runtime_us

这个参数定义专用于运行实时线程的微秒的时间周期。默认值为 950000 crius，或 0.95 秒。

6.3.6.1.2. 使用 SCHED_RR 进行循环优先级调度

SCHED_RR 是 SCHED_FIFO 的循环变体。当多个线程需要以同一优先级级别运行时，此策略很有用。

与 SCHED_FIFO 一样，SCHED_RR 是一个实时策略，用于为每个线程定义固定优先级。调度程序以优先级顺序扫描所有 SCHED_RR 线程的列表，并调度准备好运行的最高优先级线程。但是，与 SCHED_FIFO 不同，在特定时间片段中会调度具有相同优先级的线程。

您可以使用 sched_rr_timeslice_ms 内核参数以毫秒为单位设置这个时间片段的值 (/proc/sys/kernel/sched_rr_timeslice_ms)。最小值为 1 毫秒。

6.3.6.1.3. 使用 SCHED_OTHER 进行正常调度

SCHED_OTHER 是 Red Hat Enterprise Linux 7 中的默认调度策略。此策略使用完全公平调度程序

(CFS)来允许对使用此策略调度的所有线程进行公平处理器访问。当有大量线程或数据吞吐量是一个优先级时，此策略最有用，因为它允许更有效地调度线程。

使用此策略时，调度程序会根据每个进程线程的 niceness 值部分创建动态优先级列表。管理员可以更改进程的 nice 值，但不能直接更改调度程序的动态优先级列表。

有关更改流程 niceness 的详细信息，请参阅《红帽企业 Linux 7 系统管理员指南》。

6.3.6.2. 隔离 CPU

您可以使用 isolcpus 引导参数将一个或多个 CPU 与调度程序隔离。这可以防止调度程序在此 CPU 上调度任何用户空间线程。

隔离 CPU 后，您必须手动将进程分配给隔离的 CPU，或者使用 CPU 关联性系统调用或 numactl 命令。

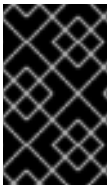
要将系统中的第八个 CPU 隔离，请在内核命令行中添加以下内容：

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

您还可以使用 Tuna 工具隔离 CPU。tuna 可以随时隔离 CPU，而不仅仅是在引导时。但是，这种隔离方法与 isolcpus 参数完全不同，目前没有实现与 isolcpus 相关的性能。有关此工具的详情，请查看 第 6.3.8 节 “使用 Tuna 配置 CPU、线程和中断关联性”。

6.3.7. 在 AMD64 和 Intel 64 中设置 Interrupt Affinity

中断请求具有关联的关联性属性 smp_affinity，它定义将处理中断请求的处理器。若要提高应用性能，可将中断关联和进程相关性分配到同一处理器或同一核心上的处理器。这允许指定的中断和应用程序线程共享缓存行。



重要

本节仅涵盖 AMD64 和 Intel 64 架构。在其他架构上中断关联性配置有很大不同。

过程 6.1. 自动平衡中断

-

如果您的 BIOS 导出它的 NUMA 拓扑，则 irqbalance 服务可自动为节点上对请求服务的硬件进行中断请求。

有关配置 irqbalance 的详情，请参考 第 A.1 节 “irqbalance”。

过程 6.2. 手动平衡中断

1. 检查哪些设备与您要配置的中断请求对应。

   从 Red Hat Enterprise Linux 7.5 开始，系统会自动配置特定设备及其驱动程序的最优中断关联性。您无法再手动配置它们的关联性。这适用于以下设备：

   - 使用 be2iscsi 驱动程序的设备
   - NVMe PCI 设备

2. 查找您的平台的硬件规格。检查系统上的芯片组是否支持分发中断。

   - 如果存在，您可以按照以下步骤中所述配置中断交付。

     此外，检查您的芯片组用于平衡中断的算法。某些 BIOS 拥有配置中断发送的选项。

   - 如果没有，您的芯片组总是将所有中断路由到单个静态 CPU。您无法配置使用哪个 CPU。

3. 检查您的系统上正在使用哪个高级可编程中断控制器(APIC)模式。

   只有非物理平面模式（flat）支持将中断分发到多个 CPU。这个模式只适用于最多 8 个 CPU 的系统。

   ```
   $ journalctl --dmesg | grep APIC
   ```

在命令输出中：

- 如果您的系统使用 flat 以外的模式，您可以看到一个类似于 Setting APIC routing to physical flat 的行。

- 如果看不到这个信息，代表您的系统使用 flat 模式。

如果您的系统使用 x2apic 模式，您可以通过在 bootloader 配置中的内核命令行中添加 nox2apic 选项来禁用它。

4. 计算 smp_affinity 掩码。

smp_affinity 值存储为代表系统中所有处理器的十六进制位掩码。每个位配置一个不同的 CPU。最不重要的位是 CPU 0。

掩码的默认值为 f，这意味着可在系统的任何处理器上处理中断请求。将此值设置为 1 表示只有处理器 0 可以处理中断。

过程 6.3. 计算掩码

1. 在二进制中，对处理中断的 CPU 使用值 1。

例如，要处理 CPU 0 和 CPU 7 的中断，请使用 0000000010000001 作为二进制代码：

表 6.1. CPU 的二进制 Bit

CPU	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
二进制	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

2. 将二进制代码转换为十六进制代码。

例如，使用 Python 转换二进制代码：

```
>>> hex(int('0000000010000001', 2))
'0x81'
```

在有 32 个处理器的系统上，您必须限制离散的 32 位组的 smp_affinity 值。例如，如果您只想 64 位处理器系统的第一个 32 个处理器来服务中断请求，请使用 0xffffffff,00000000。

5. 设置 smp_affinity 掩码。

特定中断请求的中断关联性值存储在关联的 /proc/irq/irq_number/smp_affinity 文件中。

将计算的掩码写入关联的文件：

```
# echo mask > /proc/irq/irq_number/smp_affinity
```

其它资源

- 在支持中断中断的系统上，修改中断请求的 smp_affinity 属性可设置硬件，以便决定使用特定处理器在硬件级别提供中断，而无需在内核中干预。

有关中断中断的详情请参考 第 9 章 网络。

6.3.8. 使用 Tuna 配置 CPU、线程和中断关联性

tuna 是调整正在运行的进程的工具，并可控制 CPU、线程和中断关联性，并为它可以控制的每种实体提供很多操作。有关 Tuna 的详情，请参考 第 4 章 tuna。

第 7 章 内存

本章概述了 Red Hat Enterprise Linux 7 的内存管理功能。第 7.1 节 “注意事项” 讨论影响性能的内存相关因素。第 7.2 节 “监控和诊断性能问题” 教您如何使用红帽企业 Linux 7 工具诊断与内存使用率或配置详情相关的性能问题。第 7.5 节 “配置系统内存容量” 和 第 7.3 节 “配置 HugeTLB Huge 页面” 讨论您可以使用哪些工具和策略解决 Red Hat Enterprise Linux 7 中的与内存相关的性能问题。

7.1. 注意事项

默认情况下，红帽企业 Linux 7 针对中等工作负载进行了优化。如果您的应用程序或工作负载需要大量内存，改变系统管理虚拟内存的方式可能会提高应用程序的性能。

7.1.1. 较大的页大小

物理内存以块的形式进行管理，称为页面。在 Red Hat Enterprise Linux 7 支持的大多数构架中，内存页面的默认大小为 4 KB。此默认页面大小已证明适合通用操作系统，如支持多种不同工作负载的 Red Hat Enterprise Linux 7。

然而，在某些情况下，特定应用程序可能会因使用更大的页面大小而受益。例如，处理数百兆字节甚至几十兆字节（甚至几十 GB）的大型且相对固定的数据集的应用在使用 4 KB 时性能问题可能会有问题。此类数据集可能需要数百千个 4 KB 页面，这可能会导致操作系统和 CPU 的开销。

红帽企业 Linux 7 支持将更大的页面大小用于处理大数据集的应用程序。使用更大的页面大小可提高此类应用程序的性能。

Red Hat Enterprise Linux 7 中提供了两个不同的 大页面功能： HugeTLB 功能，也称为静态巨页，以及 Transparent Huge Page 功能。

7.1.2. 翻译后备缓冲器大小

从页表中读取地址映射非常耗时且耗费资源，因此 CPU 构建时会使用最近使用的地址的缓存：转换查找缓冲区(TLB)。但是，默认 TLB 只能缓存一定数量的地址映射。如果请求的地址映射不在 TLB 中（也就是说，缺少 TLB），则系统仍需要读取页表以确定物理到虚拟地址映射。

由于应用程序内存要求和用于缓存地址映射的页面大小之间的关系，具有大型内存要求的应用程序更有可能遭受 TLB 未命中性能下降的问题，而不是内存要求最低的应用程序。因此，尽可能避免 TLB 未命中非常重要。

HugeTLB 和 Transparent Huge Page 功能允许应用程序使用大于 4 KB 的页面。这允许 TLB 中存储的地址引用更多内存，这可减少 TLB 未命中并提高应用性能。

7.2. 监控和诊断性能问题

红帽企业 Linux 7 提供了很多工具，这些工具可用于监控系统性能并诊断与系统内存相关的性能问题。本节概述了可用的工具，并提供了如何使用它们来监控和诊断与内存相关的性能问题的示例。

7.2.1. 使用 vmstat 监控内存使用情况

vmstat 由 procps-ng 软件包提供，输出系统进程、内存、分页、块输入/输出、中断和 CPU 活动的报告。它提供自计算机上一次启动或上次报告之后平均这些事件的即时报告。

以下命令显示各种事件计数器和内存统计信息的列表。

```
$ vmstat -s
```

有关如何使用 vmstat 的详情，请参考 第 A.8 节 “vmstat” 或 man page：

```
$ man vmstat
```

7.2.2. 使用 Valgrind 分析应用程序内存使用情况

Valgrind 是一个为用户空间二进制文件提供检测的框架。它随附了大量工具，可用于分析和分析程序性能。本节概述的 valgrind 工具可帮助您检测内存错误，如未初始化内存使用和不正确的内存分配或释放。

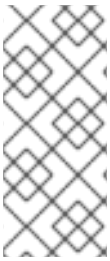
要使用 valgrind 或其任何工具，请安装 valgrind 软件包：

```
# yum install valgrind
```

7.2.2.1. 使用 Memcheck 分析内存使用情况

Memcheck 是默认的 valgrind 工具。它检测并报告一些难以检测和诊断的内存错误，例如：

- 不应发生的内存访问
- 未定义或未初始化值使用
- 释放堆内存不正确
- 指针重叠
- 内存泄漏



注意

Memcheck 只能报告这些错误，它无法防止它们发生。如果您的程序以通常会导致分段错误的方式访问内存，则分段错误仍会发生。但是，memcheck 会在故障前立即记录错误消息。

因为 memcheck 使用检测程序，所以通过 memcheck 执行的应用程序会运行 10 到三十倍，比通常要慢十倍。

要在应用程序上运行 memcheck，请执行以下命令：

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

您还可以使用以下选项将 memcheck 输出专注于特定类型的问题。

--leak-check

在应用程序完成执行后，memcheck 会搜索内存泄漏。默认值为 --leak-check=summary，它打印找到的内存泄漏数量。您可以指定 --leak-check=yes 或 --leak-check=full 来输出每个泄漏的详情。要禁用，请指定 --leak-check=no。

--undef-value-errors

默认值为 --undef-value-errors=yes，在使用未定义值时报告错误。您还可以指定 --undef-value-errors=no，这将禁用此报告并稍微加快 Memcheck。

--ignore-ranges

指定在检查内存可寻址时 memcheck 应该忽略的一个或多个范围，例如 ：--ignore-ranges=0xPP-0xQQ,0xR-0xSS。

有关 memcheck 选项的完整列表，请查看包含在 /usr/share/doc/valgrind-version/valgrind_manual.pdf 的文档。

7.2.2.2. 使用 Cachegrind 分析缓存使用情况

cachegrind 模拟应用程序与系统的缓存层次结构和分支预测器的交互。它跟踪模拟第一级指令和数据缓存的使用情况，以检测与这一缓存级别交互较差的代码。它还跟踪最后一级缓存（第二或第三级），以跟踪内存访问。因此，使用 cachegrind 执行的应用程序会运行 twenty，比通常要慢一百倍。

cache grind 收集应用程序执行期间的统计信息，并输出控制台的摘要。要在应用程序上运行 cachegrind，请执行以下命令：

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

您还可以使用以下选项将 cachegrind 输出专注于特定问题。

--I1

指定第一个指令缓存的大小、关联和行大小，例如 ：--I1=size、关联性、line_size。

--D1

指定第一级数据缓存的大小、关联性和行大小，例如 ：--D1=size、关联性、line_size。

--LL

指定最后一个缓存的大小、关联和行大小，例如 ：--LL=size、关联性、line_size。

--cache-sim

启用或禁用缓存访问和未命中计数的集合。默认启用(--cache-sim=yes)。禁用这个和 --branch-sim 会保留 cachegrind，没有要收集的信息。

--branch-sim

启用或禁用分支指令的集合和不正确的预测计数。默认启用(**--branch-sim=yes**)。禁用这个和 **--cache-sim** 会保留 **cachegrind**，没有要收集的信息。

cachegrind 将详细的性能分析信息写入每个进程 **cachegrind.out.pid** 文件，其中 **pid** 是进程标识符。此详细信息可以通过 **companion cg_annotate** 工具进一步处理，例如：

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

cachegrind 还提供 **cg_diff** 工具，这有助于在代码更改之前和之后图表程序性能。若要比较输出文件，可执行以下命令，首先将 替换为初始配置文件输出文件，第二个则替换为后续配置文件输出文件：

```
# cg_diff first second
```

可以通过 **cg_annotate** 工具查看生成的输出文件。

有关 **cachegrind** 选项的完整列表，请查看 **/usr/share/doc/valgrind-version/valgrind_manual.pdf** 的文档。

7.2.2.3. 使用 Massif 分析 Heap 和堆栈空间

Massif 测量指定应用程序使用的堆空间。它测量有用的空间和分配用于预订和调整的其他空间。**massif** 帮助您了解如何减少应用程序的内存使用来加快执行速度，并降低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大约会比通常的 **Tenty** 慢。

要在应用程序上运行 **massif**，请执行以下命令：

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

您还可以使用以下选项将 **massif** 输出专注于特定问题。

--heap

指定 **massif** 配置集是否为堆。默认值为 **--heap=yes**。可以通过将其设置为 **--heap=no** 来禁用堆性能分析。

--heap-admin

指定启用堆性能分析时要用于管理的每个块的字节数。默认值为 8 字节。

--stacks

指定堆栈的 massif 配置集。默认值为 --stack=no，因为堆栈性能分析可能会大大减慢 massif。将此选项设置为 --stack=yes 以启用堆栈性能分析。请注意，mass if 假定主堆栈以 0 大小开头，以便更好地指示与正在配置文件应用相关的堆栈大小的变化。

--time-unit

指定 massif 收集性能分析数据的间隔。默认值为 i （已执行）。您还可以指定 ms （毫秒或实时）和 B （在堆和堆栈上分配的字节或取消分配）。检查分配的字节对于短期运行的应用和测试非常有用，因为它可以在不同的硬件之间重复生成。

Massif 输出性能分析数据到 massif.out.pid 文件中，其中 pid 是指定应用程序的进程标识符。ms_print 工具会图形此性能分析数据，以显示应用的执行上的内存消耗，以及负责在峰值内存分配点处分配的站点的详细信息。要绘制 massif.out.pid 文件中的数据，请执行以下命令：

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

有关 Massif 选项的完整列表，请查看 /usr/share/doc/valgrind-version/valgrind_manual.pdf 的文档。

7.3. 配置 HUGETLB HUGE 页面

从 Red Hat Enterprise Linux 7.1 开始，可以通过两种方式保留巨页：在引导时 和 在运行时。在引导时进行保留会增加成功的可能性，因为内存还没有很大的碎片。但是，在 NUMA 计算机上，页面数会自动分割在 NUMA 节点之间。通过 run-time 方法，您可以为每个 NUMA 节点保留巨页。如果在引导过程中尽快执行运行时保留，则内存碎片的可能性会降低。

7.3.1. 在引导时配置大页面

要在引导时配置巨页，在内核引导命令行中添加以下参数：

hugepages

定义在启动时在内核中配置的持久大页面数量。默认值为 0。只有当系统中有足够的物理连续的空闲页面时，才可以分配巨页。此参数保留的页面不能用于其他目的。

通过更改 /proc/sys/vm/nr_hugepages 文件的值，可以在安装后调整这个值。

在 NUMA 系统中，使用这个参数分配的巨页会在节点间平均分割。您可以通过更改节点的 /sys/devices/system/node/node_id/hugepages/hugepages-1048576kB/nr_hugepages 文件在运行时将巨页分配给特定节点。

如需更多信息，请参阅包括在 /usr/share/doc/kernel-doc-kernel_version/Documentation/vm/hugetlbpage.txt 中的相关内核文档。

hugepagesz

定义在启动时在内核中配置的持久大页面大小。有效值为 2 MB 和 1 GB。默认值为 2 MB。

default_hugepagesz

定义在启动时在内核中配置的持久性巨页的默认大小。有效值为 2 MB 和 1 GB。默认值为 2 MB。

有关如何在内核引导命令行中添加参数的详情，请参考 Chapter 3。Red Hat Enterprise Linux 7 内核管理指南中的内核参数和值列表。

过程 7.1. 在早期引导期间保留 1 GB 页面

HugeTLB 子系统支持的页面大小取决于架构。在 AMD64 和 Intel 64 构架中，支持 2 MB 的巨页和 1 GB 巨页。

1. 以 root 用户身份将以下行附加到 /etc/default/grub 文件中的内核命令行选项中，为 1 GB 页面创建一个 HugeTLB 池：

```
default_hugepagesz=1G hugepagesz=1G
```

2. 使用编辑的默认 文件重新生成 GRUB2 配置。如果您的系统使用 BIOS 固件，请执行以下命令：

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

*在带有 **UEFI** 固件的系统中，执行以下命令：*

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

3. *创建名为 **/usr/lib/systemd/system/hugetlb-gigantic-pages.service** 的文件，其内容如下：*

```
[Unit]
Description=HugeTLB Gigantic Pages Reservation
DefaultDependencies=no
Before=dev-hugepages.mount
ConditionPathExists=/sys/devices/system/node
ConditionKernelCommandLine=hugepagesz=1G

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/lib/systemd/hugetlb-reserve-pages.sh

[Install]
WantedBy=sysinit.target
```

4. *创建名为 **/usr/lib/systemd/hugetlb-reserve-pages.sh** 的文件，其内容如下：*

```
#!/bin/sh

nodes_path=/sys/devices/system/node/
if [ ! -d $nodes_path ]; then
 echo "ERROR: $nodes_path does not exist"
 exit 1
fi

reserve_pages()
{
 echo $1 > $nodes_path/$2/hugepages/hugepages-1048576kB/nr_hugepages
}

reserve_pages number_of_pages node
```

*在最后一行中，将 **number_of_pages** 替换为要保留的 **1GB** 页面数量，将 **node** 替换为要保留这些页面的节点名称。*

-

例 7.1. 在 node0 和 node1上保留页面

例如，要在 node0 上保留两个 1GB 页面，并在 node1 上保留 1GB 页面，将最后一行替换为以下代码：

```
reserve_pages 2 node0
reserve_pages 1 node1
```

您可以根据需要进行修改，或添加更多行以在其他节点中保留内存。

5. 使脚本可执行：

```
# chmod +x /usr/lib/systemd/hugetlb-reserve-pages.sh
```

6. 启用早期引导保留：

```
# systemctl enable hugetlb-gigantic-pages
```

注意

您可以在运行时尝试保留更多 1GB 页面，方法是写入 nr_hugepages。但是，为了避免因为内存碎片而出现故障，请在引导过程早期保留 1GB 页面。

7.3.2. 在运行时配置大页面

在运行时使用以下参数来影响巨页行为：

/sys/devices/system/node/node_id/hugepages/hugepages-size/nr_hugepages

定义分配给指定 NUMA 节点的指定大小的巨页数量。从 Red Hat Enterprise Linux 7.1 开始支持它。以下示例移动会将 twenty 2048 kB 巨页添加到 node2。

```
# numastat -cm | egrep 'Node|Huge'
              Node 0 Node 1 Node 2 Node 3  Total add
AnonHugePages         0      2      0      8     10
HugePages_Total       0      0      0      0      0
HugePages_Free        0      0      0      0      0
HugePages_Surp        0      0      0      0      0
# echo 20 > /sys/devices/system/node/node2/hugepages/hugepages-2048kB/nr_hugepages
```

```
# numastat -cm | egrep 'Node|Huge'
              Node 0 Node 1 Node 2 Node 3  Total
AnonHugePages      0      2      0      8     10
HugePages_Total    0      0     40      0     40
HugePages_Free     0      0     40      0     40
HugePages_Surp     0      0      0      0      0
```

/proc/sys/vm/nr_overcommit_hugepages

定义系统可以通过过量使用内存来创建和使用的额外巨页的最大数量。将任何非零值写入该文件表示系统会从内核的普通页面池中获取巨页的数量（如果持久的大页面池已耗尽）。当这些大页面变得未使用后，它们会被释放并返回到内核的普通页面池。

7.4. 配置 THG

Transparent Huge Pages(THP)是 HugeTLB 的替代解决方案。使用 THP 时，内核会自动为进程分配巨页，因此不需要手动保留巨页。

THP 功能有两种操作模式：系统范围以及每个进程。当在系统范围内启用 THP 时，当有可能分配大页面且进程使用很大的连续虚拟内存区域时，内核会尝试为任何进程分配大页面。如果每个进程启用了 THP，则内核只为使用 madvise() 系统调用指定的单个进程的内存区域分配巨页。

请注意，**THP 功能只支持 2-MB 页面。默认启用透明大内存页。要查看当前状态，请运行：**

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
```

要启用透明巨页，请运行：

```
# echo always > /sys/kernel/mm/transparent_hugepage/enabled
```

要防止应用程序分配超过必要数量的内存资源，您可以通过运行以下命令在 MADV_HUGEPAGE madvise 区域中禁用巨页：

```
# echo madvise > /sys/kernel/mm/transparent_hugepage/enabled
```

要禁用透明巨页，请运行：

```
# echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

有时，为短期的分配提供低延迟的优先级高于立即通过长期分配实现最佳性能。在这种情况下，可以在启用 THP 时禁用 直接压缩。

直接紧凑是在巨页分配期间异步内存紧凑。禁用直接紧凑不保证保存内存，但可以降低频繁页面错误时出现较高延迟的风险。请注意，如果工作负载显著受益于 THP，性能会降低。要禁用直接压缩，请运行：

```
# echo madvise > /sys/kernel/mm/transparent_hugepage/defrag
```

有关透明大内存页的综合信息，请查看 /usr/share/doc/kernel-doc-kernel_version/Documentation/vm/transhuge.txt 文件，该文件可在安装 kernel-doc 软件包后可用。

7.5. 配置系统内存容量

本节讨论与内存相关的内核参数，这些参数可能有助于提高系统上的内存利用率。通过更改 /proc 文件系统中对应文件的值，可以临时为测试目的设置这些参数。确定为用例生成最佳性能的值后，您可以使用 sysctl 命令永久设置它们。

内存用量通常通过设置一个或多个内核参数的值来配置。这些参数可以通过更改 /proc 文件系统中文件的内容来临时设置，也可以使用 sysctl 工具（由 procps-ng 软件包提供）永久设置。

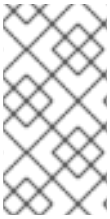
例如，要临时将 overcommit_memory 参数设置为 1，请运行以下命令：

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

要永久设置这个值，请在 /etc/sysctl.conf 中添加 sysctl vm.overcommit_memory=1，然后运行以下命令：

```
# sysctl -p
```

临时设置参数对于确定参数对您的系统的影响很有用。当您确定参数的值具有所需的效果时，您可以永久设置 参数。



注意

为了扩展您的专业知识，您可能还对红帽企业 Linux 性能调优(RH442) 培训课程感兴趣。

7.5.1. 虚拟内存参数

本节中列出的参数位于 /proc/sys/vm 中，除非另有说明。

dirty_ratio

百分比值.当修改系统内存总量的百分比时，系统会开始使用 pdflush 操作将修改写入磁盘。默认值为 20%。

dirty_background_ratio

百分比值.修改系统内存总量的百分比时，系统开始在后台将修改写入磁盘。默认值为 10%。

overcommit_memory

定义确定大型内存请求是否被接受或拒绝的条件。

默认值为 0。默认情况下，内核通过估算可用内存量和失败的请求量来执行启发式内存过量使用处理。但是，由于内存是使用启发式而不是精确算法来分配的，因此此设置可能会造成内存过载。

当这个参数设置为 1 时，内核不会执行内存过量使用处理。这增加了内存过载的可能性，但提高了内存密集型任务的性能。

当这个参数设置为 2 时，内核拒绝对内存的请求等于或大于可用交换空间总量以及 overcommit_ratio 中指定的物理 RAM 百分比。这降低了过量使用内存的风险，但建议仅用于交换区域大于物理内存的系统。

overcommit_ratio

指定当 overcommit_memory 设置为 2 时所考虑的物理 RAM 的百分比。默认值为 50。

max_map_count

定义进程可以使用的最大内存映射区域数。默认值(65530)适用于大多数情况。如果您的应用需要映射超过这个数目的文件，请提高这个值。

min_free_kbytes

指定在系统中保持可用最小 KB 数。这用于为每个低内存区域确定适当的值，每个低内存区域会根据它们的大小分配大量保留的可用页面。

警告

极端值可能会破坏您的系统。将 min_free_kbytes 设置为非常低的值可防止系统回收内存，这可能会导致系统挂起和 OOM 终止进程。但是，设置 min_free_kbytes 太高（例如，系统内存总量为 5-10%）会导致系统立即进入内存不足状态，从而导致系统花费太多时间回收内存。

oom_adj

如果系统内存不足，panic_on_oom 参数被设置为 0， oom_killer 函数会终止进程，直到系统可以恢复，从具有最高 oom_score 的进程开始。

oom_adj 参数帮助确定进程的 oom_score这个参数是为每个进程标识符设置的。值 -17 可禁用该进程的 oom_killer。其他有效值包括从 -16 到 15。

注意

由调整的进程生成的进程会继承 oom_score 进程的 oom_score。

swappiness

swappiness 值范围从 0 到 100，控制系统优先选择匿名内存或页面缓存的程度。高值提高了文件系统性能，同时将较少的活动进程从 RAM 交换出来。低值可避免将进程交换出内存，这通常会降低 I/O 性能。默认值为 60。

警告

设置 swappiness==0 将非常积极避免交换，这会增加 OOM 终止遭受强内存和 I/O 压力的风险。

7.5.2. 文件系统参数

本节中列出的参数位于 /proc/sys/fs 中，除非另有说明。

aio-max-nr

定义所有活跃的异步输入/输出上下文中允许的最大事件数。默认值为 65536。修改这个值不会预先分配或重新定义任何内核数据结构。

file-max

确定整个系统处理的最大文件数。Red Hat Enterprise Linux 7 的默认值是 8192 的最大值，或者内核启动时可用的空闲内存页中最多的 10 个。

增加这个值可解决缺少可用文件 handle 导致的错误。

7.5.3. 内核参数

以下参数的默认值（位于 /proc/sys/kernel/ 目录中）可在引导时由内核计算，具体取决于可用的系统资源。

msgmax

定义消息队列中任意条消息的最大允许大小，以字节为单位。这个值不能超过队列的大小 (msgmnb)。要确定系统中的当前 msgmax 值，请使用：

```
# sysctl kernel.msgmax
```

msgmnb

定义单个消息队列的最大字节大小。要确定系统中的当前 msgmnb 值，请使用：

```
# sysctl kernel.msgmnb
```

msgmni

定义消息队列标识符的最大数量，以及队列的最大数量。要确定系统中的当前 msgmni 值，请使用：

```
# sysctl kernel.msgmni
```

shmall

定义一次系统上可以使用的共享内存页面总量。例如，AMD64 和 Intel 64 构架中的页面是 4096 字节。

要确定系统中的当前 shmall 值，请使用：

```
# sysctl kernel.shmall
```

shmmax

定义内核允许的单个共享内存段的最大大小（以字节为单位）。要确定系统中的当前 shmmax 值，请使用：

```
# sysctl kernel.shmmax
```

shmmni

定义系统范围的最大共享内存片段数。所有系统上的默认值为 4096。

threads-max

定义一次内核可用的系统范围最大线程数。要确定系统中的当前 threads-max 值，请使用：

```
# sysctl kernel.threads-max
```

默认值为以下结果：

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

最小值为 20。

第 8 章 存储和文件系统

本章概述了在 Red Hat Enterprise Linux 7 中影响应用程序性能的文件系统和配置选项。第 8.1 节 “注意事项” 讨论影响性能的 I/O 和文件系统相关因素。第 8.2 节 “监控和诊断性能问题” 教您如何使用红帽企业 Linux 7 工具诊断与 I/O 或文件系统配置详细信息相关的性能问题。第 8.4 节 “配置工具” 讨论可用于解决 Red Hat Enterprise Linux 7 中的 I/O 和文件系统相关性能问题的工具和策略。

8.1. 注意事项

存储和文件系统性能的相应设置高度依赖于存储的用途。I/O 和文件系统性能可能会受到以下任何因素的影响：

- 数据写入或读取模式
- 与底层几metry 的数据一致
- 块大小
- 文件系统大小
- 日志大小和位置
- 记录访问时间
- 确保数据可靠性
- 预获取数据
- 预分配磁盘空间
- 文件碎片

- 资源争用

阅读本章，以了解影响文件系统吞吐量、可扩展性、响应性、资源使用情况和可用性的格式和挂载选项。

8.1.1. I/O 调度程序

I/O 调度程序决定 I/O 操作在存储设备上运行的时间和时长。它也称为 I/O 电梯。

红帽企业 Linux 7 提供了三个 I/O 调度程序：

deadline

所有块设备的默认 I/O 调度程序，SATA 磁盘除外。截止时间尝试为请求到达 I/O 调度程序的点提供有保证的延迟。这个调度程序适合大多数用例，特别是那些读取操作的频率比写操作多。

排队的 I/O 请求被分类为读或写批处理，然后计划以增加 LBA 顺序来执行。默认情况下，读取批处理优先于写入批处理，因为应用更有可能阻止读取 I/O。处理批处理后，截止时间检查写操作在处理器时间不足，并根据情况调度下一个读取或写入批处理。每个批处理处理的请求数量、每个写入批处理要发布的读取批处理数以及请求过期前的时间数量都是可配置的；有关详细信息，请参阅 第 8.4.4 节 “调整截止调度程序”。

cfq

仅识别为 SATA 磁盘的设备的默认调度程序。完全公平队列调度程序 cfq 将进程划分为三个单独的类：实时时间、尽力和空闲。实时类中的进程始终在最佳工作类进程之前执行，这些进程始终在空闲类中的进程之前执行。这意味着实时类中的进程可以缺少最佳工作量和空闲处理器时间的进程。默认情况下，进程分配到最佳工作类。

cfq 使用历史数据来预测应用程序是否在近期发出更多 I/O 请求。如果预期有更多 I/O，cfq 会闲置等待新的 I/O，即使等待处理其他进程的 I/O。

因此，除非调整 cfq 调度程序，否则不应将 cfq 调度程序与不会产生大量追求罚款的硬件一起使用。它不应该与其他非工作服务的调度程序（如基于主机的硬件 RAID 控制器）一起使用，因为堆栈这些调度程序会导致大量延迟。

cfq 行为高度可配置；详情请参阅 第 8.4.5 节 “调优 CFQ 调度程序”。

noop

noop I/O 调度程序实施一个简单的 FIFO （第一出）调度算法。请求通过简单的最后一个缓存合并到通用块层。这可以是使用快速存储的 CPU 绑定系统的最佳调度程序。

有关设置不同默认 I/O 调度程序或为特定设备指定不同调度程序的详情，请参考 第 8.4 节 “配置工具”。

8.1.2. 文件系统

阅读本节，以了解 Red Hat Enterprise Linux 7 中支持的文件系统及其推荐用例以及文件系统可使用的格式和挂载选项的详细信息。有关这些文件系统的详细调整建议，请参考 第 8.4.7 节 “为性能配置文件系统”。

8.1.2.1. XFS

XFS 是可靠、高度可扩展的 64 位文件系统。它是 Red Hat Enterprise Linux 7 中的默认文件系统。XFS 使用基于扩展的分配，具有许多分配方案，包括预分配和延迟分配，二者均减少了碎片和辅助性能。它还支持元数据日志，这可促进崩溃恢复。XFS 可以在挂载和激活时进行碎片整理和扩展，Red Hat Enterprise Linux 7 支持几个特定于 XFS 的备份和恢复实用程序。

从红帽企业 Linux 7.0 GA 开始，支持 XFS 的最大文件系统大小为 500 TB，最大文件偏移为 8 EB（稀疏文件）。有关管理 XFS 的详情，请查看红帽企业 Linux 7 存储管理指南。有关针对特定目的调优 XFS 的帮助信息，请参阅 第 8.4.7.1 节 “调优 XFS”。

8.1.2.2. ext4

Ext4 是 ext3 文件系统的可扩展扩展。它的默认行为是大部分工作负载的最佳方式。但是，它只支持的最大文件系统大小为 50 TB，最大文件大小为 16 TB。有关管理 ext4 的详情，请查看红帽企业 Linux 7 存储管理指南。有关针对特定目的调整 ext4 的帮助信息，请参阅 第 8.4.7.2 节 “调优 ext4”。

8.1.2.3. Btrfs（技术预览）

Red Hat Enterprise Linux 7 的默认文件系统是 XFS。Btrfs（B-tree 文件系统）是相对较新的写时复制(COW) 文件系统，作为技术预览提供。些独特的 Btrfs 功能包括：

- *能够对特定文件、卷或子卷拍摄快照，而不是整个文件系统；*

- 支持多种版本的冗余磁盘阵列(RAID)；
- 将映射 I/O 错误回引用文件系统对象；
- 透明压缩（分区上的所有文件会自动压缩）
- 数据和元数据的校验和.

虽然 Btrfs 被视为稳定的文件系统，但它仍在持续开发中，因此与更成熟的文件系统相比，某些功能（如修复工具）是基本的。

目前，当需要高级功能（如快照、压缩和文件数据校验和）时，选择 Btrfs 非常适合，但性能相对不重要。如果不需要高级功能，则随着时间推移，失败的风险和性能相对较差，这让其他文件系统更可取。与其他文件系统相比，另一个缺点是最大支持 50 TB 的文件系统大小。

如需更多信息，请参阅 Red Hat Enterprise Linux 7 存储管理指南中的 第 8.4.7.3 节 “调优 Btrfs” 和 Btrfs 章节。

8.1.2.4. GFS2

全局文件系统 2(GFS2)是高可用性附加组件的一部分，为红帽企业 Linux 7 提供集群文件系统支持。GFS2 在集群中的所有服务器中提供一个一致的文件系统镜像，允许服务器从单一共享文件系统读取和写入。

GFS2 支持的最大文件系统大小为 100 TB。

有关管理 GFS2 的详情，请参阅全局文件系统 2 指南或红帽企业 Linux 7 存储管理指南。有关根据特定目的调整 GFS2 的详情请参考 第 8.4.7.4 节 “调整 GFS2”。

8.1.3. 文件系统的通用调整注意事项

本节介绍所有文件系统常见的调优注意事项。有关特定于您的文件系统的建议，请参考 第 8.4.7 节 “为性能配置文件系统”。

8.1.3.1. 格式时的注意事项

在设备格式化后，无法更改一些文件系统配置决策。本节介绍了在格式化存储设备前必须做出的决策的选项。

Size

为您的工作负载创建适当大小的文件系统。较小的文件系统备份时间按比例缩短，需要较少的时间和内存进行文件系统检查。但是，如果您的文件系统太小，其性能将会高碎片。

块大小

块是文件系统的工作单元。块大小决定了单个块中可以存储多少数据，因此一次写入或读取的最小数据量。

默认块大小适用于大多数用例。但是，如果块大小（或多个块的大小）与通常一次读取或写入的数据量相同，那么文件系统的性能将会更好，并且存储数据效率更高。个小文件仍将使用整个 块。文件可以分散到多个块中，但这会产生额外的运行时开销。此外，某些文件系统仅限于特定数量的块，进而限制文件系统的最大大小。

使用 **mkfs** 命令格式化设备时，将块大小指定为文件系统选项的一部分。指定块大小的参数因文件系统而异；有关您的文件系统的详情，请查看 **mkfs** 手册页。例如，若要查看在格式化 **XFS** 文件系统时可用的选项，请执行以下命令：

```
$ man mkfs.xfs
```

geometry

文件系统 **geometry** 涉及文件系统中数据的分布。如果您的系统使用分条存储（如 **RAID**），您可以在格式化该设备时将数据和元数据与底层存储几兆对齐来提高性能。

许多设备会导出建议的 **geometry**，然后在设备使用特定文件系统格式化时自动设置。如果您的设备没有导出这些建议，或者要更改推荐的设置，您必须在使用 **mkfs** 格式化设备时手动指定 **geometry**。

指定文件系统 **geometry** 的参数因文件系统而异；详情请参阅 **mkfs man page**。例如，若要查看在格式化 **ext4** 文件系统时可用的选项，请执行以下命令：

```
$ man mkfs.ext4
```

外部日志

日志文件系统记录了在执行操作之前，日志文件中写入操作期间将要进行的更改。这降低了存储设备在系统崩溃或电源故障时损坏的可能性，并加快恢复过程。

元数据密集型工作负载涉及非常频繁的日志更新。较大的日志使用更多的内存，但会降低写入操作的频率。此外，您可以通过将其日志放在专用存储中，使其与主存储快或快于主存储，从而缩短具有元数据密集型工作负载的设备的搜索时间。

警告

确保外部日志可靠。丢失外部日志设备将导致文件系统损坏。

必须在格式时创建外部日志，并在挂载时指定日志设备。详情请查看 mkfs 和 mount man page。

```
$ man mkfs
```

```
$ man mount
```

8.1.3.2. 挂载时间注意事项

本节论述了适用于大多数文件系统的调优决策，可以在挂载设备时指定。

障碍

文件系统障碍可确保在持久性存储中正确写入和排序文件系统元数据，且使用 fsync 传输的数据在断电后保留。在以前的 Red Hat Enterprise Linux 版本中，启用文件系统障碍可能会明显减慢大量依赖 fsync 的应用程序，或者创建和删除很多小文件。

在红帽企业 Linux 7 中，文件系统障碍性能得到了改进，从而禁用文件系统障碍的性能影响可忽略不计（少于 3%）。

如需更多信息，请参阅《红帽企业 Linux 7 存储管理指南》。

访问时间

每次读取文件时，都会使用哪个时间(time)更新其元数据。这涉及额外的写入 I/O。在大多数情况下，这个开销最小，因为默认情况下，Red Hat Enterprise Linux 7 只有在以前的访问时间早于上次修改时间(mtime)或状态更改(ctime)时，才会更新 atime 字段。

但是，如果更新此元数据非常耗时，并且不需要准确的访问时间数据，您可以使用 noatime 挂载选项挂载文件系统。这会在文件读取时禁用对元数据的更新。它还启用 nodiratime 行为，这可在读取目录时禁用对元数据的更新。

read-ahead

读出行为可加快文件访问的速度，方法是：快速获取可能需要的数据并将其加载到页面缓存中，其中的检索速度比磁盘上快。读-ad-head 值越大，系统预填充数据前面是另一个。

Red Hat Enterprise Linux 尝试根据它检测到的文件系统设置适当的读号值。但是，准确检测并不总是被允许。例如，如果存储阵列作为单个 LUN 向系统显示自己，则系统会检测到单个 LUN，且不会为阵列设置适当的读-ahead 值。

涉及大量流传输顺序 I/O 的工作负载通常得益于高读头值。红帽企业 Linux 7 提供的与存储相关的调优配置集提高了读头值，与使用 LVM 条带一样，这些调整并非始终足以满足所有工作负载的需要。

定义读取行为的参数因文件系统而异；有关详细信息，请参阅 mount man page。

```
$ man mount
```

8.1.3.3. 维护

对于固态磁盘和精简置备的存储，建议定期丢弃文件系统未使用的块。丢弃未使用块有两种方法：批量丢弃和在线丢弃。

批量丢弃

这种丢弃是 fstrim 命令的一部分。它丢弃文件系统中的所有未使用块，这些块符合管理员指定的标准。

Red Hat Enterprise Linux 7 支持在支持物理丢弃操作的 XFS 和 ext4 格式的设备上批量丢弃（即，在 /sys/block/devname/queue/discard_max_bytes 的值不是 0）的 HDD 设备，其中

/sys/block/devname/queue/discard_granularity 的值不是 0的 SSD 设备。

在线丢弃

这种丢弃操作在挂载时通过 discard 选项进行配置，并在用户不干预的情况下实时运行。但是，在线丢弃仅丢弃从已使用到空闲的块。Red Hat Enterprise Linux 7 支持在 XFS 和 ext4 格式的设备上进行在线丢弃。

红帽建议批量丢弃，除非需要在线丢弃才能保持性能，或者在系统工作负载无法使用批量丢弃的情况下。

预分配

预分配将磁盘空间标记为分配给文件，而不将任何数据写入该空间。这可用于限制数据碎片和读性能差。Red Hat Enterprise Linux 7 支持挂载时在 XFS、ext4 和 GFS2 设备上预先分配空间；有关您的文件系统的适当参数的 mount man page。应用程序也可以使用 fallocate(2) glibc 调用从预先分配空间中受益。

8.2. 监控和诊断性能问题

红帽企业 Linux 7 提供了很多工具，这些工具可用于监控系统性能以及诊断与 I/O 和文件系统及其配置相关的性能问题。本节概述了可用的工具，并提供了如何使用它们监控和诊断 I/O 和文件系统相关性能问题的示例。

8.2.1. 使用 vmstat 监控系统性能

vmstat 报告整个系统中进程、内存、分页、块 I/O、中断和 CPU 活动。它可以帮助管理员确定 I/O 子系统是否负责处理任何性能问题。

与 I/O 性能最相关的信息如下：

si

在 KB 中交换或读取交换空间.

因此

KB 交换或写入交换空间.

bi

在 KB 中阻止或阻止写入操作.

bo

在 KB 中阻止或阻止读取操作.

wa

等待 I/O 操作完成的队列部分。

当您的交换空间和数据位于同一个设备上以及作为内存使用情况指示器时，交换和交换出特别有用。

此外，free、buff 和 cache 列可以帮助识别回写频率。缓存值的突然下降以及可用值的增加表示已启动回写和页面缓存无效。

如果使用 vmstat 进行分析显示 I/O 子系统负责降低性能，管理员可以使用 iostat 来确定负责的 I/O 设备。

vmstat 由 procps-ng 软件包提供。有关使用 vmstat 的详情，请查看 man page：

```
$ man vmstat
```

8.2.2. 使用 iostat 监控 I/O 性能

iostat 由 sysstat 软件包提供。它报告您系统中的 I/O 设备负载。如果使用 vmstat 进行分析显示 I/O 子系统负责降低性能，您可以使用 iostat 确定负责的 I/O 设备。

您可以使用 iostat man page 中定义的参数将 iostat 报告输出到特定设备：

```
$ man iostat
```

8.2.2.1. 使用 blktrace 的详细 I/O 分析

blktrace 提供有关在 I/O 子系统中如何使用时间的详细信息。companion 工具 blkparse 读取来自

blktrace 的原始输出，并生成由 blktrace 记录的输入和输出操作的人类可读摘要。

有关这个工具的详情，请查看 blktrace(8) 和 blkparse(1) man page：

```
$ man blktrace
```

```
$ man blkparse
```

8.2.2.2. 使用 btt 分析 blktrace 输出

btt 工具作为 blktrace 软件包的一部分提供。它分析 blktrace 输出并显示 I/O 堆栈各个区域所花费的时间，从而更轻松地发现 I/O 子系统中的瓶颈。

由 blktrace 机制跟踪并由 btt 分析的一些重要事件有：

- I/O 事件队列(Q)
- I/O 的发送到驱动程序事件(D)
- 完成 I/O 事件(C)

您可以通过检查事件组合来包含或排除与 I/O 性能问题相关的因素。

要检查每个 I/O 设备的子端口时间，请查看捕获的 blktrace 事件之间的时间。例如，以下命令报告了内核 I/O 堆栈较低部分所花费的总时间(Q2C)，其中包括调度程序、驱动程序和硬件层，平均在等待时间下：

```
$ iostat -x

[...]
Device:         await r_await w_await
vda             16.75    0.97  162.05
dm-0            30.18    1.13  223.45
dm-1             0.14    0.14    0.00
[...]
```

如果设备需要很长时间才能服务请求(D2C)，设备可能会超载，或者发送到该设备的工作负载可能是

子优化。如果在分配给存储设备之前，块 I/O 排队了很长时间(Q2G)，这可能表示正在使用的存储无法为 I/O 负载提供服务。例如，达到 LUN 队列完全条件，并阻止 I/O 分配给存储设备。

查看相邻 I/O 之间的时间可以让大家了解某些类型的瓶颈情况。例如，如果 btt 显示发送到块层(Q2Q)的请求之间的时间大于块层中消耗的请求总数(Q2C)，这表明 I/O 请求和 I/O 子系统之间有空闲时间可能不负责性能问题。

比较相邻 I/O 的 Q2C 值可显示存储服务时间的可变性量。值可以是：

- 非常符合一个小范围，或者
- 分发范围中高度可变，这表示可能存在存储设备侧拥塞问题。

有关这个工具的详情，请查看 btt(1) man page：

```
$ man btt
```

8.2.2.3. 使用 iowatcher 分析 blktrace 输出

iowatcher 工具可以使用 blktrace 输出来随着时间的推移图形 I/O。它侧重于磁盘 I/O 的逻辑块地址(LBA)、每秒吞吐量（以 MB 为单位）、每秒请求数和 I/O 操作数。这有助于识别何时达到设备的操作数秒限制。

有关这个工具的详情请参考 iowatcher(1) man page。

8.2.3. 使用 SystemTap 进行存储监控

红帽企业 Linux 7 SystemTap 初学者指南包括几个可用于分析和监控存储性能的示例脚本。

以下 SystemTap 示例脚本与存储性能相关，在诊断存储或文件系统性能问题时可能很有用。默认情况下，它们安装到 /usr/share/doc/systemtap-client/examples/io 目录中。

disktop.stp

检查每 5 秒读取/写入磁盘的状态，并输出该期间内前十个条目。

iotime.stp

打印读取和写入操作所需的时间，以及读取和写入的字节数。

traceio.stp

根据观察到的累积 I/O 流量（每秒）打印前十个可执行文件。

traceio2.stp

在发生对指定设备的读取和写入时，打印可执行名称和进程标识符。

inodewatch.stp

每次对指定主/次设备上的指定索引节点发生读取和写入时，打印可执行名称和进程标识符。

inodewatch2.stp

每次在指定主/次设备上的指定索引节点上更改属性时，打印可执行名称、进程标识符和属性。

8.3. 固态磁盘

固态磁盘(SSD)使用 NAND 闪存芯片，而不是旋转磁带来存储持久数据。它们为整个逻辑块地址范围内的数据提供恒定的访问时间，而且不会像其对应人员那样产生可观的搜索成本。它们每 GB 存储空间的成本更高，存储密度也更低，但它们的延迟和吞吐量也比 HDD 更低。

性能通常降级为 SSD 上使用的块，即磁盘容量。降级程度因供应商而异，但所有设备在这种情况下都会出现降级情况。启用丢弃行为有助于缓解这种降级。详情请查看 第 8.1.3.3 节 “维护”。

默认 I/O 调度程序和虚拟内存选项适合用于 SSD。

有关 SSD 的更多信息，请参阅《Red Hat Enterprise Linux 7 存储管理指南》中的"固态磁盘部署指南"一章。

SSD 调优注意事项

在配置可影响 SSD 性能的设置时，请考虑以下因素：

I/O 调度程序

大多数 SSD 预期任何 I/O 调度程序都能正常工作。但是，与任何其他存储类型一样，红帽建议采用基准测试来确定给定工作负载的最佳配置。在使用 SSD 时，红帽建议仅更改 I/O 调度程序以测试特定工作负载。有关如何在 I/O 调度程序间切换的步骤，请参阅 / usr/share/doc/kernel-版本/Documentation/block/switching-sched.txt 文件。

从 Red Hat Enterprise Linuxnbsp 7.0 开始，默认的 I/O 调度程序是 Deadline，但 SATA 驱动器除外，它使用 CFQ 作为默认的 I/O 调度程序。为加快存储速度，Deadline 可以提高 CFQ 的性能，从而提高 I/O 性能，而无需具体的调优。有时，默认值不适用于某些磁盘，如 SAS 旋转磁盘。在这种情况下，将 I/O 调度程序更改为 CFQ。

虚拟内存

与 I/O 调度程序一样，虚拟内存(VM)子系统不需要特殊调优。由于 SSD 上的 I/O 的快速性质，尝试关闭 vm_dirty_background_ratio 和 vm_dirty_ratio 设置，因为增加的写外活动通常不会影响对磁盘其他操作的延迟。不过，这种调优可能会产生更多整体 I/O，因此在未进行特定于工作负载的测试的情况下，通常不建议这样做。

swap

SSD 也可以用作交换设备，并且可能会生成良好的 page-out 和 page-in 性能。

8.4. 配置工具

Red Hat Enterprise Linux 提供了很多工具来协助管理员配置存储和文件系统。本节概述了可用的工具，并提供了如何使用它们来解决 Red Hat Enterprise Linux 7 中的 I/O 和文件系统相关性能问题的示例。

8.4.1. 为存储性能配置调优配置集

Tuned 服务提供了很多配置集，旨在提高特定用例的性能。以下配置集对提高存储性能特别有用：

- *latency-performance*

- *throughput-performance（默认）*

要在您的系统上配置配置集，请运行以下命令，将 name 替换为您要使用的配置集的名称。

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

tuned-adm recommend 命令为您的系统推荐适当的配置集。

有关这些配置集或附加配置选项的详情，请参考 第 A.5 节 “tuned-adm”。

8.4.2. 设置默认 I/O 调度程序

如果没有为该设备明确指定其他调度程序，则默认 I/O 调度程序是调度程序。

如果没有指定默认调度程序，则 cfq 调度程序用于 SATA 驱动器，截止时间调度程序 用于所有其他驱动器。如果您按照本节中的说明指定默认调度程序，则默认调度程序将应用到所有设备。

要设置默认的 I/O 调度程序，您可以使用 Tuned 工具，或者手动修改 /etc/default/grub 文件。

红帽建议使用 Tuned 工具在引导的系统上指定默认的 I/O 调度程序。要设置 elevator 参数，启用 磁盘插件。有关 disk 插件的详情，请参考 Tuned 章节中的 第 3.1.1 节 “插件”。

要使用 GRUB 2 修改默认调度程序，请在内核命令行中附加 elevator 参数（在引导时或系统引导时）。您可以使用 Tuned 工具，或者手动修改 /etc/default/grub 文件，如 过程 8.1, “使用 GRUB 2 设置默认 I/O 调度程序” 所述。

过程 8.1. 使用 GRUB 2 设置默认 I/O 调度程序

要在引导的系统中设置默认 I/O 调度程序，并在重启后保留配置：

1. *在 /etc/default/grub 文件中的 GRUB_CMDLINE_LINUX 行中添加 elevator 参数。*

```
# cat /etc/default/grub
...
```

```
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=vg00/lvroot
rd.lvm.lv=vg00/lvswap elevator=noop"
...
```

在 Red Hat Enterprise Linux 7 中，可用的调度程序为 deadline、noop、和 cfq。如需更多信息，请参阅内核文档中的 cfq-iosched.txt 和 deadline-iosched.txt 文件，在安装 kernel-doc 软件包后可用。

2. 创建一个新的配置，并添加了 elevator 参数。

GRUB 2 配置文件的位置在具有 BIOS 固件的系统和 UEFI 系统中的位置不同。使用下列命令之一重新创建 GRUB 2 配置文件：

- 在带有 BIOS 固件的系统中，请使用：

```
# grub2-mkconfig -o /etc/grub2.cfg
```

- 在带有 UEFI 固件的系统中，请使用：

```
# grub2-mkconfig -o /etc/grub2-efi.cfg
```

3. 重启系统以使更改生效。

有关 GNU GRand Unified Bootloader(GRUB 2)版本 2 的更多信息，请参阅 Red Hat Enterprise Linux 7 系统管理员指南的使用 GRUB 2 Boot Loader 章节。

8.4.3. 通用块设备调优参数

本节中列出的通用调优参数位于 /sys/block/sdX/queue/ 目录中。列出的调优参数与 I/O 调度程序调优分开，并适用于所有 I/O 调度程序。

add_random

有些 I/O 事件贡献到 /dev/random 的熵池。如果这些贡献的开销成为可测量，则此参数可以设置

为 0。

iostats

默认值为 1 (enabled)。将 iostats 设置为 0 可禁用为该设备收集 I/O 统计，这会删除 I/O 路径的少量开销。将 iostats 设置为 0 可能会稍提高高性能设备的性能，比如某些 NVMe 固态存储设备。建议启用 iostats，除非供应商为给定的存储模型指定。

如果您禁用 iostats，/proc/diskstats 文件中不再存在该设备的 I/O 统计信息。/sys/diskstats 的内容是监控 I/O 工具的 I/O 信息源，如 sar 或 iostats。因此，如果您为某个设备禁用 iostats 参数，则 I/O 监控工具输出中不再存在该设备。

max_sectors_kb

以 KB 为单位指定 I/O 请求的最大大小。默认值为 512 KB。此参数的最小值由存储设备的逻辑块大小决定。此参数的最大值由 max_hw_sectors_kb 的值决定。

当 I/O 请求大于内部纠删块大小时，某些固态磁盘的性能不佳。确定与系统连接的固态磁盘模型是否如此，请与硬件供应商核对并遵循他们的建议。红帽建议 max_sectors_kb 始终是最佳 I/O 大小和内部清除块大小的倍数。如果参数为零或未由存储设备指定，则使用 logical_block_size 值。

Nomerges

大多数工作负载均可从请求合并中受益。但是，禁用合并对于调试很有用。默认情况下，nomerges 参数设置为 0， 它可启用合并。要禁用简单的单页合并，将 nomerges 设置为 1。要禁用所有类型的合并，将 nomerges 设置为 2。

nr_requests

指定一次可以排队的最大读取和写入请求数。默认值为 128，这意味着在下一个进程请求读取或写入请求之前可以排队 128 个读取请求和 128 个写入请求，以请求读取或写入到 sleep。

对于对延迟敏感的应用，降低此参数的值，并在存储上限制命令队列深度，以便回写 I/O 无法为设备队列填入写入请求。当设备队列填满时，试图执行 I/O 操作的其他进程将处于睡眠状态，直到队列空间可用为止。然后以轮循方式分配请求，防止一个进程持续占用队列中的所有位置。

I/O 调度程序内的最大 I/O 操作数是 nr_requests*2。如前文所述，nr_requests 会单独应用读和写。请注意，nr_requests 只适用于 I/O 调度程序中的 I/O 操作，而不是已经分配给底层设备的 I/O 操作。因此，针对某个设备的 I/O 操作的最大未处理限制是 (nr_requests*2)+(queue_depth)，其中 queue_depth 是 /sys/block/sdN/device/queue_depth，有时也称为 LUN 队列深度。您可以在一个 vg qu-sz 列中看到此未完成的 I/O 操作数，例如 iostat 的输出。

optimal_io_size

有些存储设备通过此参数报告最佳 I/O 大小。如果报告这个值，红帽建议应用程序尽可能在最佳 I/O 大小的倍数中与 I/O 一致。

read_ahead_kb

定义操作系统可在后续读取操作期间提前读取的最大 KB 数。因此，下一顺序读取的内核页面缓存中已存在所需的信息，这提高了读取 I/O 性能。

设备映射器通常受益于高 read_ahead_kb 值。要映射每个设备的 128 KB 是很好的起点，但将 read_ahead_kb 值增加到 4-8 MB 可能会在出现大量文件顺序读取的应用程序环境中提高性能。

rotational

有些固态磁盘无法正确公布其固态状态，而是作为传统旋转磁盘挂载。如果您的固态设备没有自动将其设置为 0，请手动将其设置为调度程序中的不必要的 seek-reducing 逻辑。

rq_affinity

默认情况下，可在不同于发出 I/O 请求的处理器的不同处理器处理 I/O 完成。将 rq_affinity 设置为 1 可禁用此功能，并仅在发出 I/O 请求的处理器中执行完成。这可以提高处理器数据缓存的效率。

scheduler

要为特定存储设备设置调度程序或调度程序首选项顺序，请编辑 /sys/block/devname/queue/scheduler 文件，其中 devname 是您要配置的设备的名称。

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

8.4.4. 调整截止调度程序

当使用 截止时间 时，排队的 I/O 请求将排序为读取或写入批处理，然后调度以增加 LBA 顺序执行。默认情况下，读取批处理优先于写入批处理，因为应用更有可能阻止读取 I/O。处理批处理后，截止时间检查写操作在处理器时间不足，并根据情况调度下一个读取或写入批处理。

以下参数会影响 截止时间调度程序 的行为。

fifo_batch

在单个批处理中发布的读取或写入操作数量。默认值为 16。更高的值可以提高吞吐量，但也会增加延迟。

front_merges

如果您的工作负载永远不会生成前端合并，可以将此可调项设置为 0。但是，除非您衡量了这个检查的开销，否则红帽推荐使用默认值 1。

read_expire

服务调度读取请求的毫秒数。默认值为 500 (0.5 秒)。

write_expire

服务调度写入请求的毫秒数。默认值为 5000 (5 秒)。

writes_starved

在处理写入批处理前可以处理的读取批处理数量。设置这个值越大，读取批处理的首选值越大。

8.4.5. 调优 CFQ 调度程序

使用 CFQ 时，进程被分为三个类：实时、尽力和空闲。在任何最佳工作进程之前，所有实时进程都排期，在任何空闲进程之前调度。默认情况下，进程被归类为最佳工作。您可以使用 ionice 命令手动调整进程的类。

您可以使用下列参数进一步调整 CFQ 调度程序的行为：这些参数通过在 /sys/block/devname/queue/iosched 目录下更改指定的文件来基于每个设备设置。

back_seek_max

CFQ 将执行向后寻道的最大距离（以千字节为单位）。默认值为 16 KB。向后寻寻通常会破坏性能，因此不建议大量价值。

back_seek_penalty

当磁盘头是决定前进还是向后时，用于向后寻找的倍数。默认值为 2。如果磁盘头位置为 1024 KB，且系统中存在 equidistant 请求（例如，1008 KB 和 1040 KB），则 back_seek_penalty 会被应用到向后看到距离和磁盘移动转发。

fifo_expire_async

异步（缓冲写入）请求可以保持未服务的时间长度（以毫秒为单位）。达到此时间后，一个有不足的异步请求将移到分配列表中。默认值为 250 毫秒。

fifo_expire_sync

同步（读取或 O_DIRECT 写入）请求的时间长度（以毫秒为单位）。达到此时间后，将一个无助的同步请求移到分配列表中。默认值为 125 毫秒。

group_idle

这个参数默认设置为 0 （禁用）。当设置为 1 (启用)时，cfq 调度程序会在控制组中发出 I/O 的最后一个进程上闲置 cfq 调度程序。这在使用比例权重 I/O 控制组群以及 slice_idle 设置为 0 时很有用（快速存储）。

group_isolation

这个参数默认设置为 0 （禁用）。当设置为 1 (启用)时，它会在组之间提供更强大的隔离，但会降低吞吐量，因为公平性则应用于随机和顺序工作负载。当 group_isolation 被禁用时（设置为 0），只为后续工作负载提供公平性。如需更多信息，请参阅 /usr/share/doc/kernel-doc-version/Documentation/cgroups/blkio-controller.txt 中安装的文档。

low_latency

默认情况下，此参数设置为 1 (启用)。启用后，cfq 通过为每个进程在设备上发出 I/O 的最大等待时间为 300 毫秒来取代吞吐量。当此参数被设置为 0 （禁用）时，会忽略目标延迟，每个进程都会获得完整的时间分片。

quantum

此参数定义 cfq 一次发送到一个设备的 I/O 请求数，本质上限制队列深度。默认值为 8 个请求。使用的设备可能会支持更大的队列深度，但增加 Quantum 的值也会增加延迟，特别是对于较大的连续写入工作负载。

slice_async

此参数定义分配给每个进程发布异步 I/O 请求的时间片段（以毫秒为单位）。默认值为 40 毫秒。

slice_idle

这个参数指定 cfq 在等待进一步请求时闲置的时长，以毫秒为单位。默认值为 0 (在队列或服务树级别闲置)。默认值为外部 RAID 存储上的吞吐量的理想选择，但可能会在内部非 RAID 存储中降级

吞吐量，因为它会增加寻道操作总数。

slice_sync

此参数定义分配给每个进程发布同步 I/O 请求的时间片段的长度（以毫秒为单位）。默认值为 100 ms。

8.4.5.1. 为快速存储调优 CFQ

对于不会遭受大的看法（如快速外部存储阵列或固态磁盘）的硬件，不建议使用 cfq 调度程序。如果您的用例需要在此存储中使用 cfq，则需要编辑以下配置文件：

- 将 /sys/block/devname/queue/iosched/slice_idle 设置为 0
- 将 /sys/block/devname/queue/iosched/quantum 设置为 64
- 将 /sys/block/devname/queue/iosched/group_idle 设置为 1

8.4.6. 调整 noop 调度程序

noop I/O 调度程序主要用于使用快速存储的 CPU 绑定系统。另外，noop I/O 调度程序通常不是特别的，在对虚拟磁盘执行 I/O 操作时用于虚拟机。

没有特定于 noop I/O 调度程序的可调参数。

8.4.7. 为性能配置文件系统

这部分论述了特定于 Red Hat Enterprise Linux 7 支持的每个文件系统的调优参数。根据以下条件来划分参数：在格式化存储设备时应配置其值，还是挂载格式化的设备时。

如果性能丢失是由文件碎片或资源争用导致的，通常可以通过重新配置文件系统来提高性能。但在某些情况下，可能需要更改应用。在这种情况下，红帽建议联系客户支持以获取帮助。

8.4.7.1. 调优 XFS

本节介绍在格式化和挂载时供 XFS 文件系统使用的一些调优参数。

XFS 的默认格式化和挂载设置适用于大多数工作负载。红帽建议仅在特定的配置更改能让您的工作负载受益时才更改它们。

8.4.7.1.1. 格式化选项

有关这些格式化选项的详情，请查看 man page：

```
$ man mkfs.xfs
```

目录块大小

目录块大小会影响每个 I/O 操作可以检索或修改的目录信息量。目录块大小的最小值是文件系统块大小（默认为 4 KB）。目录块大小的最大值为 64 KB。

在给定的目录块大小中，较大的目录需要比较小的目录更多的 I/O。与目录块大小较小的系统相比，每个 I/O 操作的处理能力也越多。因此，建议针对您的工作负载，尽可能小的目录和目录块大小。

对于不超过 write-heavy 和 read-heavy 工作负载列出的文件系统，红帽建议 表 8.1 “目录块大小的建议最大目录条目” 中列出的目录块大小。

表 8.1. 目录块大小的建议最大目录条目

目录块大小	max. 条目（读-heavy）	max. 条目(write-heavy)
4 KB	100,000–200,000	1,000,000–2,000,000
16 KB	100,000–1,000,000	1,000,000–10,000,000
64 KB	>1,000,000	>10,000,000

有关目录块大小对不同大小的文件系统中读写工作负载的影响的详情，请查看 XFS 文档。

要配置目录块大小，请使用 mkfs.xfs -l 选项。详情请查看 mkfs.xfs 手册页。

分配组

分配组是一种独立结构，用于索引可用空间并在文件系统的某一部分分配索引节点。每个分配组都可以独立修改，允许 XFS 同时执行分配和分配操作，只要并发操作影响不同的分配组。因此，文件系统中可执行的并发操作数等于分配组的数量。但是，由于执行并发操作的能力也受到能够执行操作的处理器数量的限制，红帽建议分配组的数量大于或等于系统中的处理器数量。

单个目录无法由多个分配组同时修改。因此，红帽建议创建和删除大量文件的应用程序不会将所有文件存储在单个目录中。

要配置分配组，请使用 mkfs.xfs -d 选项。详情请查看 mkfs.xfs 手册页。

增长限制

如果您需要在格式化时间后增加文件系统的大小（可以通过添加更多硬件或通过精简配置），您必须仔细考虑初始文件布局，因为在格式化完成后将无法更改分配组大小。

分配组的大小必须根据文件系统的最终容量而非初始容量来确定大小。全增长的文件系统中的分配组数量不应超过数百个，除非分配组的最大大小为 1 TB。因此，对于大多数文件系统，建议允许文件系统的最大增长为初始大小的十倍。

在 RAID 阵列中增大文件系统时，需要格外小心，因为设备大小必须与分配组大小的倍数一致，以便在新添加的存储上正确匹配新的分配组标头。新存储还必须具有与现有存储相同的地理位置，因为 geometry 在格式化后无法更改，因此无法针对同一块设备上不同地理位置的存储进行优化。

索引节点大小和内联属性

如果索引节点有足够的可用空间，XFS 可以直接将属性名称和值写入索引节点。这些内联属性可以比检索独立属性块更快检索和修改，因为不需要额外的 I/O。

默认索引节点大小为 256 字节。根据存储在索引节点中的数据区块指针数量，该属性存储仅可使用大约 100 字节。当您格式化文件系统时，增加索引节点大小可以增加存储属性的可用空间量。

属性名称和属性值都限制为最大 254 字节大小。如果 name 或值的长度超过 254 字节，则该属性将被推送到单独的属性块，而不是内嵌存储。

要配置 inode 参数，请使用 mkfs.xfs -i 选项。详情请查看 mkfs.xfs 手册页。

RAID

如果使用软件 RAID，mkfs.xfs 会自动使用适当的条带单元和宽度配置底层硬件。但是，如果硬件 RAID 使用，可能需要手动配置条带单元和宽度，因为并非所有硬件 RAID 设备都会导出此信息。要配置条带单元和宽度，请使用 mkfs.xfs -d 选项。详情请查看 mkfs.xfs 手册页。

日志大小

待处理的更改将在内存中聚合，直到触发同步事件（在此点写入日志）。日志的大小决定了一次可在进行中的并发修改数。它还决定了内存中可以聚合的最大更改量，以及记录数据写入磁盘的频率。较小的日志强制将数据写回到磁盘比更大的日志更频繁。但是，较大的日志使用更多内存记录待处理的修改，因此内存有限的系统将无法从更大的日志中受益。

当日志与底层条带单元一致时，性能会更好；即，它们以条带单元边界开始和结束。要将日志对齐到条带单元，请使用 mkfs.xfs -d 选项。详情请查看 mkfs.xfs 手册页。

要配置日志大小，请使用以下 mkfs.xfs 选项，将 logsize 替换为日志的大小：

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

详情请查看 mkfs.xfs 手册页：

```
$ man mkfs.xfs
```

日志分条单元

当使用 RAID5 或 RAID6 布局的存储设备在以条带单元边界开始和结束时（与底层条带单元一致）时，其日志写入性能可能会更好。mkfs.xfs 会尝试自动设置适当的日志条带单元，但这取决于导出此信息的 RAID 设备。

如果您的工作负载经常触发同步事件，设置较大的日志条带单元可能会影响性能，因为较小的写入需要添加到日志条带单元的大小，这可能会增加延迟。如果您的工作负载受日志写入延迟约束，红帽建议将日志条带单元设置为 1 块，以便尽可能触发未对齐的日志写入。

支持的最大日志条带单元是最大日志缓冲区大小(256 KB)。因此，底层存储可能具有比日志中配置的更大型的条带单元。在这种情况下，mkfs.xfs 会发出警告，并设置 32 KB 的日志条带单元。

要配置日志条带单元，请使用以下选项之一，其中 N 是用作分条单元的块数，而 size 是 KB 中分条单元的大小。

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

详情请查看 **mkfs.xfs** 手册页：

```
$ man mkfs.xfs
```

8.4.7.1.2. Mount Options

内节点分配

强烈建议用于大于 1 TB 的文件系统。inode64 参数将 XFS 配置为在整个文件系统中分配 inode 和数据。这样可确保索引节点不会在文件系统的开头大量分配，而且不会在文件系统的末尾大量分配数据，从而提高大型文件系统的性能。

日志缓冲区大小和编号

日志缓冲区越大，向日志写入所有更改时所用的 I/O 操作较少。更大的日志缓冲可以改进具有 I/O 密集型工作负载且没有非易失性写入缓存的系统的性能。

日志缓冲区大小使用 logbsize 挂载选项进行配置，并且定义可在日志缓冲区中存储的最大信息量；如果没有设置日志条带单元，缓冲区写入可能会比最大值更短，因此不需要减少同步密集型工作负载的日志缓冲区大小。日志缓冲区的默认大小为 32 KB。最大值为 256 KB 和其他支持的大小为 64 KB、128 KB 或日志分条单元在 32 KB 和 256 KB 之间的两个倍数。

日志缓冲区数量由 logbufs 挂载选项定义。默认值为 8 日志缓冲（最大值），但只能配置两个日志缓冲区。通常不需要减少日志缓冲区的数量，除非在内存绑定系统上无法将内存分配给额外的日志缓冲。减少日志缓冲区数量可能会降低日志性能，特别是在对日志 I/O 延迟敏感的工作负载上。

延迟更改日志

XFS 可以选择在将更改写入日志之前聚合内存中的更改。delaylog 参数允许定期写入日志，而不是每次修改的元数据。这个选项会增加崩溃时可能会丢失的操作数量，并会增加用于跟踪元数据的内存量。但是，它也可以通过一个量级来提高元数据修改速度和可扩展性，当 fsync、fdatasync 或 sync 用于确保将数据和元数据写入磁盘时，它不会降低数据或元数据的完整性。

有关挂载选项的更多信息，请参阅 **man xfs**

8.4.7.2. 调优 ext4

本节介绍 ext4 文件系统在格式和挂载时可用的一些调优参数。

8.4.7.2.1. 格式化选项

内节点表初始化

在非常大的文件系统中初始化所有索引节点可能需要很长时间。默认情况下，初始化过程会被延迟（启用索引节点表初始化）。但是，如果您的系统没有 ext4 驱动程序，则默认情况下禁用 lazy inode 表初始化。它可以通过将 lazy_itable_init 设置为 1 来启用。在这种情况下，内核进程在挂载文件系统后继续初始化文件系统。

本节仅介绍一些在格式时可用的选项。有关进一步格式化参数，请查看 mkfs.ext4 手册页：

```
$ man mkfs.ext4
```

8.4.7.2.2. Mount Options

索引节点表初始化率

启用 lazy 内节点表初始化时，您可以通过为 init_itable 参数指定值来控制初始化的速率。执行后台初始化所需的时间大致等于 1 与此参数的值除以。默认值为 10。

自动文件同步

有些应用程序在重命名现有文件后，或者在截断和重写后无法正确执行 fsync。默认情况下，ext4 会在每个这些操作后自动同步文件。然而，这可能非常耗时。

如果不需要这种同步级别，您可以在挂载时指定 noauto_da_alloc 选项来禁用此行为。如果设置了 auto_da_alloc，应用程序必须显式使用 fsync 来确保数据持久性。

日志 I/O 优先级

默认情况下，日志 I/O 的优先级为 3，它比普通 I/O 的优先级稍高。您可以在挂载时使用 journal_ioprio 参数控制日志 I/O 的优先级。journal_ioprio 范围的有效值从 0 到 7，0 是最高优先级 I/O。

本节仅介绍挂载时可用的部分选项。有关更多挂载选项，请参阅 mount 手册页：

```
$ man mount
```

8.4.7.3. 调优 Btrfs

从红帽企业 Linux 7.0 开始，Btrfs 作为技术预览提供。应始终进行调优，以根据系统当前的工作负载优化系统。有关创建和挂载选项的详情，请参考《Red Hat Enterprise Linux 7 存储管理指南》中关于 Btrfs 的章节。

数据压缩

默认的压缩算法是 zlib，但特定的工作负载可能会给出更改压缩算法的原因。例如，如果您有一个包含大量文件 I/O 的线程，则最好使用 lzo 算法。挂载时的选项有：

- compress=zlib - 默认的选项，具有高压缩率，对于较旧的内核而言是安全的。
- compress=lzo - 压缩速度快，但比 zlib 更低.
- compress=no - 禁用压缩.
- press-force=方法 - 即使对于不压缩效果的文件（如视频和磁盘映像）也启用压缩。可用的方法为 zlib 和 lzo。

仅压缩在挂载选项中添加后创建或更改的文件。要压缩现有文件，请在使用 zlib 或 lzo 替换方法后运行以下命令：

```
$ btrfs filesystem defragment -cmethod
```

要使用 lzo 重新压缩文件，请运行：

```
$ btrfs filesystem defragment -r -v -clzo /
```

8.4.7.4. 调整 GFS2

本节介绍了 GFS2 文件系统在格式和挂载时可用的一些调优参数。

目录间进行

GFS2 挂载点顶级目录中创建的所有目录都会自动占用空间，以减少碎片并提高这些目录中的写入速度。对于另一个目录（如顶级目录），请将该目录标记为 T 属性，如下所示，将 dirname 替换为

您想要空间的目录的路径：

```
# chattr +T dirname
```

chattr 作为 e2fsprogs 软件包的一部分提供。

减少争用

GFS2 使用一种全局锁定机制，需要集群节点之间的通信。多个节点间的文件和目录争用会降低性能。您可以通过最小化在多个节点间共享的文件系统区域来最小化跨缓存无效的风险。

第 9 章 网络

网络子系统由多个具有敏感连接的不同部分组成。因此，红帽企业 Linux 7 联网旨在为大多数工作负载提供最佳性能，并自动优化其性能。因此，通常不需要手动调优网络性能。本章讨论可对功能网络系统进行进一步的优化。

网络性能问题有时是硬件故障或基础架构故障造成的。解决这些问题超出了本文档的范围。

9.1. 注意事项

为了做出良好的调优决策，您需要深入了解红帽企业 Linux 中的数据包接收。本节介绍如何接收和处理网络数据包，以及如何在何处发生潜在瓶颈。

发送到 Red Hat Enterprise Linux 系统的数据包由网络接口卡(NIC)接收，并放置在内部硬件缓冲或环形缓冲区中。然后 NIC 发送硬件中断请求，提示创建软件中断操作来处理中断请求。

作为软件中断操作的一部分，数据包从缓冲区传输到网络堆栈。根据数据包和网络配置，数据包随后被转发、丢弃或传递到应用的套接字接收队列，然后从网络堆栈中删除。此过程将继续，直到 NIC 硬件缓冲区中没有数据包，或者传输了特定数量的数据包（在 /proc/sys/net/core/dev_weight中指定）。

红帽客户门户网站中提供的 Red Hat Enterprise Linux 网络性能调优指南 包含有关 Linux 内核的数据包接收的信息，并涵盖 NIC 调优的以下区域： softirq misses (netdev budget), tuned tuning daemon, numad NUMA 守护进程, CPU 电源状态, 中断平衡, 暂停帧, 中断 coalescence, 适配器队列(netdev backlog), 适配器 RX 和 TX 缓冲区、适配器 TX 队列、模块卸载、Jumbo Frames、TCP 和 UDP 协议调整以及 NUMA 本地功能。

9.1.1. Tune 之前

网络性能问题通常是硬件故障或基础架构故障造成的。红帽强烈建议您在开始调优网络堆栈之前验证您的硬件和基础架构是否按预期工作。

9.1.2. Packet Reception 中的瓶颈

虽然网络堆栈在很大程度上是自我优化的，但网络数据包处理期间存在多个点，它们可能会成为瓶颈并降低性能。

NIC 硬件缓冲或环缓冲

如果丢弃了大量数据包，硬件缓冲区可能会成为瓶颈。有关监控系统中丢弃数据包的详情请参考 第 9.2.4 节 “ethtool”。

硬件或软件中断队列

中断可能会增加延迟和处理器争用。有关处理器如何处理中断的详情请参考 第 6.1.3 节 “中断请求(IRQ)处理”。有关如何在系统中监控中断处理的详情请参考 第 6.2.3 节 “/proc/interrupts”。有关影响中断处理的配置选项，请参阅 第 6.3.7 节 “在 AMD64 和 Intel 64 中设置 Interrupt Affinity”。

应用程序的套接字接收队列

应用程序接收队列中的瓶颈由大量没有复制到请求应用程序的数据包表示，或者 /proc/net/snmp 中的 UDP 输入错误(错误)增加。有关监控您的系统以了解这些错误的详情，请参考 第 9.2.1 节 “SS” 和 第 9.2.5 节 “/proc/net/snmp”。

9.2. 监控和诊断性能问题

红帽企业 Linux 7 提供了很多工具，这些工具可用于监控系统性能并诊断与网络子系统相关的性能问题。本节概述了可用的工具，并提供了如何使用它们来监控和诊断网络相关性能问题的示例。

9.2.1. SS

ss 是一种命令行工具，可打印有关套接字的统计信息，允许管理员随着时间的推移评估设备性能。默认情况下，ss 列出了打开的非侦听已建立连接的 TCP 套接字，但提供了多个有用的选项，以帮助管理员过滤特定套接字的统计信息。

红帽建议在 Red Hat Enterprise Linux 7 中使用 ss 而不是 netstat。

s 由 iproute 软件包提供。如需更多信息，请参阅 man page:

```
$ man ss
```

9.2.2. ip

ip 工具可让管理员管理和监控路由、设备、路由策略和隧道。ip monitor 命令可以持续监控设备、地址和路由的状态。

IP 由 iproute 软件包提供。有关使用 ip 的详情，请查看 man page：

```
$ man ip
```

9.2.3. dropwatch

dropwatch 是一个交互式工具，用于监控和记录内核丢弃的数据包。

如需更多信息，请参阅 dropwatch 手册页：

```
$ man dropwatch
```

9.2.4. ethtool

ethtool 工具允许管理员查看和编辑网络接口卡设置。它可用于观察某些设备的统计信息，如该设备丢弃的数据包数。

您可以使用 ethtool -S 和您要监控的设备名称查看指定设备计数器的状态。

```
$ ethtool -S devname
```

如需更多信息，请参阅 man page:

```
$ man ethtool
```

9.2.5. /proc/net/snmp

/proc/net/snmp 文件显示 snmp 代理用于 IP、ICMP、TCP 和 UDP 监控和管理的数据。定期检查此文件可帮助管理员识别异常值，从而识别潜在的性能问题。例如，/proc/net/snmp 中的 UDP 输入错误 (InErrors)的增加可能会表示套接字接收队列中的瓶颈。

9.2.6. 使用 SystemTap 进行网络监控

红帽企业 Linux 7 SystemTap Beginner 指南包括几个可用于分析和监控网络性能的示例脚本。

以下 SystemTap 示例脚本与网络相关，在诊断网络性能问题时可能很有用。默认情况下，它们安装到 /usr/share/doc/systemtap-client/examples/network 目录中。

nettop.stp

每 5 秒，打印一个进程（进程标识符和命令）列表，其中包含发送和接收的数据包数，以及进程在该间隔期间发送和接收的数据量。

socket-trace.stp

引入 Linux 内核 net/socket.c 文件中的每个功能，并打印 trace 数据。

dropwatch.stp

每 5 秒，打印在内核中的位置释放的套接字缓冲区数量。使用 --all-modules 选项查看符号名称。

latencytap.stp 脚本记录了不同类型的延迟对一个或多个进程的影响。它每 30 秒打印延迟类型列表，按进程或进程等待的总时间降序排列。这可用于识别存储和网络延迟的原因。红帽建议将 --all-modules 选项与该脚本一起使用来更好地映射延迟事件。默认情况下，此脚本安装到 /usr/share/doc/systemtap-client-版本/examples/profiling 目录中。

如需更多信息，请参阅《红帽企业 Linux 7 SystemTap 入门指南》。

9.3. 配置工具

Red Hat Enterprise Linux 提供了很多工具来协助管理员配置系统。本节概述了可用的工具，并提供了有关如何使用它们来解决 Red Hat Enterprise Linux 7 中网络相关性能问题的示例。

但是，务必要记住，网络性能问题有时是硬件故障或基础架构故障造成的。红帽强烈建议您在使用这些工具调优网络堆栈之前验证您的硬件和基础架构是否按预期工作。

此外，与重新配置网络子系统相比，更改应用可以更好地解决一些网络性能问题。通常最好将应用配置为执行频繁的 posix 调用，即使这意味着在应用空间中排队数据，因为这允许根据需要灵活存储数据并交换出内存。

9.3.1. 网络性能调优配置集

Tuned 服务提供多个不同配置集，以便在多个特定用例中提高性能。以下配置文件对于提高网络性能非常有用：

- latency-performance
- network-latency
- network-throughput

有关这些配置集的详情请参考 第 A.5 节 “tuned-adm”。

9.3.2. 配置硬件缓冲器

如果硬件缓冲区丢弃了大量数据包，则存在许多潜在的解决方案。

减慢输入流量

过滤传入流量，减少加入的多播组数量，或减少广播流量以降低队列填充的速度。有关如何过滤传入流量的详情，请查看 Red Hat Enterprise Linux 7 安全指南。有关多播组的详情，请查看 Red Hat Enterprise Linux 7 集群文档。有关广播流量的详情，请查看 Red Hat Enterprise Linux 7 系统管理员指南，或与您要配置的设备相关的文档。

调整硬件缓冲区队列的大小

通过增大队列的大小来减少丢弃的数据包数量，使其不会像轻松地溢出。您可以使用 ethtool 命令修改网络设备的 rx/tx 参数：

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

更改队列的排空率

设备权重指的是设备一次可以接收的数据包数（在一个计划的处理器访问中）。您可以通过增加设备权重来增加排空队列的速度，这由 dev_weight 参数控制。可以通过更改 /proc/sys/net/core/dev_weight 文件的内容暂时更改此参数，也可以使用 sysctl （由 procps-ng 软件包提供）永久更改。

更改队列的排空率通常是降低网络性能的最简单方法。但是，增加设备一次接收的数据包数量会使用额外的处理器时间，在此期间无法调度其他进程，因此可能会导致其他性能问题。

9.3.3. 配置中断队列

如果分析揭示了高延迟，则您的系统可能会受益于基于轮询的数据包接收，而非基于中断的数据包接收。

9.3.3.1. 配置总线轮询

繁忙的轮询允许套接字层代码轮询网络设备的接收队列，并禁用网络中断，从而降低网络接收路径的延迟。这删除了中断和结果上下文切换造成的延迟。不过，它也会增加 CPU 利用率。忙碌轮询也会阻止 CPU 休眠，这可能会产生额外的功耗。

默认情况下禁用忙碌轮询。要在特定套接字上启用繁忙的轮询，请执行以下操作：

- 将 sysctl.net.core.busy_poll 设置为 0 以外的值。此参数控制在设备队列中等待数据包以进行套接字轮询并选择的微秒数。红帽建议值为 50。
- 将 SO_BUSY_POLL 套接字选项添加到套接字。

要在全局范围内启用忙碌轮询，还必须将 sysctl.net.core.busy_read 设置为 0 以外的值。此参数控制等待设备队列中套接字读取的数据包的微秒数。它还设置 SO_BUSY_POLL 选项的默认值。对于少量插槽，红帽建议为 50，对于大量插槽，红帽建议值为 100。对于非常大的插槽（超过数百个），请使用 epoll。

以下驱动程序支持忙碌轮询行为：Red Hat Enterprise Linux 7 也支持这些驱动程序。

- bnx2x
- be2net
- ixgbe
- mlx4

- *myri10ge*

从红帽企业 Linux 7.1 开始，您还可以运行以下命令来检查特定的设备是否支持繁忙的轮询：

```
# ethtool -k device | grep "busy-poll"
```

如果此操作返回 busy-poll: on [fixed]，则设备上提供忙碌轮询。

9.3.4. 配置套接字接收队列

如果分析表明数据包因为套接字队列的排空率太慢而被丢弃，则可以通过几种方法来缓解结果的性能问题。

降低传入流量的速度

通过在到达队列前过滤或丢弃数据包，或通过降低设备的权重来降低队列填充的速度。

增加应用的套接字队列的深度

如果套接字队列接收的突发流量有限，增加套接字队列的深度以匹配突发流量的大小可能会阻止数据包被丢弃。

9.3.4.1. 减少即将到来的交通量

过滤传入流量或降低网络接口卡的设备权重，以减慢传入的流量速度。有关如何过滤传入流量的详情，请查看 Red Hat Enterprise Linux 7 安全指南。

设备权重指的是设备一次可以接收的数据包数（在一个计划的处理器访问中）。设备权重由 dev_weight 参数控制。可以通过更改 /proc/sys/net/core/dev_weight 文件的内容暂时更改此参数，也可以使用 sysctl （由 procps-ng 软件包提供）永久更改。

9.3.4.2. 增加队列拒绝

提高应用程序套接字队列的深度通常是提高套接字队列排空率的最简单方法，但不太可能是长期解决方案。

要增大队列的深度，请通过以下更改之一增加套接字接收缓冲的大小：

增加 /proc/sys/net/core/rmem_default 的值

此参数控制套接字使用的接收缓冲区的默认大小。这个值必须小于或等于 /proc/sys/net/core/rmem_max 的值。

使用 setsockopt 配置更大的 SO_RCVBUF 值

此参数控制套接字接收缓冲区的最大字节大小。使用 getsockopt 系统调用来确定缓冲区的当前值。如需更多信息，请参阅 socket(7) 手册页。

9.3.5. 配置接收扩展(RSS)

接收扩展(RSS)（也称为多队列接收）可在多个基于硬件的接收队列之间分发网络接收处理，允许多个 CPU 处理入站网络流量。RSS 可用于减轻因单个 CPU 过载导致的接收中断处理方面的瓶颈，并降低网络延迟。

要确定您的网络接口卡是否支持 RSS，请检查多个中断请求队列是否与 /proc/interrupts 中的接口关联。例如，如果您对 p1p1 接口感兴趣：

```
# 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
```

前面的输出显示 NIC 驱动程序为 p1p1 接口创建了 6 个接收队列(p1p1-0 到 p1p1-5)。它还显示每个队列处理了多少个中断，以及服务中断的 CPU。在这种情况下，有 6 个队列，因为默认情况下，这个特定 NIC 驱动程序会为每个 CPU 创建一个队列，此系统具有 6 个 CPU。在 NIC 驱动程序中，这是相当常见的模式。

另外，您可以在载入网络驱动程序后检查 ls -1 /sys/devicesAttr/device_pci_address/msi_irqs 的输出。例如，如果您对带有 PCI 地址 0000:01:00.0 的设备感兴趣，您可以使用以下命令列出该设备的中断请求队列：

```
# ls -1 /sys/devices/*/*/0000:01:00.0/msi_irqs
101
102
```

```
103
104
105
106
107
108
109
```

默认启用 RSS。RSS 的队列（或应处理网络活动的 CPU）数量在适当的网络设备驱动程序中配置。对于 bnx2x 驱动程序，它在 num_queues 中配置。对于 sfc 驱动程序，它在 rss_cpus 参数中配置。无论如何，它通常都在 /sys/class/net/设备/queues/queues/rx-queue 中配置，其中 device 是网络设备的名称（如 eth1），rx-queue 是适当的接收队列的名称。

在配置 RSS 时，红帽建议将队列数量限制为每个物理 CPU 内核一个队列。超线程通常在分析工具中作为单独的核心来表示，但为所有核心（如超线程）配置队列并不对网络性能有用。

启用后，RSS 根据每个 CPU 的处理量在可用 CPU 之间均匀分配网络处理。但是，您可以使用 ethtool --show-rxfh-indir 和 --set-rxfh-indir 参数修改网络活动的分布方式，并将某些类型的网络活动视为比其他更重要。

irqbalance 守护进程可以与 RSS 结合使用，以减少跨节点内存传输和缓存行退回的可能性。这降低了处理网络数据包的延迟。

9.3.6. 配置接收数据包控制(RPS)

接收数据包声明(RPS)与 RSS 类似，因为它用于将数据包定向到特定的 CPU 以进行处理。但是，RPS 在软件级别上实施，有助于防止单个网络接口卡的硬件队列成为网络流量的瓶颈。

与基于硬件的 RSS 相比，RPS 具有几个优点：

- RPS 可以和任何网络接口卡一起使用。
- 可以在 RPS 中添加软件过滤器，以处理新协议。
- RPS 不会增加网络设备的硬件中断率。然而，它确实引入了处理器间中断。

RPS 会针对每个网络设备和接收队列进行配置，在 /sys/class/net/设备/queues/rx-queue/rps_cpus

文件中，其中 device 是网络设备的名称（如 eth0），rx-queue 是适当的接收队列的名称（如 rx-0）。

rps_cpus 文件的默认值为 0。这会禁用 RPS，因此处理网络中断的 CPU 也处理数据包。

要启用 RPS，请使用应处理指定网络设备的数据包和接收队列的 CPU 配置适当的 rps_cpus 文件。

rps_cpus 文件使用以逗号分隔的 CPU 位映射。因此，要允许 CPU 处理接口上接收队列的中断，请将它们在位图中的位置值设置为 1。例如，要处理 CPU 0、1、2 和 3 的中断，请将 rps_cpus 的值设置为 f，这是 15 的十六进制值。在二进制表示中，15 为 00001111 (1+2+4+8)。

对于具有单传输队列的网络设备，可以通过将 RPS 配置为使用同一内存域中的 CPU 来实现最佳性能。在非 NUMA 系统上，这意味着可以使用所有可用的 CPU。如果网络中断率非常高，除处理网络中断的 CPU 外，也可以提高性能。

对于具有多个队列的网络设备，配置 RPS 和 RSS 时通常没有好处，因为 RSS 被配置为默认将 CPU 映射到每个接收队列。但是，如果硬件队列比 CPU 少，并且 RPS 配置为在同一内存域中使用 CPU，则 RPS 可能仍然很有用。

9.3.7. 配置接收流(RFS)

接收流速(RFS)扩展 RPS 行为，以提高 CPU 缓存命中率，从而减少网络延迟。当 RPS 仅根据队列长度转发数据包时，RFS 使用 RPS 后端来计算最合适的 CPU，然后根据使用数据包的应用程序的位置转发数据包。这提高了 CPU 缓存效率。

默认情况下禁用 RFS。要启用 RFS，您必须编辑两个文件：

/proc/sys/net/core/rps_sock_flow_entries

将此文件的值设置为预期的并发活跃连接的最大数量。我们推荐在服务器负载中取 32768。输入的所有值在实践中均被舍入到 2 最接近的指数。

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

使用您要配置的 网络设备 的名称替换 device （例如 eth0）和 rx-queue 替换为您要配置的接收队列（例如 rx-0）。

将此文件的值设置为 rps_sock_flow_entries 的值，其中 N 是设备上接收队列的数量。例如，

如果 rps_flow_entries 设为 32768，并且有 16 个配置的接收队列，则 rps_flow_cnt 应设置为 2048。对于单队列设备，rps_flow_cnt 的值与 rps_sock_flow_entries 的值相同。

从单个发送器接收的数据不会发送到多个 CPU。如果单个发送方收到的数据量大于单个 CPU 可处理的数据量，请配置更大的帧大小以减少中断数量，从而减少 CPU 处理工作的数量。或者，考虑 NIC 卸载选项或更快速的 CPU。

考虑将 numactl 或 taskset 与 RFS 结合使用，以将应用固定到特定的内核、插槽或 NUMA 节点。这有助于防止数据包被不按顺序处理。

9.3.8. 配置加速的 RFS

加快 RFS 通过添加硬件辅助来提升 RFS 速度。与 RFS 一样，数据包根据使用数据包的应用程序的位置进行转发。但与传统的 RFS 不同，数据包直接发送到使用数据的线程本地的 CPU：执行应用的 CPU 或缓存层次结构中该 CPU 本地的 CPU。

只有满足以下条件时，加速的 RFS 才可用：

- 加速的 RFS 必须由网络接口卡支持。导出 ndo_rx_flow_steer() netdevice 功能的卡支持加速 RFS。
- 必须启用 ntuple 过滤。

满足这些条件后，根据传统的 RFS 配置自动停用 CPU 到队列映射。也就是说，CPU 到队列映射将根据驱动程序为每个接收队列配置的 IRQ 影响而降低。有关配置传统 RFS 的详情，请参阅 第 9.3.7 节 “配置接收流(RFS)”。

红帽建议使用 RFS 最多使用加速的 RFS，且网络接口卡支持硬件加速。

附录 A. 工具参考

本附录提供了可用于调整性能的红帽企业 Linux 7 中各种工具的快速参考。有关您的工具，请参见相关 man page，了解完整的、最新的详细参考材料。

A.1. IRQBALANCE

irqbalance 是一种命令行工具，可在处理器之间分发硬件中断以提高系统性能。默认情况下，它作为守护进程运行，但只能使用 --oneshot 选项运行一次。

下列参数可用于提高性能：

--powerthresh

设置 CPU 进入 powerave 模式前可以闲置的 CPU 数量。如果超过阈值的 CPU 超过阈值，则平均 softirq 工作负载低于 1 个标准 deviation，且没有 CPU 比平均值多于一个标准 deviation，并有多个 irq 个分配给它们的 irq，则会将 CPU 置于节能模式。在 powersave 模式中，CPU 不是 irq 平衡的一部分，因此不会意外中断。

--hintpolicy

决定如何处理 irq 内核关联性提示。有效值是 准确的 （始终应用irq 关联性 hint）、subset (irq 为 balanced，但分配的对象是关联性提示的子集)或 ignore （完全忽略irq affinity hint）。

--policyscript

定义为每个中断请求执行的脚本的位置，使用设备路径和 irq 编号作为参数传递，以及 irqbalance 期望的零退出代码。定义的脚本可以指定零个或更多键值对，以指导 irqbalance 在管理传递的 irq 中。

以下识别为有效的键值对：

ban

有效值为 true （不包括传递的 irq from balancing）或 false （这个 irq的平衡性能）。

balance_level

允许用户覆盖传递的 irq 的平衡级别。默认情况下，平衡级别基于拥有 irq 的设备的 PCI 设备类。有效值为 none、package、cache 或 core。

numa_node

允许用户覆盖被视为通过 irq 的本地 NUMA 节点。如果没有在 ACPI 中指定有关本地节点的信息，则所有节点中的设备都会被视为一样。有效值为整数（从 0 开始）来标识特定 NUMA 节点，以及 -1， 它指定 irq 应被视为来自所有节点的 equidistant。

--banirq

带有指定中断请求号的中断添加到被禁止的中断列表中。

您还可以使用 IRQBALANCE_BANNED_CPUS 环境变量指定 irqbalance 忽略的 CPU 掩码。

详情请查看 man page:

```
$ man irqbalance
```

A.2. ETHTOOL

ethtool 工具允许管理员查看和编辑网络接口卡设置。它可用于观察某些设备的统计信息，如该设备丢弃的数据包数。

ethtool、其选项及其用法完全记录在 man page 中。

```
$ man ethtool
```

A.3. SS

ss 是一种命令行工具，可打印有关套接字的统计信息，允许管理员随着时间的推移评估设备性能。默认情况下，ss 列出了打开的非侦听已建立连接的 TCP 套接字，但提供了多个有用的选项，以帮助管理员过滤特定套接字的统计信息。

一个常用的命令是 ss -tmpie，它显示所有 TCP 套接字（t、内部 TCP 信息(i)、套接字内存用量(m)、使用套接字(p)的进程以及详细的套接字信息(i)。

红帽建议在 Red Hat Enterprise Linux 7 中使用 ss 而不是 netstat。

s 由 iproute 软件包提供。如需更多信息，请参阅 man page:

```
$ man ss
```

A.4. TUNED

TuneD 是一个调优守护进程，可通过设置调优配置文件来调整操作系统，以便在特定工作负载下更好地执行。它还可以配置为响应 CPU 和网络使用的变化，并调整设置以提高活动设备的性能并降低不活动设备的功耗。

要配置动态性能优化行为，请编辑 /etc/tuned/tuned-main.conf 文件中的 dynamic_tuning 参数。然后 tuned 会定期分析系统统计信息，并使用它们更新您的系统调优设置。您可以使用 update_interval 参数在这些更新之间配置时间间隔（以秒为单位）。

有关 tuned 的详情，请查看 man page：

```
$ man tuned
```

A.5. TUNED-ADM

tuned-adm 是一个命令行工具，可让您在 Tuned 配置集间切换以提高特定用例的性能。它还提供了 tuned-adm recommend 子命令，用于评估您的系统并输出推荐的调优配置文件。

从 Red Hat Enterprise Linux 7 开始，Tuned 包含了作为启用或禁用调优配置文件的一部分运行任何 shell 命令的功能。这可让您使用尚未集成到 Tuned 的功能来扩展 Tuned 配置集。

Red Hat Enterprise Linux 7 还在配置集定义文件中提供 include 参数，允许您在现有配置集中基于自己的 Tuned 配置集。

以下调优配置集由 Tuned 提供，在 Red Hat Enterprise Linux 7 中被支持。

throughput-performance

侧重于提高吞吐量的服务器配置文件。这是默认配置集，建议用于大多数系统。

此配置集通过设置 intel_pstate 和 min_perf_pct =100 来优先选择性能而不是节能。它启用了透明大内存页，并使用 cpupower 设置 性能 cpufreq governor。它还将 kernel.sched_min_granularity_ns 设置为 10 101s，kernel.sched_wakeup_granularity_ns 设置为 15 crius，将 vm.dirty_ratio 设置为 40%。

latency-performance

侧重于降低延迟的服务器配置文件。建议对可通过 c-state tuning 和透明巨页提高 TLB 效率的对延迟敏感的工作负载进行。

此配置集通过设置 intel_pstate 和 max_perf_pct =100 来优先选择性能而不是节能。它启用了透明大内存页，使用 cpupower 设置 性能 cpufreq governor，并请求 cpu_dma_latency 值 1。

network-latency

侧重于降低网络延迟的服务器配置文件。

此配置集通过设置 intel_pstate 和 min_perf_pct =100 来优先选择性能而不是节能。它禁用透明大内存页和自动 NUMA 平衡。它还使用 cpupower 设置 性能 cpufreq governor，并请求 cpu_dma_latency 值 1。它还将 busy_read 和 busy_poll 的次数设置为 50 mvapich，并将 tcp_fastopen 设置为 3。

network-throughput

侧重于提高网络吞吐量的服务器配置文件。

该配置集通过设置 intel_pstate 和 max_perf_pct =100 并增加内核网络缓冲区大小，将性能优于节能性能。它启用了透明大内存页，并使用 cpupower 设置 性能 cpufreq governor。它还将 kernel.sched_min_granularity_ns 设置为 10 101s，kernel.sched_wakeup_granularity_ns 设置为 15 crius，将 vm.dirty_ratio 设置为 40%。

virtual-guest

该配置集侧重于优化红帽企业 Linux 7 虚拟机和 VMware 客户机的性能。

此配置集通过设置 intel_pstate 和 max_perf_pct =100 来优先选择性能而不是节能。它还减少了虚拟内存的交换性。它启用了透明大内存页，并使用 cpupower 设置 性能 cpufreq governor。它还

将 kernel.sched_min_granularity_ns 设置为 10 101s，kernel.sched_wakeup_granularity_ns 设置为 15 crius，将 vm.dirty_ratio 设置为 40%。

virtual-host

侧重于优化红帽企业 Linux 7 虚拟化主机性能的配置文件.

此配置集通过设置 intel_pstate 和 max_perf_pct =100 来优先选择性能而不是节能。它还减少了虚拟内存的交换性。这个配置集启用了透明大内存页，并更频繁地将脏页面写入磁盘。它使用 cpupower 设置 性能 cpufreq governor。它还将 kernel.sched_min_granularity_ns 设置为 10 101s，kernel.sched_wakeup_granularity_ns 设置为 15 crius，kernel.sched_migration_cost 设置为 5 unmarshals， vm.dirty_ratio 设置为 40%。

cpu-partitioning

cpu-partitioning 配置集将系统 CPU 划分为隔离和内务 CPU。为减少隔离 CPU 上的 jitter 和中断，配置集清除了与用户空间进程、可移动内核线程、中断处理程序和内核计时器隔离的 CPU。

内务 CPU 可以运行所有服务、shell 进程和内核线程。

您可以在 /etc/tuned/ cpu-partitioning -variables.conf 文件中配置 cpu-partitioning 配置集。配置选项为：

isolated_cores=cpu-list

列出要隔离的 CPU。隔离 CPU 的列表用逗号分开，用户可以指定范围。您可以使用短划线（如 3-5 ）指定范围。此选项是必需的。此列表中缺少的任何 CPU 都会自动被视为内务 CPU。

no_balance_cores=cpu-list

列出内核在系统范围范围的进程负载均衡期间没有考虑的 CPU。此选项是可选的。这通常与 isolated_cores 相同。

有关 cpu-partitioning 的详情，请查看 tuned-profiles-cpu-partitioning(7) man page。

有关 tuned-adm 提供的节能配置文件的详情， 请查看 Red Hat Enterprise Linux 7 电源管理指南。

有关使用 tuned-adm 的详情，请查看 man page：

```
$ man tuned-adm
```

A.6. PERF

perf 工具提供了很多有用的命令，其中一些在本节中列出。有关 perf 的详情，请查看 Red Hat Enterprise Linux 7 开发人员指南，或参阅 man page。

perf stat

此命令提供常见性能事件的整体统计信息，包括执行的指令和消耗的时钟周期。您可以使用 option 标志来收集默认测量事件以外的事件统计信息。从 Red Hat Enterprise Linux 6.4 开始，可以使用 perf stat 根据一个或多个指定的控制组(cgroups)过滤监控。

如需更多信息，请参阅 man page:

```
$ man perf-stat
```

perf 记录

此命令将性能数据记录到文件中，稍后可以使用 perf report 进行分析。详情请查看 man page:

```
$ man perf-record
```

perf 报告

此命令从文件中读取性能数据，并分析记录的数据。详情请查看 man page:

```
$ man perf-report
```

perf 列表

此命令会列出特定计算机上可用的事件。这些事件因系统的性能监控硬件和软件配置而异。如需更多信息，请参阅 man page:

```
$ man perf-list
```

perf 顶部

此命令执行与 top 工具类似的功能。它实时生成并显示性能计数器配置文件。如需更多信息，请参阅 man page:

```
$ man perf-top
```

perf trace

此命令执行与 strace 工具类似的函数。它监控指定线程或进程使用的系统调用，以及该应用收到的所有信号。还有其他 trace 目标；请参阅 man page 获得完整列表：

```
$ man perf-trace
```

A.7. PERFORMANCE CO-PILOT(PCP)

Performance Co-Pilot(PCP)提供了大量命令行工具、图形工具和库。有关这些工具的更多信息，请参阅相应的 man page。

表 A.1. 在红帽企业 Linux 7 中与 Performance Co-Pilot 发布的系统服务

名称	描述
pmcd	Performance Metric Collector Daemon(PMCD)。
pmie	性能指标参考引擎.
pmlogger	性能指标日志记录器。
pmmgr	根据零个或多个配置目录，为一组已发现的本地和远程主机管理 PCP 守护进程集合，运行 Performance Metric Collector Daemon(PMCD)。
pmproxy	Performance Metric Collector Daemon(PMCD)代理服务器。
pmwebd	使用 HTTP 协议将 Performance Co-Pilot 客户端 API 的子集绑定到 RESTful Web 应用。

表 A.2. 在红帽企业 Linux 7 中与 Performance Co-Pilot 发布的工具

名称	描述
pcp	显示 Performance Co-Pilot 安装的当前状态。
pmatop	从性能角度显示最重要的硬件资源的系统级别：CPU、内存、磁盘和网络。
pmchart	绘制 Performance Co-Pilot 工具提供的性能指标值。

名称	描述
pmclient	使用性能指标应用程序编程接口(PMAPI)显示高级别系统性能指标。
pmcollectl	从实时系统或 Performance Co-Pilot 存档文件收集和显示系统级数据。
pmconfig	显示配置参数的值。
pmdbg	显示可用的 Performance Co-Pilot 调试控制标志及其值。
pmdiff	比较给定时间窗口中一个或多个存档中每个指标的平均值，以了解在搜索性能回归时可能感兴趣的更改。
pmdumplog	显示 Performance Co-Pilot 归档文件中的控制、元数据、索引和状态信息。
pmdumptext	输出实时或从 Performance Co-Pilot 归档收集的性能指标值。
pmerr	显示可用的 Performance Co-Pilot 错误代码及其对应的错误消息。
pmfind	查找网络上的 PCP 服务。
pmie	定期评估一组算术、逻辑和规则表达式的推理引擎。指标从实时系统或从 Performance Co-Pilot 归档文件收集。
pmieconf	显示或设置可配置的 **pmie** 变量。
pminfo	显示性能指标信息。指标从实时系统或从 Performance Co-Pilot 归档文件收集。
pmiostat	报告 SCSI 设备的 I/O 统计信息（默认情况下）或 设备**映射器**设备 （使用 **-x dm** 选项）。
pmlc	交互式配置活跃的 **pmlogger** 实例。
pmlogcheck	在 Performance Co-Pilot 归档文件中识别无效的数据。
pmlogconf	创建和修改 **pmlogger** 配置文件。
pmloglabel	验证、修改或修复 Performance Co-Pilot 存档文件的标签。
pmlogsummary	计算 Performance Co-Pilot 存档文件中存储的性能指标的统计信息。
pmprobe	决定性能指标的可用性。
pmrep	报告所选、易于定制的性能指标值。
pmsocks	允许通过防火墙访问 Performance Co-Pilot 主机。

名称	描述
pmstat	定期显示系统性能的简短摘要。
pmstore	修改性能指标的值。
pmtrace	为跟踪性能指标域代理(PMDA)提供命令行界面。
pmval	显示性能指标的当前值。

表 A.3. XFS 的 PCP 指标组

指标组	提供的指标
xfs.*	常规 XFS 指标，包括读取和写入操作计数、读取和写入字节计数。与计数器一起清空、群集化和群集失败次数的内节点数。
xfs.allocs.* xfs.alloc_btree.*	有关在文件系统中分配对象的指标范围，其中包括扩展数目和块创建/释放。分配树查找，并与从 btree 中的扩展记录创建和删除进行比较。
xfs.block_map.* xfs.bmap_tree.*	指标包括块映射读/写和块删除的数量，用于插入、删除和查找的扩展列表操作。另外，用于从 blockmap 中比较、查找、插入和删除操作的操作计数器。
xfs.dir_ops.*	XFS 文件系统中的目录操作计数器，用于创建、条目删除、"getdent"操作计数.
xfs.transactions.*	元数据事务计数器包括同步和异步交易数量计数，以及空事务的数量。
xfs.inode_ops.*	针对操作系统在索引节点缓存中查找具有不同结果的 XFS 索引节点的次数，计数器.这些计数缓存命中、缓存未命中等。
xfs.log.* xfs.log_tail.*	通过 XFS 文件符号链接写入的日志缓冲区数量计数器包括写入到磁盘的块数。日志清空和固定数量的指标。
xfs.xstrat.*	XFS flush deamon 清除出的文件数据的字节数，以及刷新到磁盘上连续和非相邻空间的缓冲区数量。
xfs.attr.*	所有 XFS 文件系统上的属性 get、设置、删除和列出操作的数量。
xfs.quota.*	XFS 文件系统的配额操作指标包括配额重新声明数量的计数器、配额缓存未命中、缓存命中和配额数据重新声明。
xfs.buffer.*	有关 XFS 缓冲区对象的指标范围。计数器包括请求的缓冲区调用数量、成功缓冲区锁定、等待的缓冲区锁定、failure_locks、failure_retries 和 buffer hits（查找页面时）。
xfs.btree.*	有关 XFS btree 操作的指标。

指标组	提供的指标
xfs.control.reset	用于重置 XFS 统计的指标计数器的配置指标。控制指标通过 pmstore 工具切换。

表 A.4. 每个设备的用于 XFS 的 PCP 指标组

指标组	提供的指标
xfs.perdev.*	常规 XFS 指标，包括读取和写入操作计数、读取和写入字节计数。与计数器一起清空、群集化和群集失败次数的内节点数。
xfs.perdev.allocs.* xfs.perdev.alloc_btree.*	有关在文件系统中分配对象的指标范围，其中包括扩展数目和块创建/释放。分配树查找，并与从 btree 中的扩展记录创建和删除进行比较。
xfs.perdev.block_map.* xfs.perdev.bmap_tree.*	指标包括块映射读/写和块删除的数量，用于插入、删除和查找的扩展列表操作。另外，用于从 blockmap 中比较、查找、插入和删除操作的操作计数器。
xfs.perdev.dir_ops.*	XFS 文件系统的目录操作计数器，用于创建、条目删除、"getdent"操作计数.
xfs.perdev.transactions.*	元数据事务计数器包括同步和异步交易数量计数，以及空事务的数量。
xfs.perdev.inode_ops.*	针对操作系统在索引节点缓存中查找具有不同结果的 XFS 索引节点的次数，计数器.这些计数缓存命中、缓存未命中等。
xfs.perdev.log.* xfs.perdev.log_tail.*	通过 XFS fileytems 写入日志缓冲区数量的计数器包括写入到磁盘的块数。日志清空和固定数量的指标。
xfs.perdev.xstrat.*	XFS flush deamon 清除出的文件数据的字节数，以及刷新到磁盘上连续和非相邻空间的缓冲区数量。
xfs.perdev.attr.*	所有 XFS 文件系统上的属性 get、设置、删除和列出操作的数量。
xfs.perdev.quota.*	XFS 文件系统的配额操作指标包括配额重新声明数量的计数器、配额缓存未命中、缓存命中和配额数据重新声明。
xfs.perdev.buffer.*	有关 XFS 缓冲区对象的指标范围。计数器包括请求的缓冲区调用数量、成功缓冲区锁定、等待的缓冲区锁定、failure_locks、failure_retries 和 buffer hits（查找页面时）。
xfs.perdev.btree.*	有关 XFS btree 操作的指标。

A.8. VMSTAT

vmstat 输出报告您系统的进程、内存、分页、块输入/输出、中断和 CPU 活动。它提供自计算机上一次启动或上次报告之后平均这些事件的即时报告。

-a

显示活跃和不活跃的内存.

-f

显示自启动以来的 fork 数量.这包括 fork、vfork 和 clone 系统调用，相当于创建的任务总数。每个进程都由一个或多个任务表示，具体取决于线程使用情况。此显示不会重复。

-m

显示 slab 信息。

-n

指定标头将出现一次，而不是定期出现。

-s

显示包含各种事件计数器和内存统计信息的表。此显示不会重复。

delay

报告之间的延迟（以秒为单位）。如果没有指定延迟，则只打印一个报告，自计算机上一次启动以来的平均值为平均值。

数量

报告系统的次数。如果没有指定计数并定义了延迟，vmstat 会无限期报告。

-d

显示磁盘统计信息.

-p

将分区名称取为值，并报告该分区的详细统计信息。

-S

定义报告的单元输出。有效值为 k (1000 字节), K (1024 bytes), m (1,000,000 字节), 或 M

(1,048,576 字节)。

-D

报告有关磁盘活动的汇总统计信息。

有关每个输出模式提供的输出的详情，请查看 **man page**：

```
$ man vmstat
```

A.9. X86_ENERGY_PERF_POLICY

x86_energy_perf_policy 工具允许管理员定义性能和能源效率相对的重要性。它由 **kernel-tools** 软件包提供。

要查看当前的策略，请运行以下命令：

```
# x86_energy_perf_policy -r
```

要设置新策略，请运行以下命令：

```
# x86_energy_perf_policy profile_name
```

将 **profile_name** 替换为以下配置集之一：

performance

处理器不会为了节省能源而牺牲性能。这是默认值。

Normal

处理器可容忍细微的性能损失，从而可能显著节省能源。这是为大多数服务器和桌面节省合理的费用。

powerave

处理器接受性能可能显著降低，以便最大程度提高能源效率。

有关如何使用 x86_energy_perf_policy 的详情，请查看手册页：

```
$ man x86_energy_perf_policy
```

A.10. TURBOSTAT

turbostat 工具提供有关系统处于不同状态的时间长度的详细信息。turbostat 由 kernel-tools 软件包提供。

默认情况下，turbostat 会在以下标题下显示整个系统的计数器结果摘要，每 5 秒显示计数器结果：

pkg

处理器软件包号。

core

处理器内核号。

CPU

Linux CPU（逻辑处理器）编号。

%c0

CPU 停用指令的间隔百分比。

GHz

当这个数字大于 TSC 中的值时，CPU 处于 turbo 模式

TSC

整个间隔的平均时钟速度。

%c1、%c3 和 %c6

处理器分别处于 c1、c3 或 c6 状态的间隔百分比。

%pc3 或 %pc6

处理器分别处于 pc3 或 pc6 状态的间隔百分比。

使用 -i 选项指定计数器结果之间的不同周期，例如，运行 turbostat -i 10 以每 10 秒打印结果。

注意

即将到来的 Intel 处理器可能会添加其他 c 状态.从 Red Hat Enterprise Linux 7.0 开始，turbostat 为 c7、c8、c9 和 c10 状态提供支持。

A.11. NUMASTAT

numastat 工具由 numactl 软件包提供，并显示每个 NUMA 节点的进程和操作系统的内存统计信息（如分配点击和未命中）。numastat 命令的默认跟踪类别概述如下：

numa_hit

成功分配给此节点的页面数。

numa_miss

由于预期节点上的内存不足，在此节点上分配的页面数量。每个 numa_miss 事件在另一个节点上都有对应的 numa_foreign 事件。

numa_foreign

最初用于此节点的页面数量改为分配给另一节点。每个 numa_foreign 事件在另一个节点上都有对应的 numa_miss 事件。

interleave_hit

成功分配给此节点的 interleave 策略页面数量。

local_node

此节点上的进程在此节点上成功分配的页面数。

other_node

此节点上由另一节点上的进程分配的页面数量。

提供以下任何选项会将显示的单位更改为 MB 内存（舍入为两个十进制位置），并更改其他特定的 numastat 行为，如下所述。

-c

水平精简显示的信息表。这在有大量 NUMA 节点的系统中非常有用，但列宽和列列间间隔有些不可预测。使用此选项时，内存量将四舍五入到最接近的兆字节。

-m

根据每个节点显示系统范围的内存用量信息，类似于 /proc/meminfo 中提供的信息。

-n

显示与原始 numastat 命令相同的信息 (numa_hit,numa_miss,numa_foreign,interleave_hit,local _node及其他_node)，使用更新的格式，将 MB 用作测量单位。

-p pattern

显示指定模式的每个节点内存信息。如果模式的值由数字组成，则 numastat 会假定它是一个数字进程标识符。否则，numastat 会搜索指定模式的进程命令行。

在 -p 选项的值后面输入的命令行参数假定为过滤的其他模式。其他模式扩展而不是缩小过滤器。

-s

按降序排列显示的数据，以便首先列出最大内存消费者（根据总列）。

（可选）您可以指定一个节点，并将根据节点列对表进行排序。在使用这个选项时，节点值必须立即遵循 -s 选项，如下所示：

```
numastat -s2
```

不要在 选项及其值之间包括空格。

-v

显示更详细的信息.即，多个进程的进程信息将显示每个进程的详细信息。

-V

显示 numastat 版本信息。

-z

从显示的信息中省略表格行和列，其中仅为零值。请注意，显示输出中不会省略一些舍入为零的近零值。

A.12. NUMACTL

numactl 允许管理员使用指定的调度或内存放置策略运行进程。numactl 也可以为共享内存段或文件设置持久策略，并设置进程的处理器关联和内存关联性。

numactl 提供了很多有用的选项。本附录概述了其中一些选项，并提供了有关使用选项的建议，但并不详尽。

--hardware

显示系统上可用节点的清单，包括节点之间的相对距离。

--membind

确保仅从特定节点分配内存。如果指定位置中的可用内存不足，分配会失败。

--cpunodebind

确保指定的命令及其子进程仅在指定节点上执行。

--phycpubind

确保指定的命令及其子进程仅在指定的处理器上执行。

--localalloc

指定始终从本地节点分配内存。

--preferred

指定从中分配内存的首选节点。如果无法从此指定节点分配内存，则另一个节点将用作回退。

有关这些参数和其他参数的详情，请查看 **man page**：

```
$ man numactl
```

A.13. NUMAD

numad 是一个自动 **NUMA** 关联性管理守护进程。它监控系统中 **NUMA** 拓扑和资源使用情况，以便动态改进 **NUMA** 资源的分配和管理。

请注意，当启用 **numad** 时，其行为会覆盖自动 **NUMA** 平衡的默认行为。

A.13.1. 从命令行使用 numad

要将 **numad** 用作可执行文件，请运行：

```
# numad
```

在 **numad** 运行时，其活动记录在 **/var/log/numad.log** 中。它将运行直到使用以下命令停止：

```
# numad -i 0
```

停止 **numad** 不会删除它对提高 **NUMA** 关联性所做的更改。如果系统使用显著的变化，再次运行 **numad** 会调整关联性，以便在新条件下提高性能。

要将 numad 管理限制到特定进程，请使用以下选项启动它：

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

-p pid

此选项将指定的 添加到明确包含列表中。在满足 numad 进程信号阈值前，指定的进程不会被管理。

-S 0

这会将进程扫描类型设置为 0， 这会将 numad 管理限制为显式包含的进程。

有关可用 numad 选项的详情，请参考 numad 手册页：

```
$ man numad
```

A.13.2. 使用 numad 作为服务

虽然 numad 作为服务运行，它会尝试根据当前的系统工作负载动态调整系统。其活动记录在 /var/log/numad.log 中。

要启动该服务，请运行：

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

要使服务在重启后保持有效，请运行：

```
# chkconfig numad on
```

有关可用 numad 选项的详情，请参考 numad 手册页：

```
$ man numad
```

A.13.3. 预放置公告

numad 提供了一个预放置建议服务，可由各种作业管理系统查询，为进程提供 CPU 和内存资源的初始绑定。无论 numad 作为可执行文件还是服务运行，这个预替换建议都可用。

A.13.4. 使用 KSM 的 numad

如果在 NUMA 系统上使用 KSM，请将 /sys/kernel/mm/ksm/merge_nodes 参数的值更改为 0，以避免跨 NUMA 节点合并页面。否则，KSM 会增加远程内存访问，因为它跨节点合并页面。此外，在大量跨节点合并后，内核内存核算统计最终可能会相互冲突。因此，在 KSM 守护进程合并多个内存页面后，numad 可能会成为可用内存的正确数量和位置。仅当您在系统上过量使用内存时，KSM 才有意义。如果您的系统有足够的可用内存，可以通过关闭和禁用 KSM 守护进程来实现更高的性能。

A.14. OPROFILE

OProfile 是一个低开销，系统范围的性能监控工具，由 oprofile 软件包提供。它使用处理器上的性能监控硬件来检索有关系统上内核和可执行文件的信息，如引用内存、二级缓存请求数以及收到的硬件中断数量。OProfile 还可以对 Java 虚拟机(JVM)中运行的应用程序进行性能分析。

OProfile 提供以下工具：请注意，传统的 opcontrol 工具和新的 operf 工具是互斥的。

ophelp

显示系统处理器的可用事件，以及各个处理器的简短描述。

opimport

将示例数据库文件从外部二进制格式转换为系统的原生格式。仅当从不同架构分析示例数据库时，才使用这个选项。

opannotate

如果应用编译了调试符号，则为可执行文件创建带注解的源。

opcontrol

配置在分析运行中收集的数据。

operf

旨在替换 opcontrol。operf 工具使用 Linux 性能事件子系统，允许您更精确地将性能分析作为单个进程或系统范围的目标，并允许 OProfile 与系统上的性能监控硬件更好地共存其他工具。与

opcontrol 不同，不需要初始设置，并且可以在没有 root 特权的情况下使用，除非使用了 --system-wide 选项。

opreport

检索配置文件数据.

oprofiled

作为守护进程运行，定期将样本数据写入磁盘。

旧模式(opcontrol、oprofiled 和后处理工具)仍然可用，但不再是推荐的性能分析方法。

有关这些命令的详情，请查看 OProfile man page：

```
$ man oprofile
```

A.15. TASKSET

taskset 工具由 util-linux 软件包提供。它允许管理员检索和设置正在运行的进程的处理器关联，或者启动具有指定处理器相关性的进程。

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优势，红帽建议使用 numactl 而不是 taskset。

要设置正在运行的进程的 CPU 关联性，请运行以下命令：

```
# taskset -pc processors pid
```

使用以逗号分隔的处理器或处理器范围（例如 1,3,5-7 ）替换处理器。使用您要重新配置的进程的进程标识符替换。

要启动具有指定关联性的进程，请运行以下命令：

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

使用以逗号分隔的处理器或处理器范围列表替换处理器。使用 命令、选项和要运行的应用的参数替换应用。

有关 taskset 的更多信息，请参阅 man page：

```
$ man taskset
```

A.16. SYSTEMTAP

SystemTap 广泛记录在其自己的 指南中：SystemTap Beginners 指南的 Red Hat Enterprise Linux 7 版本和 SystemTap Tapset 参考。

附录 B. 修订历史记录

修订 10.13-59 异步更新.	Mon May 21 2018	Marek Suchánek
修订 10.14-00 为 7.5 GA 发布准备文档.	Fri Apr 6 2018	Marek Suchánek
修订 10.13-58 新部分：pqos.	Fri Mar 23 2018	Marek Suchánek
修订 10.13-57 异步更新.	Wed Feb 28 2018	Marek Suchánek
修订 10.13-50 发布 7.4 GA 的文件版本.	Thu Jul 27 2017	Milan Navrátil
修订 10.13-44 异步更新.	Tue Dec 13 2016	Milan Navrátil
修订 10.08-38 7.2 GA 版本.	Wed Nov 11 2015	Jana Heves
修订 0.3-23 为 RHEL 7.1 GA 而构建.	Tue Feb 17 2015	Laura Bailey
修订 0.3-3 为 RHEL 7.0 GA 重建.	Mon Apr 07 2014	Laura Bailey