Red Hat Enterprise Linux 7

电源管理指南

在 RHEL 7 中管理和优化功耗

Last Updated: 2023-10-10

Red Hat Enterprise Linux 7 电源管理指南

在 RHEL 7 中管理和优化功耗

Marie Doleželová
Red Hat Customer Content Services
mdolezel@redhat.com

Jana Heves
Red Hat Customer Content Services

Jacquelynn East
Red Hat Customer Content Services

Don Domingo
Red Hat Customer Content Services

Rüdiger Landmann
Red Hat Customer Content Services

Jack Reed
Red Hat Customer Content Services

Red Hat, Inc.

法律通告

摘要

本文档解释了如何有效地管理 Red Hat Enterprise Linux 7 系统上的功耗。以下小节讨论了降低功耗（针对服务器和笔记本电脑）的不同技术，以及每个技术如何影响系统的整体性能。

目录

第 1 章 概述

电源管理是我们对 Red Hat Enterprise Linux 7 的改进的重点之一。限制计算机系统所使用的电源是 *绿色 IT*（环境友好的计算）中最重要的方面之一，其中的一些考虑因素还包括使用回收材料、硬件生产环境影响以及系统设计和部署的环境感知。在本文档中，我们提供有关运行 Red Hat Enterprise Linux 7 的系统的电源管理指导和信息。

1.1. 电源管理的重要性

电源管理核心在于了解如何有效地优化每个系统组件的能源消耗。这对您的系统执行的不同任务进行了研究，并配置每个组件以确保其性能只用于工作。

电源管理的主要动机是：

- 降低总体功耗以节约成本

正确使用电源管理结果：

- 服务器和计算中心的发热的缩减
- 降低辅助成本，包括冷却、空间、电缆、生成器和 *不间断电源* (UPS)
- 延长笔记本电脑的电池寿命
- 降低二氧化碳的输出
- 满足有关 Green IT 的政府法规或法律要求，例如 Energy Star
- 使新系统满足公司的要求

作为规则，降低特定组件（或系统的整体）的功耗会降低 heat 和 naturally 性能。因此，您应该全面研究并测试由您进行的任何配置（特别是用于关键任务系统）所负担的性能降低。

通过研究系统执行的不同任务，并配置每个组件以确保其性能足以用于工作，您可以节省能源，为笔记本电脑生成较少的 heat 和优化电池生命周期。与功耗相关的许多系统分析和调优的原则与性能调优类似。在某种程度上，电源管理和性能调优与系统配置相反，因为系统通常针对性能或电源进行了优化。本手册介绍了红帽提供的工具，以及我们开发的技术来帮助您完成此过程。

Red Hat Enterprise Linux 7 已附带许多新的电源管理功能，这些功能默认启用。它们都是有选择的，不会影响典型服务器或桌面用例的性能。但是，对于非常具体的用例，在绝对需要最大吞吐量、最低延迟或最高 CPU 性能时，可能需要检查这些默认值。

要确定您是否要使用本文档中描述的技术优化机器，请自己询问几个问题：

问： **我必须优化吗？**

答： 电源优化的重要性取决于您的公司是否具有需要遵循的准则，或者是否有需要满足的法规。

问： **要优化，我需要多少？**

答： 我们提供的多种技术并不需要您完成整个审计和分析机器，而是提供一组通常改进功耗的常规优化。这些课程通常不如手动审核和优化的系统，但提供了很好的折衷。

问： **是否将系统性能降低到不可接受的级别？**

答： 本文档中描述的大多数技术都会显著影响您系统性能。如果您选择在 Red Hat Enterprise Linux 7 中的默认设置之外实现电源管理，您应该在电源优化后监控系统性能，并确定性能丢失是否可以接受。

问： **优化系统所实现的时间和资源是否超过其收益？**

答： 在整个流程后手动优化单个系统通常并不值得注意，因为这样做所花费的时间和成本比您在一台机器生命周期中所获得的典型优势要高得多。例如，如果您将 10000 个桌面系统部署到您的办公室中，则使用相同的配置和设置，然后创建一个优化的设置，并将这些设置应用到所有 10000 个计算机很好。

以下小节解释了在能源消耗方面，最佳的硬件性能将给您的系统带来怎样的好处。

1.2. 电源管理基础

有效电源管理基于以下原则构建：

闲置 CPU 应该在需要时唤醒

从 Red Hat Enterprise Linux 6 开始，内核会无空运行，这意味着以前的定期计时器中断已被按需中断替代。*因此，空闲的 CPU 可以在新任务排队进行处理前保持闲置状态，并且已处于较低电源状态的 CPU 可以保持这个状态更长时间。但是，如果您的系统中存在会创建不必要的计时器事件的应用程序时，此功能的好处可能会减少。轮询事件，如检查卷更改或鼠标移动是此类事件的示例。*

Red Hat Enterprise Linux 7 包括您可以根据其 CPU 使用率识别和审核应用程序的工具。详情请查看 *第 2 章 电源管理审计和分析**。*

应该完全禁用未使用的硬件和设备

对于有移动部分（例如硬盘）的设备来说，这尤其如此。除此之外，一些应用程序可能会留下未使用但已启用的设备"打开"；当发生这种情况时，内核会假定设备处于使用状态，这样可防止设备进入节能状态。

较少的活动应转代表低的电源消耗

然而，在很多情况下，这取决于现代硬件和正确的 BIOS 配置。较旧的系统组件通常不支持我们现在可在 Red Hat Enterprise Linux 7 中支持的一些新功能。确保您的系统使用了最新的官方固件，且在 BIOS 的电源管理或设备配置部分中启用了电源管理功能。要查找的一些功能包括：

- *SpeedStep*
- *PowerNow!*
- *cool'n'Quiet*
- *ACPI (C 状态)*
- *Smart*

如果您的硬件支持这些功能，且在 BIOS 中启用了这些功能，Red Hat Enterprise Linux 7 将默认使用它们。

不同的 CPU 状态形式及其影响

现代 CPU 与 高级配置和电源接口 (ACPI) 一起提供不同的电源状态。三个不同的状态是：

- *Sleep (C-states)*
- *Frequency and voltage (P-states)*

  p-state 描述了处理器及其环境操作点的频率，它们随着 P-state 的增加而扩展。

- *Heat 输出(T-states 或 "thermal state")*

在最低睡眠状态中运行的 CPU 可能会消耗最少的 watts，但在需要时从该状态唤醒所需的时间也要长得多。在非常罕见的情形中，这可能会导致 CPU 在每次将要进入睡眠状态时被立即唤醒。这种情况会导致 CPU 一直处于忙碌状态，并在已使用另一个状态时丧失一些潜在的节能好处。

关闭的机器使用最少电能

很明显，因为这听起来可能听到，实际省电功能的最佳方法之一就是关闭系统。例如，您的公司可以开发一个企业文化，专注于"绿色 IT"感知，在午休休息或离开时切换机器。您还可以将多个物理服务器整合到一个较大的服务器中，并使用我们随 Red Hat Enterprise Linux 7 提供的虚拟化技术进行虚拟化。

第 2 章 电源管理审计和分析

2.1. 审计和分析概述

通常，单个系统的详细手动审计、分析和调优是例外，因为通常要做的时间和成本远远超过了从这些最后一部分系统调节中获得的好处。但是，对于相似的、可以在所有系统中重复使用相同设置的大量系统，一次执行这些任务可能会非常有用。例如，考虑部署数千台桌面系统，或部署相同机器的 HPC 集群。进行审核和分析的另一种原因是，您可以创建一个基础，用于在以后比较环境变化造成的影响。对于需要定期更新硬件、BIOS 或软件的环境，此分析结果会非常有帮助，可以帮助避免出现与功耗相关的任何意外情况。通常，全面的审计和分析让您可以更好地了解特定系统中发生的情况。

对于电源消耗，审计和分析系统相对来说比较困难，即使对于大多数现代系统也是如此。大多数系统并不会提供通过软件测量电源使用情况的方法。然而，存在一些情况：Hewlett Packard 服务器系统的 ILO 管理控制台有一个电源管理模块，您可以通过 Web 访问。IBM 在其 BladeCenter 电源管理模块中提供类似的解决方案。在一些 Dell 系统中，IT Assistant 也提供了电源监控功能。其他供应商可能会为其服务器平台提供类似的功能，但并没有一个统一的解决方案用于所有厂商。

通常需要直接测量功耗，以尽可能最大化。不幸的是，可以使用其他方法来测量更改是否生效或系统的行为方式。本章论述了必要的工具。

2.2. POWERTOP

在 Red Hat Enterprise Linux 7 中引入无空内核可让 CPU 更频繁地进入空闲状态，从而减少功耗并改进电源管理。**PowerTOP** 工具识别频繁唤醒 CPU 的内核和用户空间应用程序的特定组件。powertop 的开发用于执行审计，这会导致在此版本中调整许多应用程序，从而减少了 10 倍的 CPU 唤醒。

Red Hat Enterprise Linux 7 附带 **PowerTOP** 版本。这个版本是 1.x 代码库的完整重写。它具有基于 tab 的更清晰的用户界面，并广泛使用内核"完美"基础架构来提供更多准确的数据。系统设备的电源行为会被跟踪并实际显示，因此可以快速发现问题。在实验性上，2.x 代码库包含一个电源估算引擎，可以指示消耗电源各个设备和进程的数量。请参阅 图 2.1 “操作中的 powertop”。

要安装 **PowerTOP** 运行，以 **root** 用户身份运行以下命令：

```
~]# yum install powertop
```

要运行 **PowerTOP**，请以 **root** 用户身份使用，使用以下命令：

```
~]# powertop
```

powertop 可以提供系统总功耗的估算，并为每个进程、设备、内核工作、计时器和中断处理程序显示单独的功耗。在此任务期间，笔记本电脑应在电池电源上运行。要校准电源估算引擎，以 **root** 用户身份运行以下命令：

```
~]# powertop --calibrate
```

校准需要时间。进程执行各种测试，并将通过亮度级别和切换设备进行循环。让进程完成，且不会在校准期间与机器进行交互。完成校准过程后，**PowerTOP** 会正常启动。让它运行大约一小时以收集数据。收集足够数据后，将在第一列中显示节能数据。

如果您在笔记本电脑上执行命令，它仍然应在电池电源上运行，以便显示所有可用的数据。

在运行期间，**PowerTOP** 会从系统中收集统计信息。在 **Overview** 选项卡中，您可以查看将 wake-ups 发送到 CPU 最频繁或消耗最多电源的组件列表（请参阅 图 2.1 “操作中的 powertop”）。相邻列显示电源估算、如何使用资源、每秒唤醒、组件的分类，如进程、设备或计时器以及组件的描述。每秒唤醒数代表如

何高效地执行内核的服务或设备和驱动程序。较少的 wakeups 表示消耗较少的电源。组件按照可进一步优化的电源使用量排序。

调整驱动程序组件通常需要内核更改，这超出了本文档的范围。但是，发送 wakeups 的 userland 进程更容易管理。首先，确定该服务或应用程序是否需要在此系统上运行。如果没有，只需取消激活它。要永久关闭旧的 System V 服务，请运行：

```
~]# systemctl disable servicename.service
```

有关进程的详情，请以 **root** 身份运行，使用以下命令：

```
~]# ps -awux | grep processname
~]# strace -p processid
```

如果 trace 看起来像是重复自己，则可能是一个忙碌的循环。修复此类错误通常需要该组件中的代码更改。

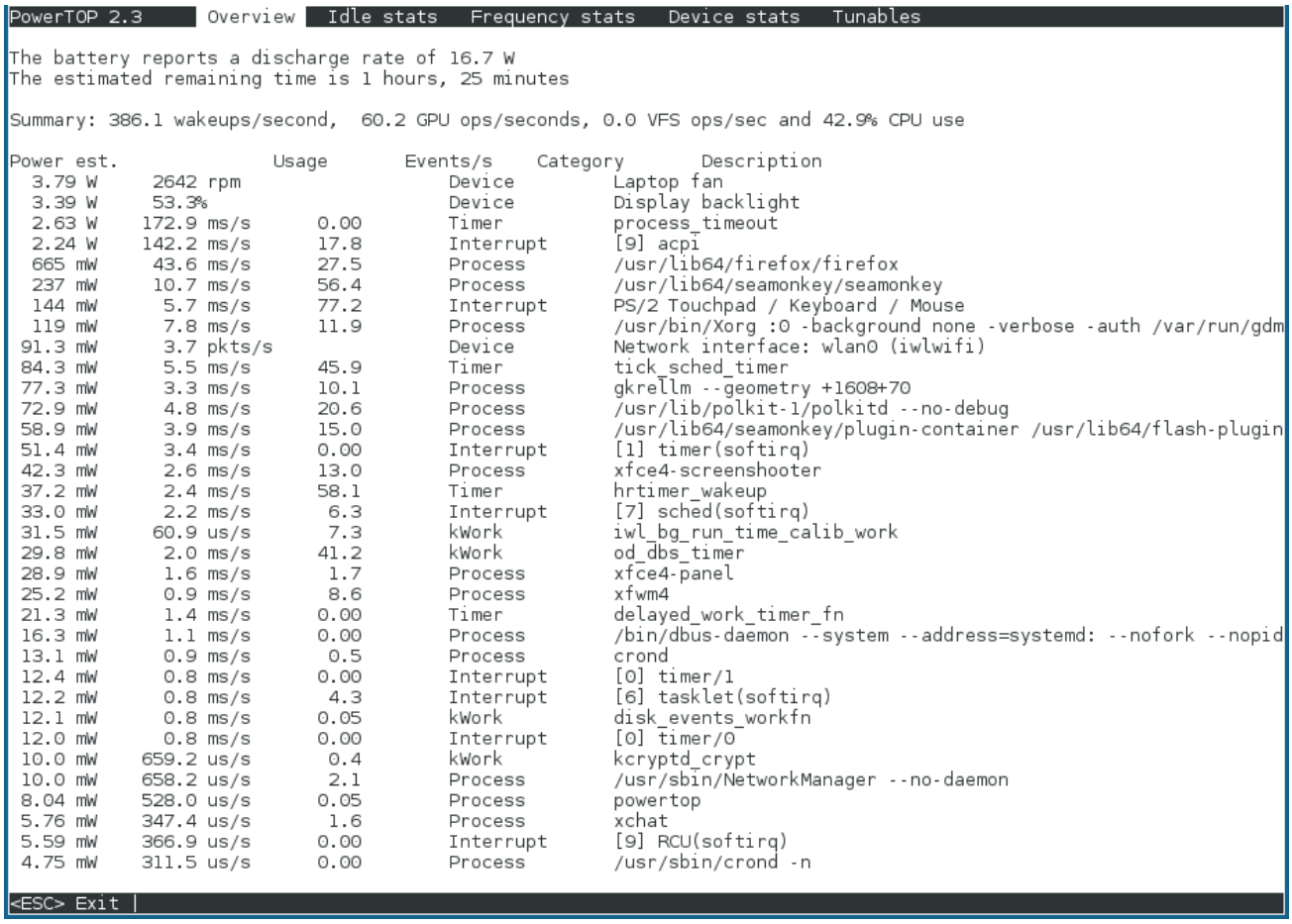
如 图 2.1 “操作中的 powertop” 中所示，如果适用，则会显示总功耗和剩余电池生命周期。以下简短概述包括每秒唤醒的总唤醒、每秒 GPU 操作数和每秒虚拟文件系统操作。在屏幕的其余部分中，有一个进程、中断、设备和其他资源列表，并根据它们的利用率排序。如果正确校准，则会显示第一列中每个列出的项目的功耗估算。

使用 **Tab** 和 **Shift+Tab** 键通过选项卡进行循环。在 **Idle stats** 选项卡中，显示所有处理器和内核使用 C-states。在 **Frequency stats** 选项卡中，显示所有处理器和内核，使用 P-states 包括 Turbo 模式（如果适用）。CPU 越长，CPU 保持在更高的 C- 或 P-states 中，更好(**C4** 大于 **C3**)。这对 CPU 用量的优化方式是一个很好的指示。在系统闲置时，最好最好在 C- 或 P-state 中达到 90% 或更高。

Device Stats 选项卡提供与 **Overview** 选项卡类似的信息，但只适用于设备。

Tunables 选项卡包含优化系统以降低功耗的建议。使用 **up** 和 **down** 键移动建议，使用 **enter** 键打开和关闭建议。

图 2.1. 操作中的 powertop



[D]

*您还可以使用 **--html** 选项运行 **PowerTOP** 来生成 HTML 报告。将 htmlfile.html 参数替换为输出文件所需名称：*

```
~]# powertop --html=htmlfile.html
```

*默认情况下，**PowerTOP** 使用 20 秒的测量，您可以使用 **--time** 选项更改它：*

```
~]# powertop --html=htmlfile.html --time=seconds
```

*有关 **PowerTOP** 的更多信息，请参阅 PowerTOP 主页。*

***powertop** 也可以与 **turbostat** 实用程序一起使用。这是一个报告工具，显示 Intel 64 处理器上的处理器拓扑、频率、空闲的电源状态统计、温度和功耗的信息。有关 **turbostat** 实用程序的更多信息，请参阅 **turbostat**(8) man page，或阅读 性能调优指南。*

2.3. DISKDEVSTAT 和 NETDEVSTAT

***diskdevstat** 和 **netdevstat** 是 **SystemTap** 工具，可收集有关系统上运行的所有应用的磁盘活动和网络活动的详细信息。这些工具由 **PowerTOP** 增长，它显示每个应用程序每秒的 CPU 唤醒数量（请参阅 第 2.2 节 "PowerTOP"）。通过这些工具收集的统计数据，您可以识别具有许多小 I/O 操作（而非更少的大操作）的功能。仅测量传输速度的其他监控工具有助于识别此类使用情况。*

*以 **root** 身份使用以下命令使用 **SystemTap** 安装这些工具：*

```
~]# yum install tuned-utils-systemtap kernel-debuginfo
```

使用以下命令运行工具：

```
~]# diskdevstat
```

或者命令：

```
~]# netdevstat
```

这两个命令最多可取三个参数，如下所示：

diskdevstat update_interval total_duration display_histogram

netdevstat update_interval total_duration display_histogram

update_interval
显示更新间隔的时间（以秒为单位）。默认：**5**

total_duration
整个运行的时间（以秒为单位）。默认：**86400** (1 天)

display_histogram
标记是否在运行结束时所有收集的数据的直方图。

输出类似于 PowerTOP。以下是较长的 diskdevstat 运行的输出示例：

```
PID  UID  DEV WRITE_CNT WRITE_MIN WRITE_MAX WRITE_AVG READ_CNT READ_MIN
READ_MAX READ_AVG COMMAND
 2789  2903 sda1     854     0.000   120.000    39.836      0     0.000     0.000     0.000 plasma
 5494     0 sda1       0     0.000     0.000     0.000    758     0.000     0.012     0.000 0logwatch
 5520     0 sda1       0     0.000     0.000     0.000    140     0.000     0.009     0.000 perl
 5549     0 sda1       0     0.000     0.000     0.000    140     0.000     0.009     0.000 perl
 5585     0 sda1       0     0.000     0.000     0.000    108     0.001     0.002     0.000 perl
 2573     0 sda1      63     0.033  3600.015   515.226      0     0.000     0.000     0.000 auditd
 5429     0 sda1       0     0.000     0.000     0.000     62     0.009     0.009     0.000 crond
 5379     0 sda1       0     0.000     0.000     0.000     62     0.008     0.008     0.000 crond
 5473     0 sda1       0     0.000     0.000     0.000     62     0.008     0.008     0.000 crond
 5415     0 sda1       0     0.000     0.000     0.000     62     0.008     0.008     0.000 crond
 5433     0 sda1       0     0.000     0.000     0.000     62     0.008     0.008     0.000 crond
 5425     0 sda1       0     0.000     0.000     0.000     62     0.007     0.007     0.000 crond
 5375     0 sda1       0     0.000     0.000     0.000     62     0.008     0.008     0.000 crond
 5477     0 sda1       0     0.000     0.000     0.000     62     0.007     0.007     0.000 crond
 5469     0 sda1       0     0.000     0.000     0.000     62     0.007     0.007     0.000 crond
 5419     0 sda1       0     0.000     0.000     0.000     62     0.008     0.008     0.000 crond
 5481     0 sda1       0     0.000     0.000     0.000     61     0.000     0.001     0.000 crond
 5355     0 sda1       0     0.000     0.000     0.000     37     0.000     0.014     0.001 laptop_mode
 2153     0 sda1      26     0.003  3600.029  1290.730      0     0.000     0.000     0.000 rsyslogd
 5575     0 sda1       0     0.000     0.000     0.000     16     0.000     0.000     0.000 cat
 5581     0 sda1       0     0.000     0.000     0.000     12     0.001     0.002     0.000 perl
 5582     0 sda1       0     0.000     0.000     0.000     12     0.001     0.002     0.000 perl
 5579     0 sda1       0     0.000     0.000     0.000     12     0.000     0.001     0.000 perl
```

```
5580    0 sda1    0    0.000    0.000    0.000    12    0.001    0.001    0.000 perl
5354    0 sda1    0    0.000    0.000    0.000    12    0.000    0.170    0.014 s  h
5584    0 sda1    0    0.000    0.000    0.000    12    0.001    0.002    0.000 perl
5548    0 sda1    0    0.000    0.000    0.000    12    0.001    0.014    0.001 perl
5577    0 sda1    0    0.000    0.000    0.000    12    0.001    0.003    0.000 perl
5519    0 sda1    0    0.000    0.000    0.000    12    0.001    0.005    0.000 perl
5578    0 sda1    0    0.000    0.000    0.000    12    0.001    0.001    0.000 perl
5583    0 sda1    0    0.000    0.000    0.000    12    0.001    0.001    0.000 perl
5547    0 sda1    0    0.000    0.000    0.000    11    0.000    0.002    0.000 perl
5576    0 sda1    0    0.000    0.000    0.000    11    0.001    0.001    0.000 perl
5518    0 sda1    0    0.000    0.000    0.000    11    0.000    0.001    0.000 perl
5354    0 sda1    0    0.000    0.000    0.000    10    0.053    0.053    0.005 lm_lid.sh
```

列是：

PID

应用程序的进程 ID

UID

运行应用程序的用户 ID

DEV

I/O 发生的设备

WRITE_CNT

写入操作的总数

WRITE_MIN

连续两个写入需要的最低时间（以秒为单位）

WRITE_MAX

连续两个写入的最大时间（以秒为单位）

WRITE_AVG

两个连续写入的平均时间（以秒为单位）

READ_CNT

读取操作的总数

READ_MIN

连续两个读取所需的最低时间（以秒为单位）

READ_MAX

连续两个读数的最大时间（以秒为单位）

READ_AVG

连续两个读取的平均时间（以秒为单位）

命令

进程的名称

在这个示例中，三个非常明显的应用程序代表：

```
PID  UID  DEV WRITE_CNT WRITE_MIN WRITE_MAX WRITE_AVG READ_CNT READ_MIN
READ_MAX READ_AVG COMMAND
 2789  2903 sda1     854     0.000   120.000    39.836      0     0.000     0.000     0.000 plasma
2573     0 sda1      63     0.033  3600.015   515.226      0     0.000     0.000     0.000 auditd
 2153     0 sda1      26     0.003  3600.029  1290.730      0     0.000     0.000     0.000 rsyslogd
```

这三个应用具有大于 **0** 的 **WRITE_CNT**，这意味着它们在测量期间执行某种形式的写入。在这些方面，**plasma** 是一大水准的先行者：它执行最多的写入操作，而在写入之间的平均时间最低。因此，如果您关注节能应用程序，**Plasma** 将是调查的最佳候选者。

通过追踪给定进程 ID 的所有系统调用，使用 **strace** 和 **ltrace** 命令检查应用程序。在目前示例中，您可以运行：

```
~]# strace -p 2789
```

在本例中，**strace** 的输出包含一次重复模式，每个 45 秒打开用户的 KDE 图标缓存文件，以便再次写入，然后立即关闭该文件。这会导致在文件元数据（特别是修改时间）时对硬盘进行必要的物理写入。最终的修复是防止没有发生对图标的更新时进行不必要的调用。

2.4. 电池生命工具工具包

Red Hat Enterprise Linux 7 引入了 **Battery Life Tool Kit** (BLTK)，它是一个模拟和分析电池生命周期和性能的测试套件。BLTK 通过执行一系列任务来模拟特定用户组并报告结果来实现此目的。虽然专门为测试笔记本性能而开发，但 BLTK 也可以在使用 **-a** 启动时报告桌面计算机的性能。

BLTK 允许您生成可重复生成的工作负载，这些工作负载与计算机的实际使用相当。例如，办**公室** 工作负载会编写一个文本，更正其中的内容，并对电子表格执行同样的操作。通过运行 BLTK 和 **PowerTOP** 或任何其他审计或分析工具，您可以测试您执行的优化时是否活跃使用计算机，而不是仅闲置。因为您可以针对不同的设置多次运行相同的工作负载，所以您可以比较不同的设置的结果。

使用以下命令安装 BLTK ：

```
~]# yum install bltk
```

使用命令运行 BLTK ：

```
~]$ bltk workload options
```

例如，要为 120 秒运行 闲**置** 工作负载：

```
~]$ bltk -I -T 120
```

默认可用的工作负载有：

-I, --idle

系统闲置，用作与其他工作负载比较的基准

-R, --reader

模拟读取文档（默认情况下，使用 Firefox）

-P, --player

模拟从 CD 或者 DVD 驱动器中监视多媒体文件（默认情况下，使用 mplayer）

-O, --office

使用 memberOf .org 套件模拟编辑文档

其他选项允许您指定：

-a, --ac-ignore

忽略 AC 电源是否可用（需要桌面使用）

-T number_of_seconds, --time number_of_seconds

运行测试的时间（以秒为单位），在 闲置 工作负载中使用这个选项

-f filename, --file filename

指定要被特定工作负载使用的文件，例如，要播放工作负载的文件，而不是访问 CD 或者 DVD 驱动器

-W application, --prog application

指定供特定工作负载使用的应用程序，例如，Firefox 以外的浏览器用于 reader 工作负载

BLTK 支持大量专门的选项。详情请查看 bltk 手册页。

BLTK 将生成的结果保存在 /etc/bltk.conf 配置文件中指定的目录中 - 默认情况下，~/.bltk/workload.results.number.例如，~/.bltk/reader.results.002/ 目录包含 读卡器 工作负载（第一个测试没有编号）的结果。结果分散到多个文本文件中。要将结果压缩为易于阅读的格式，请运行：

```
~]$ bltk_report path_to_results_directory
```

现在，结果会显示在结果目录中的名为 Report 的文本文件中。要在终端模拟器中查看结果，请使用 -o 选项：

```
~]$ bltk_report -o path_to_results_directory
```

2.5. TUNED

TuneD 是一个基于配置集的系统调优工具，它使用 udev 设备管理器监控连接的设备，并支持系统设置的静态和动态调优。动态调优是实验性功能，在 Red Hat Enterprise Linux 7 中被默认关闭。

TuneD 提供预定义的配置集来处理常见用例，如高吞吐量、低延迟或节能。您可以修改每个配置集的 Tuned 规则，并自定义如何调整特定设备。有关如何从 PowerTOP 建议创建自定义 Tuned 配置集的步骤，请参考 “使用 powertop2tuned”一节。

根据使用的产品，配置集会被自动设置为默认设置。您可以使用 tuned-adm recommend 命令来确定红帽推荐的配置集作为最适合特定产品的配置集。如果没有可用的建议，则会设置 balanced 配置集。

balanced 配置集适用于大多数工作负载，平衡能源消耗、性能和延迟。使用 balanced 配置集时，通过最大可用计算能力快速完成任务通常需要较少的能源在较长时间内执行相同的任务，而计算能力更少。

如果笔记本电脑处于空闲状态，或者只执行计算的不需要操作，使用 powersave 配置集可能会延长生命周期。对于此类操作，返回较低的能源消耗延迟通常可以接受，或者操作不需要快速完成，例如使用 IRC、查看简单的网页或播放音频和视频文件。

有关 tuned-adm 提供的 Tuned 和节能配置集的详细信息，请参阅 Red Hat Enterprise Linux 7 性能调节指南 中的 Tuned 章节。

使用 powertop2tuned

powertop2tuned 工具允许您从 PowerTOP 建议创建自定义 Tuned 配置集。有关 PowerTOP 的信息，请参考 第 2.2 节 “PowerTOP”。

要安装 powertop2tuned 工具，请使用：

```
~]# yum install tuned-utils
```

要创建自定义配置集，请使用：

```
~]# powertop2tuned new_profile_name
```

默认情况下，powertop2tuned 在 /etc/tuned/ 目录中创建配置集，并在当前选择的 Tuned 配置集中构建自定义配置集。为安全起见，所有 PowerTOP 调优最初在新配置集中禁用。要启用调整，请在 /etc/tuned/profile_name/tuned.conf 文件中取消注释。

您可以使用 --enable 或 -e 选项生成新的配置集，启用 PowerTOP 建议的大部分调整。某些潜在的调整（如 USB 自动暂停）在默认情况下被禁用，需要手动取消注释。

默认情况下，新配置集不会被激活。要激活它，请使用：

```
~]# tuned-adm profile new_profile_name
```

要获得 powertop2tuned 支持的选项列表，请使用：

```
~]$ powertop2tuned --help
```

2.6. UPOWER

在 Red Hat Enterprise Linux 6 DeviceKit-power 中假定是 HAL 一部分的电源管理功能，以及之前 Red Hat Enterprise Linux 版本中 GNOME Power Manager 一部分的一些功能（也称为 第 2.7 节 “GNOME Power Manager”）。Red Hat Enterprise Linux 7、DeviceKit-power 被重命名为 UPower。upower 提供守护进程、API 和一组命令行工具。系统上的每个电源源都表示为设备，无论是物理设备。例如，笔记本电脑电池和 AC 电源源都表示为设备。

您可以使用 upower 命令和以下选项访问命令行工具：

--enumerate, -e

显示系统中每个电源设备的对象路径，例如：

```
/org/freedesktop/UPower/devices/line_power_AC
/org/freedesktop/UPower/devices/battery_BAT0
```

--dump, -d

显示系统上所有电源设备的参数。

--wakeups, -w

显示系统上的 CPU 唤醒。

--monitor, -m

监控系统是否有电源设备的更改，例如，连接或断开 AC 电源源的连接或断开电池。按 Ctrl+C 停止监控系统。

--monitor-detail

监控系统是否有电源设备的更改，例如，连接或断开 AC 电源源的连接或断开电池。--monitor-detail 选项显示比 --monitor 选项更详细。按 Ctrl+C 停止监控系统。

--show-info object_path, -i object_path

显示可用于特定对象路径的所有信息。例如，要获取系统上由对象路径 /org/freedesktop/UPower/devices/battery_BAT0 表示的信息，请运行：

```
~]$ upower -i /org/freedesktop/UPower/devices/battery_BAT0
```

2.7. GNOME POWER MANAGER

GNOME Power Manager 是作为 GNOME 桌面环境的一部分安装的守护进程。在 Red Hat Enterprise Linux 6 中，GNOME Power Manager 提供的 GNOME Power Manager 的大部分电源管理功能已经成为 Red Hat Enterprise Linux 6 中的 DeviceKit-power 工具的一部分（请参阅 第 2.6 节 “upower”）。但是，GNOME Power Manager 仍然是该功能的前端。通过系统跟踪中的小程序，GNOME Power Manager 通知您系统的电源状态的变化；例如，从电池更改为 AC 电源。它还会报告电池状态，并在电池电源较低时提醒您。

2.8. 其他用于审计的工具

Red Hat Enterprise Linux7 提供了一些更多工具，用于执行系统审计和分析。如果您想要验证已发现的内容或需要更深入地了解某些部分，则大部分信息可以用作补充信息的来源。其中许多工具也用于性能调优。它们包括：

vmstat

vmstat 为您提供进程、内存、分页、块 I/O、陷阱和 CPU 活动的详细信息。使用它来仔细了解系统的整体操作以及它处于忙碌的位置。

iostat

iostat 与 vmstat 类似，但仅适用于块设备上的 I/O。它还提供了更详细的输出和统计数据。

blktrace

blktrace 是一个非常详细的块 I/O 跟踪程序。它将信息分解为与应用程序关联的单个块。它与 diskdevstat 相结合非常有用。

第 3 章 CORE INFRASTRUCTURE 和 MECHANICS

重要

要使用本章介绍的 cpupower 命令，请确定您安装了 kernel-tools 软件包。

3.1. CPU 空闲状态

具有 x86 架构的 CPU 支持各种状态，在 CPU 的哪些部分中被取消激活或以较低性能设置运行。这些状态称为 C-states，允许系统通过部分不使用的 CPU 来省电。C-states 是从 C0 开始增长的值，数值越大代表 CPU 功能越低，节省效果更好。给定数量的 C-State 在处理器之间广泛相似，但特定功能集的具体细节可能因处理器系列而异。c-States 0-3 定义，如下所示：

C0

操作或运行状态。在这个状态中，CPU 正常运行，根本不会被闲置。

C1, halt

处理器没有执行任何指令的状态，但通常不处于较低电源状态。CPU 可能会在实际并没有延迟的情况下继续处理。提供 C-State 的所有处理器都需要支持这个状态。Pentium 4 处理器支持名为 C1E 的增强 C1 状态，它实际上是降低功耗的状态。

C2, stop-Clock

这个处理器时钟被冻结的状态，但它会保留其寄存器和缓存的完整状态，因此在时钟再次启动它后，可以立即开始处理。这是一个可选状态。

C3, sleep

处理器真正进入睡眠状态，不需要保持其缓存最新状态。由于此状态的原因，从 C2 开始的时间要长得多。同样，这是一个可选状态。

要查看 CPUidle 驱动程序的可用空闲状态和其他统计，请输入：

```
~]$ cpupower idle-info
```

带有 "Nehalem" 微架构的最新 Intel CPU 具有新的 C-State、C6，它可以减少 CPU 为 0 的交错，但通常可将功耗减少 80% 到 90%。Red Hat Enterprise Linux 7 中的内核包括这个新 C-State 的优化。

3.2. CPUFREQ

在系统上减少功耗和 heat 输出的最有效方法是 CPUfreq。CPUfreq - 也称为 CPU 速度扩展 - 是 Linux 内核中的基础架构，允许扩展 CPU 频率以省电。可以根据系统负载、响应 ACPI 事件或用户空间程序手动进行 CPU 调用，并允许即时调整处理器的时钟速度。这可让系统以较低的时钟速度运行来省电。CPUfreq 调控器定义切换频率（无论是快速还是较慢的时钟速度）的规则。

3.2.1. CPUfreq 驱动程序

可以使用两个用于 CPUfreq 的驱动程序： ACPI CPUfreq 驱动程序和 Intel P-state 驱动程序。

ACPI CPUfreq

ACPI CPUfreq 驱动程序是一个内核驱动程序，它通过 ACPI 控制特定 CPU 的频率，这样可确保内核和硬件之间的通信。

Intel P-state

在 Red Hat Enterprise Linux 7 中，支持 Intel P-state 驱动程序。驱动程序提供了一个界面，用于根据 Intel Xeon E 系列架构或更新的架构控制处理器上的 P-state 选择。Intel P-state 实现 setpolicy （）回调。驱动程序根据 cpufreq 内核请求的策略决定使用的 P-state。如果处理器可以在内部选择其下一个 P-state，则驱动程序会将这一责任卸载到处理器。如果没有，则驱动程序实施算法来选择下一个 P-state。

Intel P-state 提供自己的 sysfs 文件来控制 P-state 选择。这些文件位于 /sys/devices/system/cpu/intel_pstate/ 目录中。对文件所做的任何更改都适用于所有 CPU。此目录包含用于设置 P-state 参数的五个文件：

- max_perf_pct: 限制驱动程序请求的最大 P-state，以可用性能百分比表示。可以通过 no_turbo 设置减少可用的 P-state 性能（请参阅以下）。
- min_perf_pct: min_perf_pct ：限制驱动程序请求的最小 P-state，以最大(no-turbo)性能级别表示的百分比。
- no_turbo: 将驱动程序限制为在 turbo frequency 频率范围下选择 P-state。
- turbo_pct ：显示 turbo 范围内的硬件支持的总性能百分比。这个号码与 turbo 是否已被禁用无关。

- num_pstates ：显示硬件支持的 P-states 数。这个号码与 turbo 是否已被禁用无关。

目前，在支持的 CPU 中默认使用 Intel P-state。通过在内核命令行中添加以下内容，用户可以使用 ACPI CPUfreq ：

```
intel_pstate=disable
```

3.2.2. CPUfreq Governors

CPUfreq governor 定义系统 CPU 的电源特征，后者又会影响 CPU 性能。每个 governor 在工作负载方面都有自己的独特行为、目的和适用性。本节论述了如何选择和配置 CPUfreq governor、每个调控器的特性以及每个调控器适合哪些工作负载。

警告

Red Hat Enterprise Linux 7 包含多个 core CPUfreq governor。默认情况下，Intel P-state 驱动程序在活动模式下运行，其中只有两个 CPUfreq governors 可用： performance 和 powersave。请注意，与相同名称的核心 CPUfreq governor 相比， performance 和 powersave Intel P-state CPUfreq governor 的功能是不同的。

3.2.2.1. Core CPUfreq Governors

Red Hat Enterprise Linux 7 中提供了不同类型的 CPUfreq governor ：

cpufreq_performance

Performance governor 强制 CPU 使用最高可能时钟频率。这个频率会被静态设置，且不会更改。因此，这个特定监管器 不提供节能功能。它只适用于数小时的工作负载，甚至仅在 CPU 中很少（或永不）闲置期间。

cpufreq_powersave

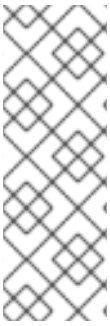
相反， Powersave governor 会强制 CPU 使用最低可能的时钟频率。

这对于可能会有过度负载问题的系统和环境中最有用。

cpufreq_ondemand

cpufreq_userspace

cpufreq_conservative



备注

这可让您在一天的指定时间自动设置特定的 governor。

3.2.2.2.

-
-
- ***被动模式***

带有硬件管理的 P-states 的活跃模式

当使用 HWP 的活跃模式时，Intel P-state 驱动程序指示 CPU 执行 P-state 选择。驱动程序可以提供频率提示。但是最终选择取决于 CPU 内部逻辑。

在带有 HWP 的活跃模式中，Intel P-state 驱动程序提供了两个 P-state 选择算法：

- *性能*
-

允许的 P-states 范围限制为驱动程序允许使用的范围的上限。

没有硬件管理的 P-states 活跃的模式

在使用没有 HWP 的活跃模式中，Intel P-state 驱动程序提供了两个 P-state 选择算法：

- *性能*
-

被动模式

所有可用的通用 CPUFreq 内核控制器都可使用。

3.2.3.

```
~]# cpupower frequency-info --governors
```

```
~]# cpupower frequency-set --governor [governor]
```

```
~]# cpupower -c 1-3,5 frequency-set --governor cpufreq_userspace
```

3.2.4.

-
-
-
-

-
-
-
-

重要的

-
-

注意

可以通过写入这些可调项来更改设置和值。

```
echo 360000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
```

3.3.

-
-
-

3.4.

--perf-bias <0-15>

--sched-mc <0|1|2>

--sched-smt <0|1|2>

3.5.

3.6.

/sys/devices/system/cpu/power/

control

auto

on

autosuspend_delay_ms

3.7.

default

powersave

performance

- ***pcie_aspm=off disables ASPM***
-

```
~]$ journalctl -b | grep ASPM
```

-
-

3.8.

min_power

medium_power

max_performance

3.9.

3.10.

3.11.

3.12. RFKILL

```
~]# rfkill block 0
```

例如：

```
~]# rfkill block wifi
```

```
~]# rfkill block all
```

第 4 章 使用案例

4.1.

Webserver

Web 服务器需要网络和磁盘 I/O。根据外部连接速度 100 Mbit/s 可能已经足够。如果机器服务于静态页面，则 CPU 性能可能并不重要。因此，电源管理选择可能包括：

- 没有适用于 tuned 的磁盘或网络插件。
- 启用 ALPM。
- ondemand governor 开启。
- 网卡限制为 100 Mbit/s。

Compute 服务器

计算服务器主要需要 CPU。电源管理选择可能包括：

- 根据作业以及数据存储发生的位置、调优的磁盘或网络插件；或对于批处理模式系统，完全活跃的 tuned 。
- 根据利用率，可能是 performance governor。

Mailserver

邮件服务器大多需要磁盘 I/O 和 CPU。电源管理选择可能包括：

- ondemand governor 打开 on，因为最后几个 CPU 性能不重要。
- 没有适用于 tuned 的磁盘或网络插件。
- 网络速度不应有限，因为邮件通常是内部的，因此可以从 1 Gbit/s 或 10 Gbit/s 链接中受益。

fileserver

fileserver 要求与 mailserver 的不同，但根据所使用的协议，可能需要更多的 CPU 性能。通常，基于 Samba 的服务器需要超过 NFS 的 CPU，NFS 通常需要超过 iSCSI。即使如此，您也应该可以使用 ondemand governor。

目录服务器

目录服务器通常对磁盘 I/O 的要求较低，特别是在有足够 RAM 时。网络延迟非常重要，尽管网络 I/O 较少。您可能会考虑延迟网络性能优化，并降低链路速度，但您应该根据您的特定网络仔细测试。

4.2. 示例 - LAPTOP

另外一个非常常见的情况，电源管理和节省可以真正区分笔记本电脑。在设计的笔记本电脑中，通常已使用比工作站或服务器的能源要低，而对于其他机器而言，绝对节省的可能性要少得多。但是，当处于电池模式时，任何保存都有助于从笔记本电脑中获得几分钟的电池寿命。虽然本节重点介绍电池模式中的笔记本电脑，但您一定的情况在 AC 电源上运行时仍可以使用其中一些或所有调优。

单个组件的节省通常比工作站在笔记本电脑上产生较大的相对差异。例如，在 100 Mbits/s 运行的 1 Gbit/s 网络接口会保存 3-4 watts。对于总功耗为 400 watts 的典型服务器，这种保存大约为 1 %。在带有大约 40 个 watts 的总功耗的笔记本电脑上，将这个组件数的节能到总计 10 %。

典型笔记本电脑上的特定节能优化包括：

- 将系统配置为禁用您不使用的所有硬件。例如，并行或串行端口、卡读卡器、Webcams、WiFi 和 Bluetooth 仅命名几个可能的候选者。
- DIM 在 darker 环境中显示，您不需要完全修改来读取屏幕。在 GNOME 桌面上使用 系统+首

选项 电源管理，Kickoff Application Launcher+Computer+System Settings+Advanced → Power Management on KDE 桌面；或者在命令行中 gnome-power-manager 或 xbacklight，或者笔记本电脑上的功能键。

另外，（或）您可以对各种系统设置执行很多小的调整：

- 使用 ondemand governor （在 Red Hat Enterprise Linux 7 中默认启用）
- 启用 AC97 音频节能（在 Red Hat Enterprise Linux 7 中默认启用）：

```
~]# echo Y > /sys/module/snd_ac97_codec/parameters/power_save
```

- 启用 USB 自动挂起：

```
~]# for i in /sys/bus/usb/devices/*/power/autosuspend; do echo 1 > $i; done
```

请注意，USB 自动挂起无法适用于所有 USB 设备。

- 使用 relatime (Red Hat Enterprise Linux 7 中的默认设置)挂载文件系统：

```
~]# mount -o remount,relatime mountpoint
```

- 将屏幕亮度减少到 50 或更少，例如：

```
~]$ xbacklight -set 50
```

- 激活 DPMS 以进行屏幕闲置：

```
~]$ xset +dpms; xset dpms 0 0 300
```

- 取消激活 Wi-Fi ：

```
~]# echo 1 > /sys/bus/pci/devices/*/rf_kill
```

附录 A. 开发人员提示

每个良好的编程书都涵盖内存分配和特定功能性能的问题。在开发软件时，请注意可能会增加软件运行系统的功耗的问题。虽然这些注意事项不会影响每一行代码，但您可以优化代码在性能频繁瓶颈的区域。

一些通常有问题的技术包括：

- 使用线程。
- 不必要的 CPU 唤醒，且不有效地使用唤醒。如果需要唤醒，请一次性执行所有操作（停止闲置），并尽快进行。
- 不必要地使用 [f]sync()。
- 不必要的活动轮询或使用短、常规超时。（改为使用事件）。
- 不有效地使用唤醒。
- 磁盘访问效率低。使用大型缓冲区以避免频繁访问磁盘。一次编写一个大块。
- 计时器的使用效率低下。如果可能，跨应用程序（甚至跨系统）分组计时器。
- 过量 I/O、功耗或内存使用情况（包括内存泄漏）
- 执行不必要的计算。

以下部分更详细地检查其中的一些区域。

A.1. 使用线程

众所周知，使用线程使应用程序性能更好、更快，但在每一个情况下都不是如此。

Python

Python 使用 Global Lock Interpreter[1]因此，线程处理只适用于更大的 I/O 操作。unladen-swallow [2] 是更快速的 Python 实施，您可能需要优化代码。

Perl

Perl 线程最初是为没有分叉（如具有 32 位 Windows 操作系统的系统）的系统上运行的应用程序创建的。在 Perl 线程中，会为每个单个线程复制数据（复制 On Write）。默认情况下，数据不共享，因为用户应该能够定义数据共享级别。对于共享 threads::shared 模块的数据，必须包含它们。但是，数据不仅会被复制（复制 On Write），但模块还会为数据创建绑定的变量，这需要更长的时间，甚至较慢。[3]

C

c 线程共享相同的内存，每个线程都有自己的堆栈，且内核不必创建新的文件描述符并分配新的内存空间。C 真正可以将更多 CPU 支持用于更多线程。因此，要最大化线程的性能，请使用 C 或 C++ 等低级别语言。如果您使用脚本语言，请考虑编写 C 绑定。使用配置文件器确定执行代码不佳的部分。[4]

A.2. WAKE-UPS

许多应用会扫描配置文件以进行更改。在很多情况下，扫描以固定间隔执行，例如每分钟的一次。这可能会有问题，因为它会强制磁盘从启动中唤醒。最好的解决方法是找到一个良好的间隔、一个良好的检查机制，或使用 inotify 来检查对事件的更改。inotify 可以查看文件或目录的各种更改。

例如：

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/inotify.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
  int fd;
  int wd;
  int retval;
  struct timeval tv;

  fd = inotify_init();
```

```
  /* checking modification of a file - writing into */
  wd = inotify_add_watch(fd, "./myConfig", IN_MODIFY);
  if (wd < 0) {
    printf("inotify cannot be used\n");
    /* switch back to previous checking */
  }

  fd_set rfds;
  FD_ZERO(&rfds);
  FD_SET(fd, &rfds);
  tv.tv_sec = 5;
  tv.tv_usec = 0;
  retval = select(fd + 1, &rfds, NULL, NULL, &tv);
  if (retval == -1)
    perror("select()");
  else if (retval) {
    printf("file was modified\n");
  }
  else
    printf("timeout\n");

  return EXIT_SUCCESS;
}
```

这种方法的优点是您可以执行的各种检查。

主要的限制是，系统中只有有限数量的监视可用。数字可以从 /proc/sys/fs/inotify/max_user_watches 获取，尽管它可以更改，但不建议这样做。另外，如果 inotify 失败，代码必须回退到不同的检查方法，这通常意味着源代码中出现很多 #if #define。

有关 inotify 的详情，请查看 inotify(7) man page。

A.3. FSYNC

Fsync 称为 I/O 昂贵的操作，但这并不完全正确。

每次点击链接时，Firefox 用于调用 sqlite 库，以转至新页面。名为 fsync 的 SQLite 和因为文件系统设置（主要是 ext3 带有数据排序模式）的 SQLite，在没有发生任何发生时会有一个较长的延迟。如果另一个进程同时复制大型文件，则可能需要很长时间（最多 30 秒）。

然而，在其它情况下，如果 fsync 没有被全部使用，交换机会出现到 ext4 文件系统的问题。Ext3 设置为数据排序模式，每几秒钟刷新内存并将其保存到磁盘中。但是，在使用 ext4 和 laptop_mode 时，保存的间隔较长，数据可能会在系统意外关闭时丢失。现在 ext4 被修补，但我们仍然必须仔细考虑应用程序的设计，并根据情况使用 fsync。

以下简单示例显示了如何备份文件或如何丢失数据：

```
/* open and read configuration file e.g. ./myconfig */
fd = open("./myconfig", O_RDONLY);
read(fd, myconfig_buf, sizeof(myconfig_buf));
close(fd);
...
fd = open("./myconfig", O_WRONLY | O_TRUNC | O_CREAT, S_IRUSR | S_IWUSR);
write(fd, myconfig_buf, sizeof(myconfig_buf));
close(fd);
```

更好的方法是：

```
/* open and read configuration file e.g. ./myconfig */
fd = open("./myconfig", O_RDONLY);
read(fd, myconfig_buf, sizeof(myconfig_buf));
close(fd);
...
fd = open("./myconfig.suffix", O_WRONLY | O_TRUNC | O_CREAT, S_IRUSR | S_IWUSR
write(fd, myconfig_buf, sizeof(myconfig_buf));
fsync(fd); /* paranoia - optional */
...
close(fd);
rename("./myconfig", "./myconfig~"); /* paranoia - optional */
rename("./myconfig.suffix", "./myconfig");
```

---

[1] http://docs.python.org/c-api/init.html#thread-state-and-the-global-interpreter-lock

[2] http://code.google.com/p/unladen-swallow/

[3] http://www.perlmonks.org/?node_id=288022

[4] http://people.redhat.com/drepper/lt2009.pdf

附录 B. 修订历史记录

修订 2.2-9 Mon Aug 05 2019 Marie Doleželová
7.7 GA 出版物的文档版本。

修订 2.2-6 Mon Jul 24 2017 Marie Doleželová
7.4 GA 出版物的文件版本。

修订 2.2-5 Tue Mar 21 2017 Milan Navrátil
异步更新： Tuned 章节重写

修订 2.0-2 Fri Oct 14 2016 Marie Doleželová
7.3 GA 出版物版本。

修订 2.0-1 Wed 11 Nov 2015 Jana Heves
7.2 GA 版本。

修订 1-3 Fri 19 Jun 2015 Jacquelynn East
修复了 Core Infrastructure 和 Mechanics 中错误的软件包名称。

修订 1-2 Wed 18 Feb 2015 Jacquelynn East
7.1 GA 版本

修订 1-1 Thu Dec 4 2014 Jacquelynn East
7.1 Beta 版本

修订 1.0-9 Tue Jun 9 2014 Yoana Ruseva
7.0 GA 版本

修订 0.9-1 Fri May 9 2014 Yoana Ruseva
为风格的更改重建。

修订 0.9-0 Wed May 7 2014 Yoana Ruseva
本书的 Red Hat Enterprise Linux 7.0 发行版用于审核目的。

修订 0.1-1 Thu Jan 17 2013 Jack Reed
从文档的 Red Hat Enterprise Linux 6 版本中分支