Red Hat Enterprise Linux 6

电源管理指南

在红帽企业版 Linux 6 中管理电源消耗

版 1.0

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 电源管理指南

在红帽企业版 Linux 6 中管理电源消耗
版 1.0

Don Domingo
红帽 工程内容服务

Rüdiger Landmann
红帽 工程内容服务
r.landmann@redhat.com

Red Hat Inc.

法律通告

摘要

这个文档解释了如何在红帽企业版 Linux 6 系统中有效管理您的电源消耗。下面的部分讨论了降低电源消耗的不同技术（服务器和笔记本电脑），以及每个技术对您系统总体性能的影响。

目录

第 1 章 概述

电源管理是我们改进红帽企业版 Linux 6 的关注点之一。限制计算机的电源消耗是*绿色 IT*（环保、友好的计算）最重要的部分之一，它可延伸到很多方面，其中包括使用可回收材料、硬件产品对环境的影响以及设计和部署系统时的环保意识。在这篇文档中我们将提供有关运行红帽企业版 Linux 6 系统的电源管理的指导和信息。

1.1. 电源管理的重要性

电源管理的核心是了解如何有效优化每个系统组件的能源消耗。这涉及到对您系统执行的不同任务的研究，并配置每个组件以便确定其性能刚好适用于该任务。

进行电源管理的主要原因是：

- 减少电源总耗以便降低成本

正确使用电源管理可有如下结果：

- 为服务器和计算中心进行降温
- 降低二次成本，其中包括冷却、空间、电缆、发电机以及*不间断供电*（UPS）
- 延长笔记本电脑电池寿命
- 降低二氧化碳排放
- 达到政府法律法规中对绿色 IT 的要求，例如：能耗星级
- 达到公司对新系统的要求

通常，降低特定组件（或者整个系统）的电源消耗将降低散热和性能。因此，您应该彻底学习并测试您进行的任何配置可承受的性能降低，特别是对关键任务系统。

通过研究您系统执行的不同任务以及每个组件的配置确定其性能刚好满足该任务的需要，这样您可以节省能源，减少散热，并为笔记本电脑优化电池寿命。有关电源消耗的分析和调整系统的很多原则与性能调整相似。在有些情况下，电源管理和性能调整是从对立方向进行系统配置，因为系统通常是根据性能或者电源进行优化的。本手册阐述了一些红帽提供的工具以及我们开发用来帮助您完成这一过程的技术。

红帽企业版 Linux 6 附带了很多新的电源管理特性，它们是默认启用的。所选的特性都不会对典型服务器或者桌面系统使用产生影响。但是对绝对需要最大吞吐量、最小延迟或者最高 CPU 性能的非常具体的使用案例则可能需要审查那些默认特性。

要决定您是否应该使用本文档中阐述的技术优化您的机器，请先问您自己几个问题：

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有要遵循的准则，或者是否有您必须执行的规定。

问： **优化的代价是什么？**

答： 我们给出的几个技术并不需要您完成详细审核和分析您的机器的整个过程，我们提供一组可改进您电源使用的常用优化。当然它们对系统的优化不如手动审核和优化的系统，但提供了一个好的折衷方案。

问： **优化是否会将系统性能降低到无法接受的程度？**

答：这个文档中描述的大多数技术都会显著影响您系统的性能。如果您选择使用使用比红帽企业版 Linux 6 中默认的现有电源管理更多的功能，您应该在电源优化后监控您的系统性能并决定性能损失是否可以接受。

问： **优化消耗的时间和资源是否超过了得到的结果？**

答： 手动完成优化单一系统的整个过程通常得不偿失，因为消耗的时间和费用比您在单一机器中获得的好处不成比例。另外，比如说您在办公室使用 10000 个桌面系统，它们的配置和设置都一样，那么生成一个优化的设置并在 10000 台机器中使用就是事半功倍了。

下面的部分解释了如何优化硬件性能以便使您的系统在节能方面受益。

1.2. 电源管理基础

有效电源管理是建立在以下原则上的：

只在需要时唤醒闲置的 CPU

红帽企业版 Linux 5 内核曾为每个 CPU 使用周期计时器。这个计时器防止 CPU 真正进入闲置状态，因为它要求 CPU 无论是否有进程在运行，都要处理每个计时器事件（根据设置几微妙出现一次）。有效电源管理的主要任务是减少 CPU 唤醒的频率。

因此，红帽企业版 Linux 6 中的 Linux 内核删除了周期计时器：结果是现在 CPU 的闲置状态是*无空循环*。这就防止了 CPU 在闲置状态下消耗不必要的电力。但是如果您的系统中有生成不必要计时器事件的应用程序就会抵消这个功能带来的好处。查询事件（比如音量更改、鼠标移动等）就属于此类事件。

红帽企业版 Linux 6 包括您可用来根据其 CPU 用量识别审核程序的工具。详情请参考 第 2 章 *电源管理工具审核及分析*。

应该完全禁用不使用的硬件和设备

特别是在那些有移动组件的设备（比如硬盘）中尤其重要。另外，有些应用程序可 "open" 未使用但启用的设备，当出现这种情况时，内核假设设备处于使用状态，这样就会阻止设备进入节电状态。

低活性可解读为低瓦数

在很多情况下要依赖先进的硬件并修正 BIOS 配置。旧的系统组件经常不支持一些红帽企业版 Linux 6 中支持的新特性。请确定您系统使用的是最新的官方固件，且在 BIOS 中的电源管理或者设备配置部分启用了电源管理功能。您需要查看的功能包括：

- 变频（SpeedStep）
- PowerNow!
- Cool'n'Quiet
- ACPI （C 状态）
- 智能

如果您的硬件支持这些特性，且在 BIOS 中启用了这些特性，红帽企业版 Linux 6 将默认使用这些特性。

不同的 CPU 状态形式及效果

先进 CPU 与*高级配置和电源接口*（ACPI）共同提供不同的电源状态。三种不同的状态为：

- 休眠（C 状态）
- 运行（P 状态）
- 散热（T 状态或者 "热状态"）

以最低休眠状态运行的 CPU 可能消耗最少的电力，但在需要将其从该状态唤醒时也需要相对较长的时间。在很少情况下 CPU 需要在刚刚进入休眠状态后就马上需要被唤醒。这种情况导致一个持久有效的忙碌 CPU，并在已经使用另一种状态时可能丧失潜在的节电效果。

关闭的机器使用的电量最少

很明显，最佳的节电方法就是关闭系统。例如：您的公司可开发一项注重 "绿色 IT" 的企业文化，让员工有意识地在午休时间或者下班时关闭机器。您还可以将几台物理服务器合并成一个较大的服务器，并使用我们在红帽企业版 Linux 6 中附带的虚拟化技术将其虚拟化。

第 2 章 电源管理工具审核及分析

2.1. 审核及分析概述

对单一系统进行详细手册审核、分析以及调整很少见，因为这样做的时间和费用消耗一般都会超过进行这些最后系统微调带来的好处。但是为可在基本一致的大量系统中重复使用相同的设置而执行这些任务是非常有用的，您可在所有系统中重复使用相同的设置。例如：想象一下部署成千个桌面系统，或者在机器都几乎完全一样的 HPC 群集中。另一个执行审核和分析的理由是提供进行比较的基准，您可使用它在将来识别系统行为退化或者改变。在常规进行硬件、BIOS 或者软件更新，且您不想被电源消耗吓到的情况下这些分析结果非常有用。通常完整的审核和分析可让您对具体系统中的情况有更好的了解。

根据电源消耗审核和分析系统比较困难，即便使用最先进的系统也是如此。大多数系统不会提供测量软件耗能的必需工具。也有例外情况：惠普服务器系统的 ILO 管理控制台有电源模块，您可使用它访问网页。IBM 在其刀片中心电源管理模块中提供类似的解决方案。在有些 Dell 系统中，IT 助理也提供电源监控功能。其他零售商也乐于为其服务器平台提供类似的功能，但没有一个所有零售商都支持的单一解决方案可用。如果您的系统没有内置测量电源消耗的机制，还有其它一些方法可用。您可以为您的系统安装特殊的电源，它可通过 USB 提供电源消耗信息。Gigabyte Odin GT 550 W PC 电源就是一个例子，软件从那些 Linux 系统中读取的值可用于 http://mgmt.sresortth.sze.hu/~andras/dev/gopsu/。最后，有些外用电表，比如 Watts up? 产品也有 USB 接口。

对电源消耗的直接测量通常只在尽可能节约能耗时需要。幸运的是还有其他方法可测量实际的更改以及系统的行为方式。本章描述了这些必需的工具。

2.2. POWERTOP

红帽企业版 Linux 6 中引进的无空循环内核（请参考 第 3.4 节 “无空循环内核”）可允许 CPU 更频繁地进入闲置状态，降低能耗并改进电源管理。新的 **PowerTOP** 工具可识别具体是哪些内核组件和用户空间程序经常唤醒 CPU。**PowerTOP** 用于开发执行 第 3.11 节 “在用户空间中优化” 中论述的审核，这样就在这个发行本中调整了很多程序，不必要的 CPU 唤醒降低了 10 倍。

使用以下命令安装 **PowerTOP**：

```
yum install powertop
```

使用以下命令运行 **PowerTOP**：

```
powertop
```

请注意：您运行 **PowerTOP** 将需要 root 特权方可允许程序有效运行。

当 **PowerTOP** 运行时会从系统中收集统计数据并为您显示最频繁向 CPU 发送唤醒请求的组件列表。**PowerTOP** 还会为降低系统能耗向您提出建议。这些建议出现在屏幕底部，并指定您要接受 **PowerTOP** 建议时需要按的按键。因为 **PowerTOP** 是周期性刷新，届时会有进一步的建议。在 图 2.1 “PowerTOP 操作中” 中，建议您**增加虚拟机脏回写时间**以及接受该建议时要按的按键（**W**）。

当 **PowerTOP** 运行时会从系统中收集统计数据并为您显示几个重要信息列表。列表顶端是您的 CPU 核处于可用 C 状态和 P 状态的时间长度。CPU 处于 C 状态或者 P 状态时间越长越好（**C4** 高于 **C3**），也显示了是如何更好利用 CPU 的。您的目标应该是 90% 或者以上的时间 CPU 处于 C 状态或者 P 状态，其余时间系统处于闲置状态。

第二项信息是机器每秒实际唤醒总数。每秒唤醒数让您根据系统电源消耗了解服务或者设备以及内核驱动程序是否运行良好。每秒唤醒数越大，耗能越多，因此这里数字越小越好。

下一步，如果可能，**PowerTOP** 提供系统电源使用的估计值。预计 **PowerTOP** 会在使用电池模式的笔记本电脑中报告这个数字。

所有可用耗电估计值都跟着一个最常向 CPU 发送唤醒的组件详细列表。列表的顶端是那些您需要更详细调查以便优化系统降低能耗的组件。如果它们是内核组件（组件名称被列在 **<>** 中），那么唤醒通常与导致这些唤醒的特定驱动程序关联。调整驱动程序通常需要更改内核，这已经超出本文档涉及范围。但是，发送唤醒的用户空间进程更容易管理。首先要确认这个服务或者应用程序是否应该在这个系统中运行。如果不是，只要失活它即可。要永久关闭某个服务，请运行：

```
chkconfig servicename off
```

如果您需要了解该组件具体执行任务的详细信息，请运行：

```
ps -awux | grep componentname
strace -p processid
```

如果追踪看来像是不断的重复，那么您可能已经找到了忙碌回路。要修复这个问题需要更改那个组件的代码，但这已经超出本文档的范围。

最后，**PowerTOP** 还会为降低系统能耗向您提出建议。这些建议出现在屏幕底部，并指定您要接受 **PowerTOP** 建议时需要按的按键。因为 **PowerTOP** 是周期性刷新，届时会有进一步的建议。在 图 2.1 “PowerTOP 操作中” 中，备注建议您**增加虚拟机脏回写时间**以及接受该建议时要按的按键（**W**）。这些更改要在下一次引导时生效。要使这些更改持久，**PowerTOP** 会显示执行这个优化具体要执行的命令。请使用您的首选文本编辑器在您的 **/etc/rc.local** 文件中添加命令以便其在每次启动计算机时都有效。

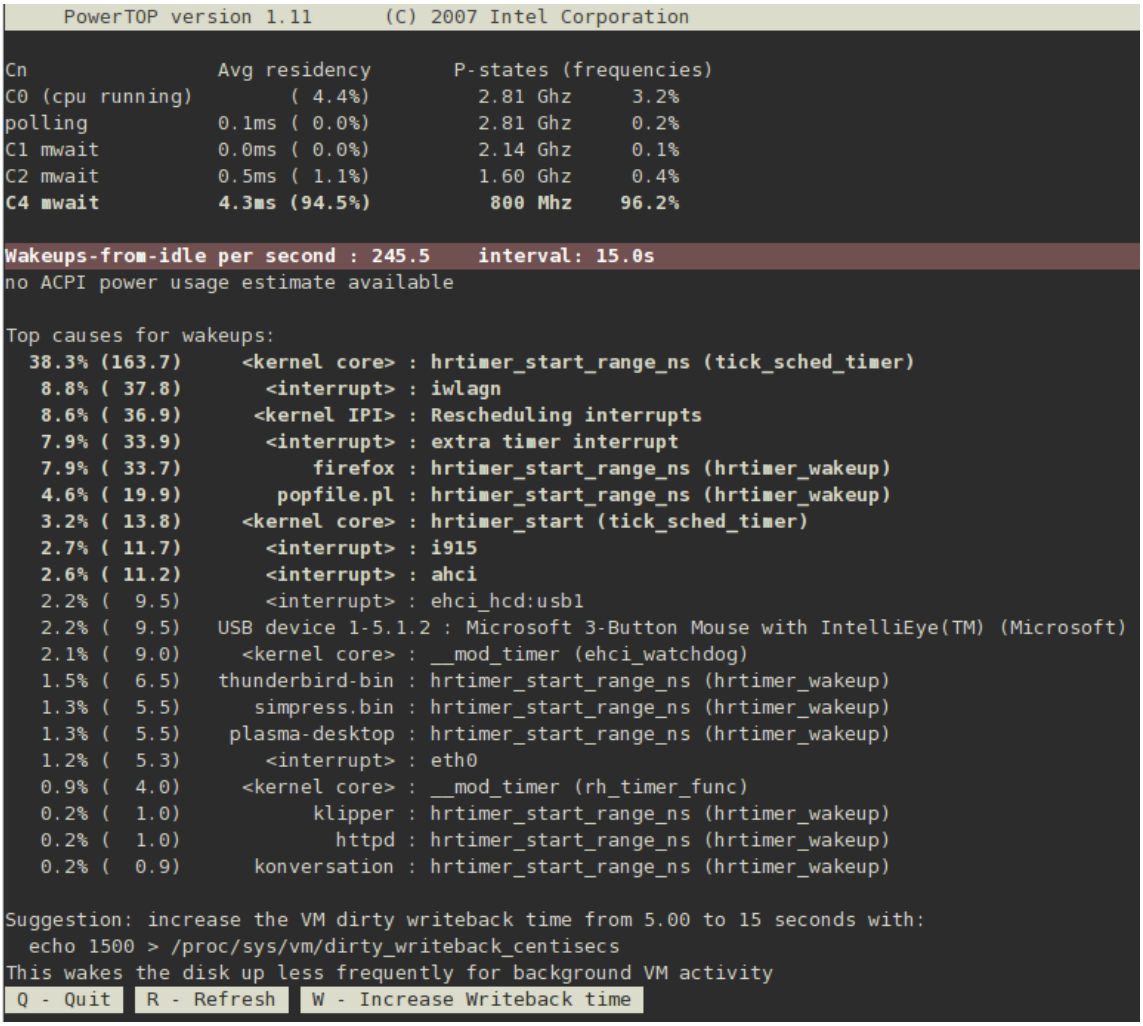

图 **2.1. PowerTOP 操作中**

Less Watts 网站发布了由 **PowerTOP** 识别的保持 CPU 活跃的应用程序列表。请参考 http://www.lesswatts.org/projects/powertop/known.php。

2.3. DISKDEVSTAT 和 NETDEVSTAT

Diskdevstat 和 **netdevstat** 是收集有关磁盘活性以及在系统中运行的所有程序的网络活性的 **SystemTap** 工具。这些工具是由 **PowerTOP** 支配，并可显示每秒钟每个程序唤醒 CPU 的次数（请参考 第 2.2 节 “PowerTOP”）。这些工具收集的统计可让您识别那些使用很多小 I/O 操作的程序，它们比少量的较大操作更耗能。其它监控工具只是测量传输速度，无法帮助确认此类使用。

使用 **SystemTap** 的命令安装这些工具：

```
yum install systemtap tuned-utils kernel-debuginfo
```

使用如下命令运行这些工具：

```
diskdevstat
```

或者命令：

```
netdevstat
```

这两个命令最多可有三个参数，如下所示：

diskdevstat *update_interval total_duration display_histogram*

netdevstat *update_interval total_duration display_histogram*

update_interval

以秒为单位的显示更新间隔时间。默认：**5**

total_duration

以秒为单位显示整体运行时间。默认：**86400**（一天）

display_histogram

是否显示在运行结束时收集的所有数据柱形图的标签。

输出结果类似 **PowerTOP**。这里是来自使用 KDE 4.2 的 Fedora 10 系统中运行的 **diskdevstat** 详细输出结果示例：

```
  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
15494     0 sda1             0     0.000     0.000     0.000         758
0.000     0.012     0.000 0logwatch
15520     0 sda1             0     0.000     0.000     0.000         140
0.000     0.009     0.000 perl
15549     0 sda1             0     0.000     0.000     0.000         140
0.000     0.009     0.000 perl
15585     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
15429     0 sda1            0     0.000     0.000     0.000          62
0.009     0.009     0.000 crond
15379     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15473     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15415     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15433     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15425     0 sda1            0     0.000     0.000     0.000          62
0.007     0.007     0.000 crond
15375     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15477     0 sda1            0     0.000     0.000     0.000          62
0.007     0.007     0.000 crond
15469     0 sda1            0     0.000     0.000     0.000          62
0.007     0.007     0.000 crond
15419     0 sda1            0     0.000     0.000     0.000          62
0.008     0.008     0.000 crond
15481     0 sda1            0     0.000     0.000     0.000          61
0.000     0.001     0.000 crond
15355     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
15575     0 sda1            0     0.000     0.000     0.000          16
0.000     0.000     0.000 cat
15581     0 sda1            0     0.000     0.000     0.000          12
0.001     0.002     0.000 perl
15582     0 sda1            0     0.000     0.000     0.000          12
0.001     0.002     0.000 perl
15579     0 sda1            0     0.000     0.000     0.000          12
0.000     0.001     0.000 perl
15580     0 sda1            0     0.000     0.000     0.000          12
0.001     0.001     0.000 perl
15354     0 sda1            0     0.000     0.000     0.000          12
0.000     0.170     0.014 sh
15584     0 sda1            0     0.000     0.000     0.000          12
0.001     0.002     0.000 perl
15548     0 sda1            0     0.000     0.000     0.000          12
0.001     0.014     0.001 perl
15577     0 sda1            0     0.000     0.000     0.000          12
0.001     0.003     0.000 perl
15519     0 sda1            0     0.000     0.000     0.000          12
0.001     0.005     0.000 perl
15578     0 sda1            0     0.000     0.000     0.000          12
0.001     0.001     0.000 perl
15583     0 sda1            0     0.000     0.000     0.000          12
0.001     0.001     0.000 perl
15547     0 sda1            0     0.000     0.000     0.000          11
0.000     0.002     0.000 perl
15576     0 sda1            0     0.000     0.000     0.000          11
0.001     0.001     0.000 perl
```

```
15518     0 sda1            0     0.000     0.000     0.000          11
0.000     0.001     0.000 perl
15354     0 sda1            0     0.000     0.000     0.000          10
0.053     0.053     0.005 lm_lid.sh
```

这些列代表：

PID

应用程序的进程 ID

UID

运行中的应用程序用户 ID

DEV

发生 I/O 的设备

WRITE_CNT

写操作总数

WRITE_MIN

两个连续写入操作所需最短时间（以秒为单位）

WRITE_MAX

两个连续写入操作所需最长时间（以秒为单位）

WRITE_AVG

两个连续写入操作所需平均时间（以秒为单位）

READ_CNT

读操作的总数

READ_MIN

两个连续读操作所需最短时间（以秒为单位）

READ_MAX

两个连续读操作所需最长时间（以秒为单位）

READ_AVG

两个连续读操作所需平均时间（以秒为单位）

COMMAND

进程名称

在这个示例中可看到三个非常明显的应用程序：

```
  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
```

这三个应用程序的 **WRITE_CNT** 都大于 **0**，就是说它们在管理期间执行了一些写操作。其中 **plasma** 是最捣乱的程序：它执行的写操作作多，当然写操作平均间隔时间就最短。如果您关注不能有效利用能源的程序，那么 **Plasma** 就是最佳审查对象。

请使用 **strace** 和 **ltrace** 命令，通过追踪所有给定进程 ID 的系统调用更进一步检查应用程序。在这个示例中，您可以运行：

```
strace -p 2789
```

在这个示例中，**strace** 的输出结果中包含一个每 45 秒重复一次的图案，该图案打开用户的 KDE 图标缓冲文件，接着写入，然后马上再次关闭该文件。这导致一个在硬盘中的必需物理写入，因为已经更改了文件原数据（特别是修改时间）。最终修复是防止那些在没有图标更新时的不必要调用。

2.4. 电池寿命工具组件（BATTERY LIFE TOOL KIT）

红帽企业版 Linux 6 引进了 **Battery Life Tool Kit**（BLTK），它是一个模拟和分析电池寿命和性能的测试组件。BLTK 通过执行一组模拟特定用户组群的任务并报告结果达到此目的。虽然是特别用于测试笔记本电脑性能的开发，BLTK 还可使用 **-a** 在启动时报告桌面电脑性能。

BLTK 允许您生成比实际机器使用要容易得多的重复负载。例如：办**公**负载写入一个文本，在其中进行一些修改，并在一个电子表格中进行同样的操作。运行带**PowerTOP** 或者任何其它审核及分析工具的 BLTK，就可让您测试您执行的优化在机器被激活后，不处于闲置状态时是否有效。因为您可以在不同设置中多次运行同样的负载，并比较不同设置得到的结果。

使用以下命令安装 BLTK：

```
yum install bltk
```

使用以下命令运行 BLTK：

```
bltk workload options
```

例如：运行**闲置**负载 120 秒：

```
bltk -I -T 120
```

默认可用工作负载有：

-I, --idle

系统闲置，将其作为与其它负载进行比较的基准。

-R, --reader

模拟读取文档（默认为 **Firefox**）

-P, --player

模拟观看 CD 或者 DVD 驱动器中的多媒体文件（默认为 **mplayer**）

-O, --office

模拟使用 **OpenOffice.org** 套件编辑文件

其它可让您指定的选项：

-a, --ac-ignore

忽略是否交流电可用（桌面系统使用时需要）

-T *number_of_seconds*, --time *number_of_seconds*

运行测试的时间（以秒为单位），在**闲置**负载中使用这个选项。

-F *filename*, --file *filename*

指定特定负载使用的文件，例如**播放器**负载要使用的文件，而不是访问 CD 或者 DVD 驱动器。

-W *application*, --prog *application*

指定特定负载使用的应用程序，例如：**Firefox** 之外用于**读卡器**负载的浏览器。

BLTK 支持大量更具体的选项。详情请参考 **bltk** man page。

BLTK 保存 **/etc/bltk.conf** 配置文件指定目录生成的结果 -- 默认为 **~/.bltk/*workload*.results.*number*/**。例如：**~/.bltk/reader.results.002/** 目录中保存第三次**阅读器**测试负载的结果（第一次不计数）。结果保存在几个文本文件中。要将这些结果压缩成方便读取的格式，请运行：

```
bltk_report path_to_results_directory
```

现在结果出现在结果目录下名为 **Report** 的文件中。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 和 KTUNE

Tuned 是监控系统组件使用的守护进程，并可根据监控信息动态调整系统设置。在任意正常运行的给定系统中，为各种系统组件进行动态帐户调整的方法都不尽相同。例如：在启动和登录过程中会大量使用硬盘，但在之后用户主要使用类似 OpenOffice 或者电子邮件客户端等程序时就几乎不使用硬盘了。同样，不同时间对 CPU 和网络设备的使用是不同的。**Tuned** 监控这些组件的活动并在其使用中有所改动。

以典型办公室工作站为例。大多数时间里，以太网网络接口将会非常不活跃。过一段时间只有一些电子邮件进出，或者载入一些网页。对于那类负载，网络接口不一定在所有时间都按默认设置那样全速运行。**Tuned** 有一个用于网络设备的监控和调整插件，可用来探测低活性，并自动降低接口速度，通常也就降低了电力消耗。如果接口活性在较长时间段内大幅度增长，例如下载 DVD 映像或者打开有大附件的电子邮件，**tuned** 可探测到这个情况并将接口速度设定为最大以便在高活性等级时提供最佳性能。这个原则也用于其他 CPU 和硬盘插件。

网络设备不是默认配置为这样动作，因为速度更改需要有几秒钟之后方可生效，因此对用户体验有直接和可见的影响。同样的考虑也适用于 CPU 和硬盘调整插件。当硬盘转速降低时，它可能需要几秒钟时间重新提高转速，这将导致系统在此阶段有明显的反应延迟。CPU 插件的延迟副作用最小，但仍可测到，只是用户很难注意到而已。

现在除 **tuned** 外，我们还为您提供 **ktune**。**Ktune** 是由红帽企业版 Linux 5.3 作为用于具体使用案例的机器优化性能而引进的框架和服务。从那时起，**ktune** 有很大程度的改进，现在我们使用它作为一般调节框架的固定部分。它主要用于 第 2.5.2 节 “Tuned-adm” 中描述的不同的预定义侧写。

使用以下命令安装 tuned 软件包及其关联的 **systemtap** 脚本：

```
yum install tuned
```

安装 tuned 软件包还会在 **/etc/tuned.conf** 中设定一个示例配置文件，并激活默认侧写。

运行以下命令启动 **tuned**：

```
service tuned start
```

要在每次机器引导时启动 **tuned**，请运行：

```
chkconfig tuned on
```

Tuned 有可选选项，您可在手动运行该程序时使用。可用选项有：

-d, --daemon

将 **tuned** 作为守护进程启动，而不是在前台启动。

-c, --conffile

使用有具体名称和路径的配置文件，例如：**--conffile=/etc/tuned2.conf**。默认为 **/etc/tuned.conf**。

-D, --debug

使用日志记录的最高级别。

2.5.1. tuned.conf 文件

tuned.conf 文件包含 **tuned** 配置设定。默认情况下，它位于 **/etc/tuned.conf**，但您可以通过启动带 **--conffile** 选项的 **tuned.conf** 指定一个不同的名称和位置。

配置文件必须含有为 **tuned** 定义通用参数的 **[main]** 部分。该文件还应为每个插件包含一个部分。

[main] 部分包含以下选项：

interval

tuned 应该监控并调节系统的时间间隔，单位为秒。默认值为 **10**。

verbose

指定输出结果是否应该详细。默认值为 **False**。

logging

指定要记录信息的最小优先权。按降序排列允许的值有：**critical**、**error**、**warning**、**info** 和 **debug**。默认值为 **info**。

logging_disable

指定要记录信息的最大优先权，任何带这个优先权或者更低的优先权将不被记录。**critical**、**error**、**warning**、**info** 和 **debug**。**notset** 禁用这个选项。

每个插件都有其自身的片段，在方括号内指定插件名称。例如：**[CPUTuning]**。每个插件可有其自身的选项，但以下内容适用于所有插件：

enabled

指定是否启用该插件。默认值为 **True**。

verbose

指定输出是否应详细。如果没有为这个插件设定，该值会继承 **[main]** 中的值。

logging

指定要记录信息的最小优先权。如果没有为这个插件设定，该值会继承 **[main]** 中的值。

示例配置文件如下：

```
[main]
interval=10
pidfile=/var/run/tuned.pid
logging=info
logging_disable=notset

# Disk monitoring section

[DiskMonitor]
enabled=True
logging=debug

# Disk tuning section

[DiskTuning]
enabled=True
hdparm=False
alpm=False
logging=debug

# Net monitoring section

[NetMonitor]
enabled=True
logging=debug

# Net tuning section

[NetTuning]
enabled=True
logging=debug

# CPU monitoring section
```

```
[CPUMonitor]
# Enabled or disable the plugin. Default is True. Any other value
# disables it.
enabled=True

# CPU tuning section

[CPUTuning]
# Enabled or disable the plugin. Default is True. Any other value
# disables it.
enabled=True
```

2.5.2. Tuned-adm

通常具体的系统审核和分析非常耗时，且这样做也不节能。之前，唯一的备选方案是使用默认设置。因此，红帽企业版 Linux 6 **tuned-adm** 工具中包含在两种极端状态间可作为备选使用的具体案例中的独立侧写，这样就可让您使用命令行在这些侧写间轻松进行切换。红帽企业版 Linux 6 包含很多用于典型案例的预定侧写，您只要使用 **tuned-adm** 命令即可选择并激活它们，但您也要自己创建、修改并删除侧写。

要列出所有可用侧写并识别目前激活的侧写，请运行：

```
tuned-adm list
```

要只显示当前激活的侧写请运行：

```
tuned-adm active
```

要切换到某个可用侧写请运行：

```
tuned-adm profile profile_name
```

例如：

```
tuned-adm profile server-powersave
```

要禁用所有调整：

```
tuned-adm off
```

您第一次安装 **tuned** 时会激活 **default** 侧写。红帽企业版 Linux 6 还包括以下事先定义的侧写：

default

默认节能侧写。它在可用侧写节能中影响最小，只启用 CPU 和 **tuned** 磁盘插件。

desktop-powersave

在桌面系统中使用的节能侧写。为 SATA 主机适配器以及 CPU、以太网和 **tuned** 磁盘插件启用 ALPM 节能（请参考 第 3.6 节 “主动连接电源管理”）。

server-powersave

在服务器系统中使用的节能侧写。为 SATA 主机适配器启用 ALPM 节能，禁止通过 **HAL** 调用光驱（请参考 hal-disable-polling man page）并激活 CPU 和 **tuned** 磁盘插件。

laptop-ac-powersave

在使用 AC 运行的笔记本电脑中使用的中度影响节能侧写。为 SATA 主机适配器、WiFi 节能以及 CPU、以太网和 **tuned** 磁盘插件启用 ALPM 节能。

laptop-battery-powersave

在使用电池运行的笔记本电脑中使用的高度影响节能侧写。它从之前的侧写中激活所有节能机制，并为低唤醒系统启用多核节能调度程序，确定按需调节器处于活跃状态，并弃用了 AC97 音频节能。您可以使用这个侧写在任何类型的系统中最大限度节能，不仅仅限于在使用电池的笔记本电脑中使用。使用这个侧写的代价是对性能的明显影响，特别是磁盘和网络 I/O 延迟。

throughput-performance

用于典型吞吐性能调整的服务器侧写。它可禁用 **tuned** 和 **ktune** 节能机制，启用 **sysctl** 设置改进您的磁盘和网络 I/O 吞吐性能，并切换到 **deadline scheduler**。

latency-performance

用于典型延迟性能调整的服务器侧写。它可禁用 **tuned** 和 **ktune** 节能机制，启用 **sysctl** 设置改进您的磁盘和网络 I/O 延迟性能。

所有侧写都保存在 **/etc/tune-profiles** 下的独立子目录中。因此 **/etc/tune-profiles/desktop-powersave** 包含所有需要的文件以及那个侧写的设置。每个目录最多包含四个文件：

tuned.conf

为这个侧写激活调整服务的配置。

sysctl.ktune

ktune 使用的 **sysctl** 设置。其格式与 **/etc/sysconfig/sysctl** 文件一致（请参考 sysctl 和 sysctl.conf man page）。

ktune.sysconfig

ktune 的自身配置文件，通常为 **/etc/sysconfig/ktune**。

ktune.sh

ktune 服务使用的 **init** 风格的 shell 脚本，可在系统启动过程中运行特定命令调整系统。

启动一个新侧写的最简单方法就是复制现有的侧写。**laptop-battery-powersave** 侧写中已经包含丰富的调整设置，因此是有益的起始点。只要将这个目录复制到如下名称的新侧写中即可：

```
cp -a /etc/tune-profiles/laptop-battery-powersave/ /etc/tune-
profiles/myprofile
```

在新侧写中修改文件以达到您的个人需要。例如：如果您要探测 CD 更改，则您可以通过注释出 ktune.sh 脚本中的相关行来禁用那个优化：

```
# Disable HAL polling of CDROMS
# for i in /dev/scd*; do hal-disable-polling --device $i; done > /dev/null
2>&1
```

2.6. DEVICEKIT-POWER 和 DEVKIT-POWER

在红帽企业版 Linux 6 中，**DeviceKit-power** 承担原本是 **HAL** 一部分的电源管理功能，以及在之前的红帽企业版 Linux 发行本中的 **GNOME Power Manager** 的一部分功能（还可参考 第 2.7 节 “GNOME 电源管理器”）。**DeviceKit-power** 提供守护进程、API 和一组命令行工具。系统中的每个电源都使用一个设备代表，无论是否为物理电源。例如：笔记本电脑电池和交流电源都使用设备代表。

您可使用 **devkit-power** 命令访问命令行工具，以下是其选项：

--enumerate, -e

显示系统中每个电源设备的对象路径，例如：

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

--dump, -d

显示系统中所有电源设备的参数。

--wakeups, -w

显示系统中的 CPU 唤醒。

--monitor, -m

监视系统电源更换，例如：连接或者断开交流电源，或者电池耗尽。按 **Ctrl+C** 停止监视系统。

--monitor-detail

监视系统电源更换，例如：连接或者断开交流电源，或者电池耗尽。**--monitor-detail** 选项会显示比 **--monitor** 选项更详细的情况。按 **Ctrl+C** 停止监视系统。

--show-info *object_path*, -i *object_path*

显示某个具体对象路径的所有可用信息。例如：获得有关系统中对象路径 **/org/freedesktop/UPower/DeviceKit/power/battery_BAT0** 代表的电池的信息，请运行：

```
devkit-power -i /org/freedesktop/UPower/DeviceKit/power/battery_BAT0
```

2.7. GNOME 电源管理器

GNOME Power Manager 是作为 GNOME 桌面一部分安装的守护进程。**GNOME Power Manager** 在之前红帽企业版 Linux 版本中提供的很多电源管理功能已经在红帽企业版 Linux 6 中成为 **DeviceKit-power** 的一部分（请参考 第 2.6 节 “DeviceKit-power 和 devkit-power”）。但是 **GNOME Power Manager** 仍保留前端功能。在系统托盘中 **GNOME Power Manager** 可使用同一程序通知您系统电源状态的改变，例如：从电池更换到交流电。它还可报告电池状态并在电池电量低时提出警告。

GNOME Power Manager 还可允许您配置一些基本电源管理设置。要访问这些设置，请点击系统托盘中的 **GNOME Power Manager** 图标，然后点击**「属性」**。

「电源管理属性」页面中有三个标签：

- **「使用交流电」**

- **「使用电池」**
- **「常规」**

使用**「使用交流电」**和**「使用电池」**标签指定必须经过多长时间方可关闭不活跃系统中的显示，必须经过多长时间方可将不活跃系统置于睡眠状态，以及是否在不使用系统时停止磁盘驱动器。**「使用电池」**标签还可允许您设置显示亮度并选择系统处于非常低电池容量时的行为。例如：默认情况下 **GNOME Power Manager** 在电池容量达到一定程度时使系统休眠。使用**「常规」** 标签设定系统中的（物理）电源按钮和挂起按钮行为，并指定在什么情况下在系统托盘中显示 **GNOME Power Manager** 图标。

2.8. 其他审核方法

红帽企业版 Linux 6 提供很多执行系统审核及分析的工具。大多数可作为附加信息资源使用，以防您需要对现有发现进行验证或者您需要对某部分信息进行更深入的研究。很多这些工具也可用于性能调试。它们包括：

vmstat

vmstat 为您给出有关进程、内存、paging、块 I/O、陷阱以及 CPU 活性的详细资料。用它可进一步查看系统都做了什么，以及什么地方忙碌。

iostat

iostat 与 **vmstat** 类似，但只在块设备中是这样。它还提供更多详细输出和统计。

blktrace

blktrace 是一个非常详细的块 I/O 追踪程序。它将信息截成与程序关联的单一块。与 **diskdevstat** 合并使用时非常有用。

第 3 章 核心基础结构及技巧

3.1. CPU 闲置状态

使用 x86 构架的 CPU 支持不同的状态，在这些状态中部分 CPU 会被取消激活或者以低性能设置运行。这些状态，也就是我们知道的 *C 状态*，允许系统通过部分取消激活其不使用的 CPU 达到节能的目的。C 状态从 C0 开始用数字计算，数字越大代表 CPU 功能降低越多，也就越节能。虽然给定数字的 C 状态在不同处理器间类似，但为特定处理器或者处理器产品线使用的特定 C 状态的含义是特定的。C 状态 0-3 定义如下：

C0

操作或者运行状态。在这个状态中，CPU 处于工作状态，完全没有空闲。

C1, 挂起

处理器不执行任何步骤的状态，但通常不处于较低功率状态。CPU 可继续进行处理而没有延迟。所有提供 C 状态的处理器都需要支持这个状态。奔腾 4 处理器支持改进的 C1 状态，即 C1E，它实际上是一个低能耗状态。

C2, 时钟停止

在这个状态中处理器停止时钟，但它让其暂存器和缓冲保持完整状态，因此重新启动时钟后，它可以立即重新启动处理。这是一个可选状态。

C3, 休眠

处理器真正进入睡眠状态且不需要保存保持更新其缓冲。因此从这个状态唤醒的时间要大大长于从 C2 唤醒的时间。这也是一个可选状态。

最近使用 "Nehalem" 微构架的 Intel CPU 有新的 C 状态，即 C6。它可将供应 CPU 的电压降低到 0，但通常的节能率在 80% 到 90% 之间。红帽企业版 Linux 6 中的内核包括对这个新 C 状态的优化。

3.2. 使用 CPUFREQ 调节器

减少系统电力消耗和散热的最有效的方法就是使用 CPUfreq。CPUfreq -- 也称 CPU 速度计，即允许随时调整处理器时钟速度。这让系统可在降低的时钟速度下运行以便节电。更改频率的规则，无论是加快还是减慢时钟速度，以及何时更改频率，都在 CPUfreq 调节器中定义。

调节器定义系统的电源属性，它可影响 CPU 性能。每个调节器有其自身的独特负载行为、目的和实用性。这部分描述了如何选择和配置 CPUfreq 调节器，每个调节器的属性以及每个调节器适用的负载种类。

3.2.1. CPUfreq 调节器类型

本节列出了红帽企业版 Linux 6 中可用的不同 CPUfreq 调节器类型。

cpufreq_performance

性能调节器强制 CPU 使用可能的最高时钟频率。这个频率是静态设置的，不会改变。因此，这个特定的调节器*不提供节能效益*。它只适用于几个小时的高负载，且即使在那种情况下也只可用于 CPU 几乎不（或者从不）空闲的时候。

cpufreq_powersave

相反，节电调节器强制 CPU 使用最低可用时钟频率。这个频率将被静态设置，且不会更改。因此，这个特定调节器提供最大节能效益，但是以*最低 CPU 性能*为代价的。

这里"节电"有时是不正确的，因为（基本上）满负载的低速 CPU 消耗的能量比没有负载的高速 CPU 要多。因此，当建议在需要低性能时将 CPU 设定为使用节电调节器时，意外的高负载可能会导致系统实际消耗了更多的能量。

节电调节器对 CPU 简单来说更象是"限速器"而不是"节能器"。在过热时会出问题的系统和环境中最有用。

cpufreq_ondemand

按需调节器是一个动态调节器，它可允许 CPU 在系统负载高时达到最大时钟频率，还允许系统处于闲置时使用最小时钟频率。虽然这允许系统根据系统负载调整电源消耗，但也确实要承受*频率切换间造成的延迟*。因此，如果系统在闲置和高负载间切换过于频繁，那么延迟可抵消任何按需调节器带来的性能/节能优势。

对大多数系统来说，按需调节器可在散热、电源消耗、性能以及管理性间提供最佳折衷方案。当系统只在每天的某个具体时间繁忙时，按许调节器将根据负载自动在最大和最小频率间切换而无须进一步操作。

cpufreq_userspace

用户空间调节器允许用户空间程序（或者任何以 root 用户运行的进程）设定频率。这个调节器通常与 **cpuspeed** 守护进程一同使用。在所有调节器中，用户空间调节器是最可自定义的，且根据其配置，它可为您的系统在性能和耗能间提供最佳平衡。

cpufreq_conservative

与按需调节器类似，传统调节器还根据用量调整时钟频率（类似按需调节器）。但是按需调节器的方式更极端（从最大到最小，再返回），传统调节器在更接近的频率间切换。

这意味着传统调节器会将时钟频率调整为它认为适合负载的频率，而不是简单的在最大和最小频率间选择。虽然这样可极大节省能源消耗，但它的代价是比按需调节器要损失*更多的延迟*。

注意

您可以使用 **cron** 任务启用调节器。这允许您自动在每天的特定时间设定具体调节器。因此，您可以在闲置时指定低频率调节器（例如工作之余），并在高负载时返回高频率调节器。

有关如何启用特定调节器，请参考 第 3.2.2 节 “CPUfreq 设置” 中的 过程 3.2, “启用 CPUfreq 调节器”。

3.2.2. CPUfreq 设置

在选择和配置 CPUfreq 调节器前，您首先需要添加正确的 CPUfreq 驱动程序。

过程 3.1. 如何添加 CPUfreq 驱动程序

1. 使用以下命令查看可用于您系统的 CPUfreq 驱动程序：

```
ls /lib/modules/[kernel
version]/kernel/arch/[architecture]/kernel/cpu/cpufreq/
```

2. 使用 **modprobe** 添加正确的 CPUfreq 驱动程序。

```
modprobe [CPUfreq driver]
```

您使用以上命令时，请确定删除 **.ko** 文件名后缀。

重要

当选择适当的 CPUfreq 驱动程序时，总是选择 **p4-clockmod** 中的 **acpi-cpufreq**。当使用 **p4-clockmod** 驱动程序降低 CPU 的时钟频率时，它不会降低电压。另一方面，**acpi-cpufreq** 会随着 CPU 时钟频率降低电压，允许每个性能单位降低有更少电力消耗和散热。

3. 设定 CPUfreq 驱动程序后，您可以查看系统目前正在使用的调节器：

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

您还可以查看某个具体 CPU 可使用的调节器，请使用：

```
cat /sys/devices/system/cpu/[cpu ID]/cpufreq/scaling_available_governors
```

有些 CPUfreq 调节器可能您无法使用。在这个情况下，请使用 **modprobe** 添加可启用您要使用的具体 CPUfreq 调节器的内核模块。您可在 **/lib/modules/*[kernel version]*/kernel/drivers/cpufreq/** 中找到这些内核模块。

过程 3.2. 启用 CPUfreq 调节器

1. 如果没有为您的 CPU 列出可用的具体调节器，请使用 **modprobe** 启用您想要使用的调节器。例如：如果 **ondemand** 调节器不可用于您的 CPU，请使用以下命令：

```
modprobe cpufreq_ondemand
```

2. 将调节器列入您 CPU 的可用列表后，您可使用以下命令启用它：

```
echo [governor] >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

3.2.3. 调整 CPUfreq 策略和速度

您选择适当的 CPUfreq 调节器后，您可以使用在 **/sys/devices/system/cpu/*[cpu ID]*/cpufreq/** 中找到的可调按钮做对每个 CPU 的速度做进一步的调节。这些可调按钮有：

- **cpuinfo_min_freq** — 显示 CPU 的最小可用操作频率（单位千赫）
- **cpuinfo_max_freq** — 显示 CPU 的最大可用操作频率（单位千赫）
- **scaling_driver** — 显示在这个 CPU 中用来设定频率的 CPUfreq 驱动程序。
- **scaling_available_governors** — 显示这个内核中可用的 CPUfreq 调节器。如果您要使用不在此文件列表中的 CPUfreq 调节器，请参考 第 3.2.2 节 “CPUfreq 设置” 中的 过程 3.2, “启用 CPUfreq 调节器” 操作说明。
- **scaling_governor** — 显示当前使用的 CPUfreq 调节器。要使用不同的调节器，只要使用 **echo *[governor]* > /sys/devices/system/cpu/*[cpu ID]*/cpufreq/scaling_governor** 命令即可（详情请参考 第 3.2.2 节 “CPUfreq 设置” 中的 过程 3.2, “启用 CPUfreq 调节器”）。
- **cpuinfo_cur_freq** — 显示 CPU 的当前速度（单位千赫）。

- **scaling_available_frequencies** — 列出 CPU 可用频率，单位千赫。
- **scaling_min_freq** 和 **scaling_max_freq** — 设定 CPU *策略限制*，单位千赫。

重要

当设定策略限制时，您应该在 **scaling_min_freq** 之前设定 **scaling_max_freq**。

- **affected_cpus** — 列出需要频率协调软件的 CPU。
- **scaling_setspeed** — 用于更改 CPU 时钟速度，单位千赫。您只能在 CPU 策略限制中设定一个速度（根据 **scaling_min_freq** 和 **scaling_max_freq**）。

要查看每个可调整部分的当前值，请使用 **cat *[tunable]***。例如：要查看 cpu0 的当前速度（单位千赫），请使用：

cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq.

要更改每个可调整部分的值，请使用 **echo *[value]* > /sys/devices/system/cpu/*[cpu ID]*/cpufreq/*[tunable]***。例如：将 cpu0 的最小时钟速度设定为 360 千赫，请使用：

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

3.3. 挂起和恢复

当系统挂起时，内核调用驱动程序保存其状态然后卸载它们。当系统恢复时，它会载入这些试图重新编程其设备的驱动程序。驱动程序完成这个任务的能力决定了系统是否可以被成功恢复。

就这一点而言视频驱动程序是最成问题的，因为*高级配置和电源界面*（ACPI）规范不要求系统固件可以重新编程视频硬件。因此，除非视频驱动程序可以从完全非启动状态编程硬件，否则它们可以阻止系统恢复。

红帽企业版 Linux 6 包括对新图形芯片组的更大支持，这个支持可确定挂起及恢复在大量平台中正常工作。特别是对 NVIDIA 芯片组的大量改进，特别是对 GeForce 8800 系列的改进。

3.4. 无空循环内核

之前，Linux 内核在预定频率时周期性断系统中的每个 CPU - 100 Hz，250 Hz 或者 1000 Hz，具体要根据平台而定。内核查询 CPU 有关执行的进程，并使用结果进行进程计数和负载平衡。也就是我们知道的*计时器刻度*，内核执行这些中断时不考虑 CPU 的电源状态。因此，即使闲置的 CPU 也要每秒相应最多 1000 次的这种请求。在为闲置 CPU 使用节能方法的系统中，计时器刻度可妨碍 CPU 让系统保持足够长的闲置状态以达到从节能中获益的目的。

红帽企业版 Linux 6 内核运行*无空循环*：即它使用按需中断替换老的周期性计时器中断。因此可允许闲置 CPU 处于闲置状态直到请求执行新任务为止，且进入低功率状态的 CPU 可在此状态保持较长的时间。

3.5. 主动式电源管理

主动式电源管理（ASPM）在 *PCI Express 或者 PCIe（Peripheral Component Interconnect Express）*子系统中的节电，其原理为当设备连接的 PCI 连接没有处于使用状态时将其设定为低功率状态。ASPM 可同时在终端和连接中控制电源状态，并在连接终端的设备处于满电状态时仍可在连接中节电。

当启用 ASPM 时会增大设备延迟，因为在不同电源状态间转换连接时需要时间。ASPM 有三个决定电源状态的策略：

默认

根据系统（例如：BIOS）中固件指定的默认设置设定 PCIe 连接电源状态。这是 ASPM 的默认状态。

节电

将 ASPM 设定为在任何可能的情况下节电，不考虑性能损失。

性能

禁用 ASPM 以便允许 PCI 链接以最佳性能操作。

ASPM 策略是在 **/sys/module/pcie_aspm/parameters/policy** 中设定，但也可在引导时使用 ***pcie_aspm*** 内核参数指定，其中 **pcie_aspm=off** 禁用 ASPM，**pcie_aspm=force** 启用 ASPM，即使在不支持 ASPM 的设备中也可以。

警告

如果设定了 **pcie_aspm=force**，不支持 ASPM 的硬件可导致系统停止响应。请在设定 **pcie_aspm=force** 前确定系统中的所有 PCI 都支持 ASPM。

3.6. 主动连接电源管理

主动连接电源管理（ALPM）是一项节能技术，其原理为通过在闲置时（就是没有 I/O 的时候）将到磁盘的 SATA 连接设定为低功率模式帮助磁盘节电。在那个连接中有 I/O 请求队列后，ALPM 会自动将 SATA 连接恢复为活跃电源状态。

ALPM 使用的节电技术是以磁盘延迟为代价的。因此，您应该只在系统可能会长时间处于闲置 I/O 状态时使用 ALPM。

ALPM 只可用于使用*高级主机控制器接口*（AHCI）的 SATA 控制器。有关 AHCI 详情请参考 http://www.intel.com/technology/serialata/ahci.htm。

可用时，默认启用 ALPM。ALPM 有三种模式：

min_power

这个模式将连接设定为其最低功率状态（SLUMBER），此时磁盘中没有任何 I/O。这个模式在可能会延长闲置周期时有用。

medium_power

这个模式将连接设定为第二低功率状态（PARTIAL），此时磁盘中没有 I/O。这个模式是设计用来允许在尽量不影响性能的条件下在电源状态间进行转换（例如：在中等 I/O 负载和闲置 I/O 时）。

medium_power 模式允许根据负载情况在 PARTIAL 和满电（即 "ACTIVE"）状态间进行连接转换。请注意不可能直接从 PARTIAL 转换到 SLUMBER 然后再转回来。在这里，这两种电源状态都需要首先转换成 ACTIVE 状态，然后方可转换到另一个状态。

max_performance

ALPM 被禁用。当磁盘中没有 I/O 时连接不会处于任何低功率状态。

要查看您的 SATA 主机适配器是否真正支持 ALPM，您可以查看是否有文件 **/sys/class/scsi_host/host*/link_power_management_policy**。要更改设置，只要在这些文件中写入本章节中描述的值或者显示文件检查当前设置即可。

重要

将 ALPM 设定为 **min_power** 或者 **medium_power** 将自动禁用"热插拔"特性。

3.7. RELATIME 驱动器访问优化

POSIX 标准要求操作系统维护记录每个文件最后一次被访问的文件系统元数据。这个时间戳被称为 **atime**，维护它需要一个重复的对存储的写入操作。这些写入操作让存储是设备及其连接保持忙碌和通电状态。因为很少应用程序会使用 **atime** 数据，所以这个存储设备活动是在浪费电力。特别是即使没有从存储中读取该文件也会发生写入存储的事件，但是从缓冲中写入。有时，Linux 内核还支持 **mount** 的 **noatime** 选项，并不在使用此选项挂载的文件系统中写入 **atime**。但是只是关闭这个特性是有问题的，因为有些应用程序会依赖 **atime** 数据，并在此数据不可用时失败。

红帽企业版 Linux 6 使用的内核之后此另一个可替换选项 - **relatime**。**Relatime** 维护 **atime** 数据，但不是每次访问该文件时都更改。启用这个选项，则只在上次更新 **atime**（**mtime**）后修改该文件时，或者最后一次访问该文件是在相当长一段时间前（默认为一天）时才会将 **atime** 数据写入磁盘。

默认情况下，所有现在挂载的文件系统都启用 **relatime**。要在整个系统中限制这个特性，请使用 boot 参数 ***default_relatime=0***。如果默认在某个系统中启用 **relatime**，您可以通过使用选项 **norelatime** 挂载某个系统来限制它在某个具体文件系统中的使用。最后，要使系统更新文件的 **atime** 数据的默认周期有所不同，请使用 ***relatime_interval=*** 引导参数，以秒为单位指定周期。默认值为 **86400**。

3.8. 功率封顶

红帽企业版 Linux 6 支持最近在硬件中使用的功率封顶，比如 HP 的*动态功率封顶*（DPC）以及 Intel 的节点管理器（NM）技术。功率封顶允许管理员使用服务器限制功率消耗，但它还可允许管理器更有效地规划数据中心，因为极大降低了现有电源供应的超载风险。管理器可在同一物理印迹中放置更多的服务器并确定如果服务器电源消耗封顶，在高负载时对电源的需求会超出可用的电源。

HP 动态功率封顶

动态功率封顶是一个在选择 ProLiant 和刀片系统服务器时的特性，它可允许系统管理员对一个服务器或者一组服务器的电源消耗封顶。这个封顶是一个绝对限制，无论其当前工作负载如何，服务器将无法超过该限制。这个封顶只在服务器达到其电源消耗限制时才起效。此时某个管理进程会调整 CPU P 状态和始终刻度来限制电力消耗。

动态功率封顶会修改独立操作系统的 CPU 行为，但是 HP 的*集成的 Lights-Out 2*（iLO2）固件允许操作系统访问管理处理器，因此用户空间中的应用程序可查询管理处理器。红帽企业版 Linux 6 中使用的内核包括用于 HP iLO 和 iLO2 固件的驱动程序，它们可允许程序查询 **/dev/hpilo/dXccb*N*** 中的管理处理器。该内核还包括 **hwmon sysfs** 接口扩展来支持功率封顶特性，以及用于 ACPI 4.0 使用 **sysfs** 接口电源米表的驱动程序。这些特性允许操作系统和用户空间工具共同读取为功率封顶配置的值以及系统的当前电源用量。

有关 HP 动态功率封顶详情请参考*《HP 功率封顶以及用于 ProLiant 服务器的 HP 动态功率封顶》*，地址为：http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01549455/c01549455.pdf

Intel 节点管理器

Intel 节点管理器在系统中强制使用功率封顶，使用处理器 P 状态和 T 状态限制 CPU 性能，并因此节能。通过设置电源管理策略，管理员可将系统配置为在系统负载低时，比如夜间或者周末，消耗较少的电力。

Intel 节点管理器使用*直接操作系统配置和电源管理*（OSPM）通过标准*高级配置和电源接口*调整 CPU 性能。当 Intel 节点管理器通知 OSPM 驱动程序更改到 T 状态时，该驱动程序会响应并更改处理器 P 状态。同样，当 Intel 节点管理器通知 OSPM 驱动程序更改到 P 状态时，该驱动程序也会相应更改 T 状态。这些更改自动进行且不需要操作系统有进一步的输入。管理员使用*Intel 数据中心管理器*（DCM）软件配置并监控 Intel 节点管理器。

有关 Intel 节点管理器详情请参考*《节点管理器 - 动态管理数据中心电源》*，地址为：http://communities.intel.com/docs/DOC-4766

3.9. 改进的图形电源管理

红帽企业版 Linux 6 通过删除不必要的资源消耗在图形和显示设备中节能。

LVDS 重新计时

低压差分信号传输（Low-voltage differential signalling，LVDS）是使用铜线承载电信号的系统。一个主要的应用是将像素信息传输到笔记本电脑的*液晶显示*（LCD）屏幕中。所有显示都有*刷新率* - 即它们从图形控制器接受新鲜数据并在屏幕中重新成像的频率。通常屏幕每秒接受 60 次新鲜数据（即频率为 60 Hz）。当屏幕和图形控制器是以 LVDS 连接时，LVDS 系统在每次刷新时都会消耗能量。当闲置时，很多 LCD 屏幕的刷新率都会下降到 30 Hz，且不会产生明显的影响（与*阴极射线管*（CRT）显示器不同，后者在降低刷新率时会产生闪烁现象）。红帽企业版 Linux 6 内核使用的 Intel 图形适配器的驱动程序可自动执行这个*降频*，并在屏幕闲置时节约 0.5 W 左右的电力。

启用内存自动刷新

同步动态随机访问内存（SDRAM） - 由于用于图形适配器的视频内存，因此每秒会重复充电上千次，以便每个内存单元可保留保存在其中的数据。除了管理数据的主要功能外，因为有数据流入或者流出内存，所以内存控制器通常负责初始化这些刷新循环。但是 SDRAM 还有一个低功率*自动刷新*模式。在这个模式中，内存使用内部计时器生成其自身刷新循环，它可允许系统在不损害当前内存数据的情况下关闭内存控制器。红帽企业版 Linux 6 使用的内核可在 Intel 图形适配器处于闲置状态时触发内存自动刷新，并可节约 0.8 W 左右的电力。

GPU 时钟修订

典型图形处理单元（GPU）包含管理其内部电路不同部分的内部时钟。红帽企业版 Linux 6 使用的内核可降低部分 Intel 和 ATI GPU 的内部时钟频率。减少 GPU 组件在给定时间内执行循环的次数可减少其在那些它们不一定要执行的循环中消耗的能量。当 GPU 闲置时，内核可自动降低这些时钟的速度；同时当 GPU 活性增强时会提高其时钟速度。降低 GPU 时钟循环最多可节省 5 W 电力。

GPU 关闭

红帽企业版 Linux 6 中使用的 Intel 和 ATI 图形驱动程序可探测到什么时候适配器中没有连接显示器，并完全关闭 GPU。这个功能对不经常连接显示器的服务器尤为重要。

3.10. RFKILL

很多计算机系统包含无线电传输，其中包括 Wi-Fi、蓝牙和 3G 设备。这些设备消耗电源，在不使用这些设备时是一种浪费。

RFKill 是 Linux 内核中的一个子系统，它可提供一个界面，在此界面中可查询、激活并取消激活计算机系统中的无线电传输。当取消激活传输时，可使其处于可被软件重新激活的状态（*软锁定*）或者将其放在软件无法重新激活的位置（*硬锁定*）。

RFKill 核为子系统提供应用程序编程界面（API）。内核驱动程序被设计为支持 RFKill 使用这个 API 注册内核，并包含启用和禁用这个设备的方法。另外，RFKill 核提供用户程序可解读的通知以及用户程序查询传输状态的方法。

RFKill 界面位于 **/dev/rfkill**，其中包含系统中所有无线电传输的当前状态。每个设备都在 **sysfs** 中注册当前 RFKill 状态。另外，在启用了 RFKill 的设备中每当状态更改时，RFKill 会发出 *uevents*。

Rfkill 是一个命令行工具，您可使用它查询和更改系统中启用了 RFKill 的设备。要获得这个工具，请安装 rfkill 软件包。

使用命令 **rfkill list** 获得设备列表，每个都包含与之关联的*索引号*，从 **0** 开始。您可以使用这个索引号让 **rfkill** 停止使用或者使用某个设备，例如：

```
rfkill block 0
```

停用系统中第一个启用 RFKill 的设备。

您还可以使用 **rfkill** 阻断某一类设备，或者所有启用了 RFKill 的设备。例如：

```
rfkill block wifi
```

停用系统中的所有 Wi-Fi 设备。要停用所有启用了 RFKill 的设备，请运行：

```
rfkill block all
```

要重新使用设备，请运行 **rfkill unblock**，而不是 **rfkill block**。要获得 **rfkill** 可停用的完整设备类别列表，请运行 **rfkill help**。

3.11. 在用户空间中优化

减少系统硬件的工作量是节能的关键。因此虽然 第 3 章 *核心基础结构及技巧* 中描述的更改允许系统在减少电源消耗的各种状态下操作，而用户空间中要求系统硬件进行不必要工作的程序会阻止硬件进入这种状态。在红帽企业版 Linux 6 开发过程中，在以下区域执行审核来降低硬件的不必要需求：

减少唤醒

红帽企业版 Linux 6 使用*无空循环内核*（请参考 第 3.4 节 “无空循环内核”），它允许 CPU 保持深度闲置状态的时间更长一些。但是*计时器刻度*不是过度 CPU 唤醒的唯一原因，来自程序的功能调用还可阻止 CPU 进入或者保持闲置状态。可在 50 多个程序中减少不必要的功能调用。

减少存储和网络 IO

输入或者输出（IO）到存储设备和网络接口会强制消耗电源。在具有处于闲置状态时减少电源消耗功能（比如 ALPM 或者 ASPM）的存储和网络设备中，这个流量可阻止该设备进入或者保持闲置状态，并可阻止硬盘在不使用时转速降低。已经在有些程序中最小化了过度或者不必要的需求。特别是那些阻止硬盘降低转速的需求。

初始化脚本审核

无论是否需要都自动启动的服务会在很大程度上浪费系统资源。系统应尽量将其设定为默认 "off" 或者 "on demand"。例如：无论是否有蓝牙设备，以前在系统启动时，启用蓝牙服务的 **BlueZ** 会自动运行。**BlueZ** 启动脚本现在在启动该服务前检查系统中是否有蓝牙设备。

第 4 章 使用案例

本章描述了两类使用案例演示本指南中描述的分析和配置方法。第一个示例说的是典型服务器，第二个是典型笔记本电脑。

4.1. 示例 - 服务器

现在典型的标准服务器基本都包含红帽企业版 Linux 6 中支持的所有所需硬件功能。您的首要考虑是该服务器主要使用的负载类型。根据这个信息您可以决定要优化哪些组件节能。

不考虑服务器类型，通常不需要图像性能。因此可打开 GPU 节能。

网页服务器

网页服务器需要网络和磁盘 I/O。根据外部连接速度，100 Mbit/s 应该足够了。如果该机器大多数提供的是静态页面，CPU 性能则并不重要。因此电源管理选项应包括：

- 无 **tuned** 的磁盘或者网络插件。
- 打开 ALPM。
- 打开**按需**调节器。
- 网卡限制为 100 Mbit/s。

计算服务器

计算服务器主要是 CPU。电源管理选择可能包括：

- 根据任务以及出现数据存储的位置，激活 **tuned** 的磁盘或者网络插件；或者在批处理系统中完全激活 **tuned**。
- 根据应用，可能是**性能**调节器。

邮件服务器

邮件服务器主要需要磁盘 I/O 和 CPU。电源管理选择应包括：

- 打开**按需**调节器，因为 CPU 最后的几个百分比并不重要。
- 无 **tuned** 的磁盘或者网络插件。
- 不应该限制网络速度，因为邮件通常是内部的，并可因此从 1 Gbit/s 或者 10 Gbit/s 连接中获益。

文件服务器

文件服务器的需要与邮件服务器类似，但根据所用协议，可能需要更多的 CPU 性能。通常，基于 Saba 的服务器需要的 CPU 比 NFS 多，而 NFS 又比 iSCSI 需要更多的 CPU。即使如此，您应可以使用**按需**调节器。

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 的情况下。网络延迟最重要，而网络 I/O 次之。您可以考虑使用较低连接速度的延迟网络调节，但您应为具体网络进行细心的测试。

4.2. 示例 - 笔记本电脑

另一个电源管理和节能通常起作用的示例就是笔记本电脑。因为笔记本电脑一般已经被设计为比工作站或者服务器节省很多电力。当使用电池模式时，节能可让您的电池使用时间多几分钟。虽然这部分着重阐述笔记本电脑的电池模式，但您仍然可在使用交流电供电时也使用一些或者所有调整。

在笔记本电脑中单一组件中的节能通常要比在工作站中更明显。例如：1 Gbit/s 网络接口以 100 Mbits/s 运行时可节电大约 3-4 瓦。对于总耗电 400 瓦的典型服务器来说，这个节能大概是在 1 %。在总耗电 40 瓦的典型笔记本电脑中，这一个组件的节能就是总耗电量的 10 %。

典型笔记本电脑中的具体节能优化包括：

- 将系统 BIOS 配置为禁用所有您不使用的硬件。例如：并口或者串口、读卡器、摄像头、WiFi 以及蓝牙，这里只给出一些可能的硬件。
- 在较暗的环境中，您不需要使用最大亮度就可舒服地阅读屏幕中的内容，此时可调暗显示。请在 GNOME 桌面中使用**「系统」**+**「首选项」** → **「电源管理」**，在 KDE 桌面中使用 **「Kickoff Application Launcher」**+**「计算机」**+**「系统设置」**+**「高级」** → **「电源管理」**，或者在命令行中使用 **gnome-power-manager** 或者 **xbacklight**，或者您笔记本电脑中的功能键。
- 使用 **tuned-adm** 的 **`laptop-battery-powersave`** 侧写启用整个节能机制。请注意会影响硬盘和网络接口的性能和延迟。

另外您可在各种系统设置中执行很多小的调整：

- 请使用**按需**调节器（红帽企业版 Linux 6 中默认启用）
- 启用笔记本电脑模式（**`laptop-battery-powersave`** 侧写的一部分）：

```
echo 5 > /proc/sys/vm/laptop_mode
```

- 增加磁盘冲洗时间（**`laptop-battery-powersave`** 侧写的一部分）：

```
echo 1500 > /proc/sys/vm/dirty_writeback_centisecs
```

- 禁用 nmi 监视器（**`laptop-battery-powersave`** 侧写的一部分）：

```
echo 0 > /proc/sys/kernel/nmi_watchdog
```

- 启用 AC97 音频节能（在红帽企业版 Linux 6 中默认启用）：

```
echo Y > /sys/module/snd_ac97_codec/parameters/power_save
```

- 启用多核节能（**`laptop-battery-powersave`** 侧写的一部分）：

```
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 自动挂起并不适用于所有 USB 设备。

- 为 ALPM 启用最小能量设置（**`laptop-battery-powersave`** 侧写的一部分）：

```
echo min_power >
/sys/class/scsi_host/host*/link_power_management_policy
```

- 使用 relatime 挂载文件系统（在红帽企业版 Linux 6 中默认使用）：

```
mount -o remount,relatime mountpoint
```

- 为硬盘激活最佳节能模式（**laptop-battery-powersave** 侧写的一部分）：

```
hdparm -B 1 -S 200 /dev/sd*
```

- 禁用 CD-ROM 调用（**laptop-battery-powersave** 侧写的一部分）：

```
hal-disable-polling --device /dev/scd*
```

- 将屏幕亮度降低至 **50** 或更小，例如：

```
xbacklight -set 50
```

- 为屏幕闲置激活 DPMS：

```
xset +dpms; xset dpms 0 0 300
```

- 降低 Wi-Fi 能源等级（**laptop-battery-powersave** 侧写的一部分）：

```
for i in /sys/bus/pci/devices/*/power_level ; do echo 5 > $i ; done
```

- 取消激活 Wi-Fi：

```
echo 1 > /sys/bus/pci/devices/*/rf_kill
```

- 将有线网络限制为 100 Mbit/s（**laptop-battery-powersave** 侧写的一部分）：

```
ethtool -s eth0 advertise 0x0F
```

附录 A. 开发者小贴示

每本优秀的编程课本都包含内存分配以及具体功能性能的问题。当您开发自己的软件时，请注意可能在运行该软件的系统中增加电源消耗的问题。虽然这些考虑不会影响每一行代码，但您可以优化那些经常成为性能瓶颈部分的代码。

经常会出问题的技术包括：

- 使用线程。
- 不必要 CPU 唤醒以及未有效使用唤醒。如果您必须执行唤醒，尽快一次做完所有的事（迅速返回闲置状态）。
- 不必要的 **[f]sync()** 使用。
- 不必要的活跃调用或者使用简短常规超时（使用响应事件）。
- 没有有效使用唤醒。
- 低效磁盘访问。使用大量缓冲来避免频繁的磁盘访问。一次写入大块信息。
- 低效使用计时器。可能时使用跨应用程序（甚至跨系统）的组群计时器。
- 过量的 I/O，电源消耗或者内存使用（包括内存泄露）
- 执行不必要计算。

下面的部分更详细地阐述这些方面。

A.1. 使用线程

普遍认为使用线程使应用程序可更好且更迅速地执行，但并不总是这样。

Python

Python 使用全局锁定解码器[1]，因此使用线程只能在有大量 I/O 操作时受益。**Unladen-swallow** [2] 是一个 Python 快速部署，您可用它来优化您的代码。

Perl

Perl 线程原是用于系统中不使用 forking 技术的应用程序（比如使用 32 位 Windows 操作系统的系统）。在 Perl 线程中会为每个单一线程复制数据（写时复制）。数据不是默认共享的，因为用户应该可以定义数据共享等级。必须包括共享 **threads::shared** 模块的数据。但是数据不仅仅是被复制（写时复制），该模块还为这些数据生成了捆绑变量，这就需要更多的时间，且速度更慢。[3]

C

C 线程共享同一内存，每个线程都有自己的栈，同时内核不一定要生成新的文件描述符并分配新的内存空间。C 可以真正在更多线程中使用更多 CPU 支持。因此要最大化您的线程性能，请使用低级语言，比如 C 或者 C++。如果您使用脚本语言，请考虑写入一个 C 绑定。请使用分析器识别不能很好执行的代码。[4]

A.2. 唤醒

很多应用程序都会扫描配置文件中的更改。在很多情况下，这种扫描的时间间隔是固定的，例如：每分钟。这可能是个问题，因为它强制将磁盘从低转速状态唤醒。最佳解决方案是找到合理的时间间隔，好的检查机制或者使用 **inotify** 检查并响应每个事件。**Inotify** 可查看文件或者目录中的各种更改。

例如：

```
int fd;
fd = inotify_init();
int wd;
/* checking modification of a file - writing into */
wd = inotify_add_watch(fd, "./myConfig", IN_MODIFY);
if (wd < 0) {
  inotify_cant_be_used();
  switching_back_to_previous_checking();
}
...
fd_set rdfs;
struct timeval tv;
int retval;
FD_ZERO(&rdfs);
FD_SET(0, &rdfs);

tv.tv_sec = 5;
value = select(1, &rdfs, NULL, NULL, &tv);
if (value == -1)
  perror(select);
else {
  do_some_stuff();
}
...
```

这个方法的优点是您可执行不同的检查。

主要的局限是一个系统中的查看次数是有限的。次数可在 **/proc/sys/fs/inotify/max_user_watches** 中获得，虽然该数字是可以更改的，但并不建议如此操作。再有，**inotify** 失败时，该代码必须返回不同的检查方法，通常意味着在源代码中会有很多 **#if #define**。

有关 **inotify** 的详情请参考 inotify man page。

A.3. FSYNC

Fsync 被视为大量消耗 I/O 的操作，但这并不完全正确。例如：参见 Theodore Ts'o's 的文章*《不要害怕 fsync！》* [5] 以及附带的讨论。

Firefox 原来在用户每次点击一个链接时都调用 **sqlite** 程序库进入新的页面。**Sqlite** 调用 **fsync**，且由于文件系统设置（主要使用数据排序模式的 ext3），什么都不发生时会有一个长时间延迟。如果另一个进程同时正在复制一个大文件，这就需要很长的时间（最长可达 30 秒）。

但在另一个示例中根本不使用 **fsync**，则在切换到 ext4 文件系统时出了问题。Ext3 是被设定为数据排序模式，它会每几秒钟排空一次内存并将其保存到磁盘中。但 ext4 使用的是笔记本电脑模式，保存内存的时间间隔较长，且可能在系统意外关闭时丢失数据。现在 ext4 有一个补丁，但我们必须仍在设计应用程序时小心谨慎，正确使用 **fsync**。

下面读取和写入配置文件的简单示例演示了如何备份文件或者数据是怎么丢失的：

```
/* open and read configuration file e.g. ~/.kde/myconfig */
fd = open("./kde/myconfig", O_WRONLY|O_TRUNC|O_CREAT);
read(myconfig);
...
write(fd, bufferOfNewData, sizeof(bufferOfNewData));
close(fd);
```

更好的方法可能是：

```
open("/.kde/myconfig", O_WRONLY|O_TRUNC|O_CREAT);
read(myconfig);
...
fd = open("/.kde/myconfig.suffix", O_WRONLY|O_TRUNC|O_CREAT);
write(fd, bufferOfNewData, sizeof(bufferOfNewData));
fsync; /* paranoia - optional */
...
close(fd);
rename("/.kde/myconfig", "/.kde/myconfig~"); /* paranoia - optional */
rename("/.kde/myconfig.suffix", "/.kde/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

[5] http://thunk.org/tytso/blog/2009/03/15/dont-fear-the-fsync/

附录 B. 修订记录

修订 1.0-5.400	**2013-10-31**	**Rüdiger Landmann**
Rebuild with publican 4.0.0		
修订 1.0-5	**2012-07-18**	**Anthony Towns**
Rebuild for Publican 3.0		
修订 1.0-1	**Thu Oct 7 2010**	**Rüdiger Landmann**
删除 "draft" 标签		
修订 1.0-0	**Thu Oct 7 2010**	**Rüdiger Landmann**
GA 发行本		