Red Hat Enterprise Linux（RHEL） 6

電源管理指南

管理 Red Hat Enterprise Linux 6 的電源消耗量

Last Updated: 2017-10-12

Red Hat Enterprise Linux（RHEL） 6 電源管理指南

管理 Red Hat Enterprise Linux 6 的電源消耗量

Don Domingo
Red Hat 工程部出版中心

Rüdiger Landmann
Red Hat 工程部出版中心

Jack Reed
Red Hat 工程部出版中心
jreed@redhat.com

Red Hat Inc.

法律聲明

摘要

本文件解釋如何有效管理 Red Hat Enterprise Linux 6 的電源消耗量。以下章節將討論降低電源消耗量的多種技巧（適用於伺服器與筆記型電腦），以及這些技巧會如何影響系統的整體效能。

內容目錄

章 1. 總覽

電源管理一直是 Red Hat Enterprise Linux 6 的專注項目之一。從宏觀的觀點來看，電源管理只是「*綠 IT*」的一部分；綠 IT 的整體層面更廣，還包括使用可回收的材質、使用不污染環境的生產系統、以及正確的計畫與規劃等等。在本文件裡，我們提供了 Red Hat Enterprise Linux 6 關於電源管理的指引與資訊。

1.1. 電源管理的重要性

電源管理的核心，是了解如何針對每個系統元件，進行有效的能源最佳化。這需要研究系統所進行的不同任務，並配置每個元件以確保其效能適用於這些工作。

採用電源管理的主要動機有：

- 降低整體的電力消耗量，以節省成本

正確使用電源管理的主要好處有：

- 降低伺服器與運算中心的熱量
- 降低次要成本，包括冷卻系統、空間、機房、發電機、UPS（*不斷電系統*，*uninterruptible power supplies*）等
- 延長筆記型電腦的電池壽命
- 減低碳排放量
- 符合綠 IT 的政府法規與法律需求，例如能源之星（Energy Star）標準
- 符合公司對於新系統的章程

降低某個特定元件（或是整個系統）的電力消耗量能降低熱能，當然這也會降低效能：這是鐵則。因此，使用者應該要徹底研究、測試任何配置，看看會降低多少效能，尤其是在每日運作的重運量系統上。

藉由研究系統上運行的工作，並針對這些工作配置每個元件，使其運作無礙，這樣就可以節省能源、降低產生的熱能、並延長筆記型電腦的電池使用時間。許多針對電源的分析與微調原則，都與針對效能的原則類似。就某些程度來講，電源管理與效能微調是相左的，因為系統最佳化的兩端分別是效能與電力。本手冊描述了 Red Hat 所提供的工具，以及我們所發展出來的技巧，好讓使用者應用在電源管理上。

Red Hat Enterprise Linux 6 已經包括了許多新的電源管理功能，而且預設上是啟用的。這些特選的功能都不會對一般的伺服器或桌上型電腦，產生效能上的影響。然而，針對最大效能產出、最低延遲時間、或最高 CPU 效能等不同情況，使用者需要檢視這些預設值是否合用。

要決定是否應該採用這些技巧，對電腦進行最佳化，請回答以下幾個問題：

問： 我一定要進行最佳化嗎？

答： 電源最佳化的重要性，端視公司是否有電源指示必須遵守，還是需要遵循某些法規。

問： 我要最佳化到什麼程度？

答： 我們在此呈現的一些技巧並不需要您仔細地稽核、分析系統；而是提供一般性的最佳化方式，藉以改變電力使用方式。預設的方式當然比不上手動稽核、最佳化的系統，但達到了很好的平衡。

問： 最佳化會不會把系統效能降低到令人無法忍受的程度？

答： 本文件所描述的技巧，都會對系統效能產生顯著的衝擊。如果您想要對 Red Hat Enterprise Linux 6 採用預設值以外的電源管理模式，請在電源最佳化之後監控系統效能，看看效能的降低是否在可接受的範圍內。

問： 與花費的時間、精力比起來，最佳化後的結果是否划算？

答： 遵循整個過程來手動最佳化一台系統，就時間與成本來說並不划算，因為這跟單一系統的生命週期不成比例。但另一方面，如果您使用同樣的配置與設定，一次建置一萬台桌上型電腦，那麼建立最佳化的設定，並一次套用到所有電腦上，那聽起來是個很好的主意。

以下章節會解釋最佳化後的硬體效能，會如何減少電力消耗。

1.2. 電源管理的基本概念

有效管理電源狀態建立於下列原則之上：

只有在需要的時候，才喚醒閒置中的 CPU

Red Hat Enterprise Linux 5 的 kernel 對每個 CPU 使用一個定期的計時器。這個計時器會避免處理器真的進入閒置狀態，因為不管 CPU 是不是正在執行程序，計時器都會要求 CPU 處理每個計時器的事件（每隔幾毫秒就會發生一次，視設定而定）。要有效管理電源，極大部分是與降低喚醒 CPU 的頻率有關。

有鑑於此，Red Hat Enterprise Linux 6 的 Linux kernel 會降低這個計時器的頻率：因此，CPU 的閒置狀態會變成 *tickless*（無計時）。這會避免 CPU 在閒置時消耗不需要的電源。然而，如果系統中有應用程式會產生不必要的計時事件，那麼這功能就毫無用武之地。輪詢事件（例如檢查磁碟卷冊的變動、滑鼠動作等等）都是很好的例子。

Red Hat Enterprise Linux 6 提供了一些工具，讓您根據 CPU 的使用量來分辨、稽核應用程式。詳情請參閱 章 2, *電源管理的稽核與分析*。

完全停用不使用的硬體與裝置

這對擁有移動零件的裝置（例如硬碟）尤其重要。除此之外，有些應用程式可能會讓不使用、但仍啟用的裝置處於開啟（open）狀態；當這情況發生時，kernel 會假定這裝置還在使用中，讓裝置無法進入省電模式。

低活動等於低瓦數

然而在許多情況下，這端賴較新的硬體與正確的 BIOS 配置。較舊的系統元件通常不支援 Red Hat Enterprise Linux 6 提供的一些新功能。請確定您電腦使用的是最新的官方韌體，同時 BIOS 裡的電源管理或裝置配置都已經啟用。要注意的功能包括：

- SpeedStep
- PowerNow!
- Cool'n'Quiet
- ACPI (C state)
- Smart

如果您的硬體支援這些功能，而且在 BIOS 裡面也已經啟用，那麼預設上 Red Hat Enterprise Linux 6 會啟用這些功能。

CPU 狀態的種類，及其效用

現代 CPU 都支援 ACPI（*Advanced Configuration and Power Interface*，進階配置與電源介面）功能，提供多種電源狀態。這三種狀態為：

- 睡眠（C-state）
- 頻率（P-state）
- 熱輸出（T-state 或稱「thermal state」：熱能狀態）

在最低睡眠狀態下執行的 CPU 會消耗最低瓦數，但要喚醒 CPU 的時間也最長。在極少數的情形下，這會讓 CPU 才剛陷入睡眠狀態，又被喚醒。這情形會導致 CPU 永遠處於忙碌狀態，這樣就不會節省電源。

電腦關機之後，電力消耗最低。

很明顯的，最省電的方式是完全關閉電腦的電源。舉例來說，您的公司可以發展出「綠 IT」的企業文化，讓員工在外出午餐或回家前，將電腦關閉。您也可以使用 Red Hat Enterprise Linux 6 的虛擬化技術，把數台伺服器整合至一台較大的伺服器。

章 2. 電源管理的稽核與分析

2.1. 稽核與分析概要

在單系統上詳盡地手動稽核、分析與微調，通常是良好電源管理的例外情況，因為所花費的時間與成本會多過系統微調所帶來的好處。然而，在許許多多規格近似的系統上，為所有系統使用同樣設定就非常有用。例如一次建置成千上萬台桌上系統，或一組擁有同樣成員的 HPC 叢集。進行稽核與分析的另一個理由，是提供比較用的基準，藉以辨別系統將來的變化或衰退。如果硬體、BIOS、或軟體都會定期更新，又如電力消耗異常的話，這分析結果會非常有用。通常完整的稽核與分析可以讓使用者更了解系統發生了什麼事。

稽核、分析系統的電力消耗情形是個困難的課題，即使在最先進的系統上亦然。大部分系統並不透過軟體，提供衡量電力消耗的方法。不過還是有例外：HP 伺服器系統的 ILO 管理主控台有電源管理模組，讓使用者透過網頁介面來存取。IBM 也以類似方式提供了 BladeCenter 電源管理模組。在一些 Dell 的系統上，IT Assistant 也提供了電源監控能力。其它廠商可能也會為其伺服器產品提供類似的能力；由此可知，市面上沒有單一的解決方案，適用於所有產品。如果您的系統沒有內建衡量電源消耗的機制，還是有其它選擇。您可以在系統上安裝特別的電源供應器，它能透過 USB 連接埠獲得電源消耗資訊。此外，有些例如 Watts up? PRO 的外部電壓計也有 USB 接頭。

通常只有在盡可能節省最大電力時，才需要直接度量電力消耗量。幸運的是，還有其它方式也可以用來度量電力消耗，得知系統的改變會造成什麼樣的結果，或了解系統的行為模式。本章將描述一些必要的工具。

2.2. POWERTOP

Red Hat Enterprise Linux 6 開始使用的無計時 kernel（tickless kernel，請參閱 節 3.6, “無計時 kernel”）能讓 CPU 更常進入閒置狀態，降低電力消耗，增進電源管理。新的 **PowerTOP** 工具會辨識核心與使用者空間應用程式的特定元件，看看這些元件是否常常喚醒 CPU。之前 **PowerTOP** 是用於發展過程，以進行 節 3.13, “使用者空間的最佳化” 所述的稽核工作；現在許多應用程式在這版本中都已經微調過，不當喚醒 CPU 的機會大約只剩原來的十分之一。

Red Hat Enterprise Linux 6 提供了版本 2.x 的 **PowerTOP**。這個版本已將 1.x 的程式碼基底完全重新編寫。它包含了更清楚、基於分頁的使用者介面，並且廣泛使用了 kernel 的 **"perf"** 架構來提供更加精確的資料。系統裝置的電源行為會被追蹤且明顯顯示出，如此一來問題便能被快速找出。此外，2.x 的程式碼基底包含了一個較為試用性的電源估算引擎，它能顯示各項裝置與程序所耗用的電源。請參閱〈圖形 2.1, “PowerTOP 的操作畫面”〉。

若要安裝 **PowerTOP**，請以 **root** 身份執行下列指令：

```
yum install powertop
```

若要執行 **PowerTOP**，請以 **root** 身份執行下列指令：

```
powertop
```

PowerTOP 能提供系統的總電源使用量估算，並顯示各項程序、裝置、kernel 工作、計時器以及插斷處理程式的個別電源使用量。手提電腦在執行這項任務時應使用電池電源。若要校準電源估算引擎，請以 **root** 身份執行以下指令：

```
powertop --calibrate
```

微調需要花上一段時間。這項程序會執行數項測試，並且將會進行螢幕亮度的循環測試，並將裝置開啟和關閉。請勿在進行校準時操作機器。當校準程序完成時，**PowerTOP** 將會正常啟動。請讓它執行約一小時以蒐集資料。當蒐集了足夠資料後，電源估算表便會顯示在第一個欄位中。

若您要在一部手提電腦上執行這項指令，它還是應該使用電池電源，如此一來才能提供所有的可用資料。

它在執行時，**PowerTOP** 會由系統蒐集數據。您可在「**總覽**」分頁中檢視一列最常傳送喚醒訊號給 CPU 的元件，或是電源使用量最高的元件（請參閱 圖形 2.1, “PowerTOP 的操作畫面”）。鄰接的欄位顯示了電源估算、資源如何被使用、每秒的喚醒動作、元件的類（例如程序、裝置或是計時器），以及元件的詳述。每秒鐘的喚醒動作代表服務或 kernel 裝置與驅動程式的效率有多高。喚醒次數愈少，代表所耗用的電源量愈低。元件會以其能夠優化電源使用量的程度來排序。

若要微調驅動程式元件，您基本上需要對 kernel 進行變更，而這項任務不包含在本指南的範圍內。然而，會傳送喚醒訊號的使用者空間程序較容易管理。首先，請判斷這項服務或應用程式是否需要在這部系統上執行。若不需要的話，請將它停用。若要永久停用一項老舊的 System V 服務，請執行：

```
chkconfig off servicename off
```

欲取得更多有關於這項程序的相關資訊，請以 **root** 身份執行以下指令：

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

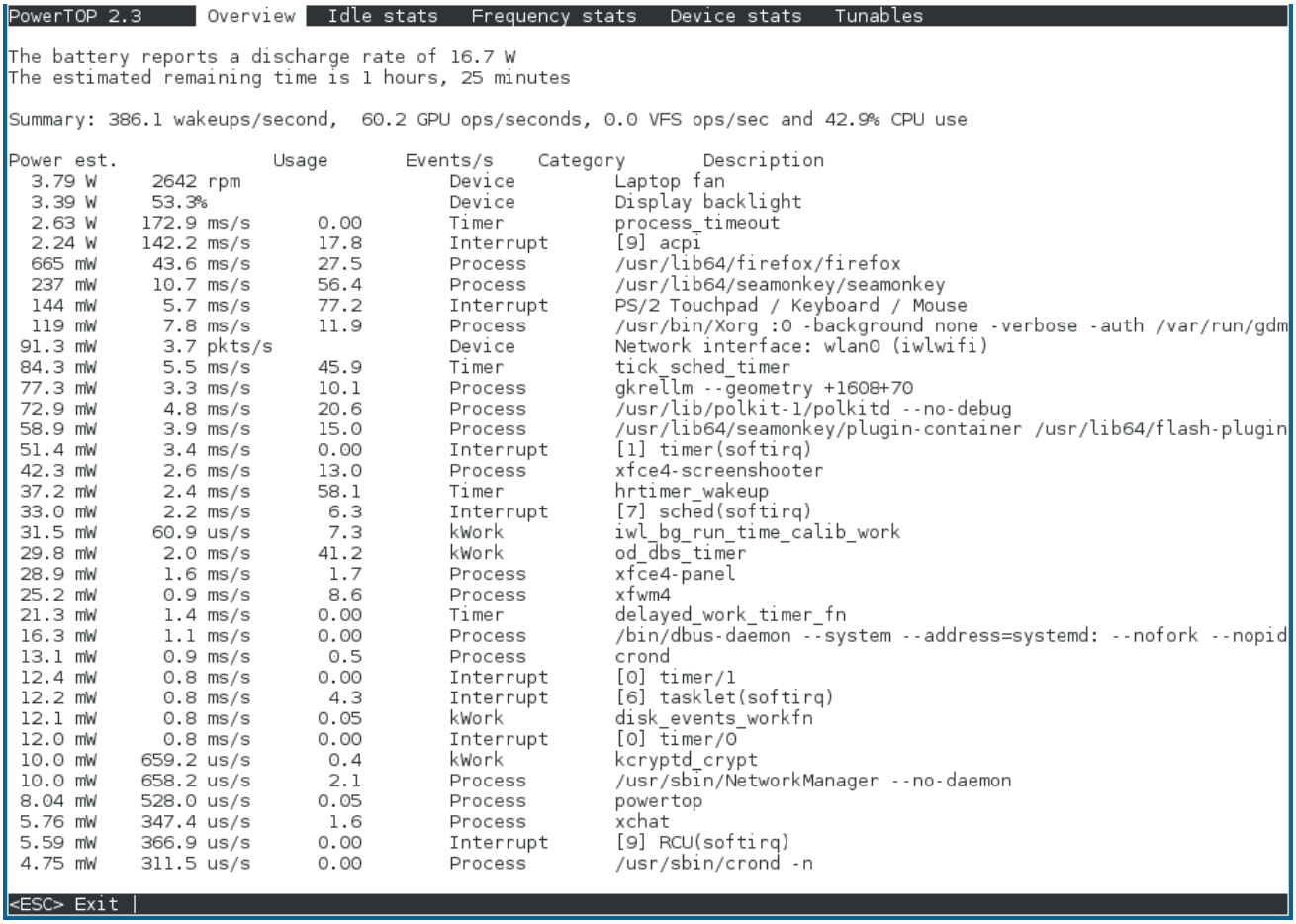
若追蹤紀錄顯示它正在重複執行，這代表它可能是個忙碌的迴圈。若要修正這種錯誤，一般需要修改元件中的程式碼。

如〈圖形 2.1, “PowerTOP 的操作畫面”〉中所見，總電源耗用量和電池剩餘電量（若存在）將會顯示。在這些資訊下面則是一項簡短的概要，它包含了每秒的總喚醒次數、每秒的 GPU 操作量，以及虛擬檔案系統每秒的操作量。在畫面剩下的部分中則包含了一列程序、插斷、裝置及其它資源，並根據其使用量來排序。若經過正確校準，列在第一個欄位中的每個項目的電源使用量估算也會顯示。

使用 **Tab** 與 **Shift+Tab** 鍵來循環瀏覽分頁。在「**閒置數據**」分頁中，所有處理器與核心的 C-state 使用將會被顯示。在「**頻率數據**」分頁中，所有處理器與核心的 P-state 使用，包含 Turbo 模式（若可用的話）皆會被顯示。CPU 留駐的時間愈久，C 或 P-state 就愈高並且愈佳（**C4** 比 **C3** 高）。這能顯示 CPU 使用量的優化程度如何。當系統閒置時，最高的 C 或 P-State 應要擁有超過 90% 的駐留時間。

「**裝置數據**」分頁提供了類似「**總覽**」分頁的資訊，不過此分頁僅提供了裝置相關的資訊。

「**可微調**」分頁包含了有關於優化系統以降低電源使用量的建議。請使用 **up** 和 **down** 箭頭鍵來瀏覽建議，以及 **enter** 鍵來將建議切換為開啟或關閉。



圖形 2.1. PowerTOP 的操作畫面

您亦可藉由執行 **PowerTOP** 並搭配 **--html** 選項來產生 HTML 報告。請將 *htmlfile.html* 參數替換為您希望使用的輸出檔案之名稱：

```
powertop --html=htmlfile.html
```

就預設值，**PowerTOP** 會以 20 秒的間隔來進行測量，您可藉由 **--time** 選項來更改此行為：

```
powertop --html=htmlfile.html --time=seconds
```

欲取得更多有關於 PowerTOP 專案的資訊，請參閱〈project's home page〉。

PowerTOP 也能與 **turbostat** 工具程式搭配使用。**turbostat** 工具程式乃一項回報工具，它能顯示 Intel 64 處理器上的處理器拓撲、速率、閒置電源狀態數據、溫度以及電源使用量上的相關資訊。欲取得更多有關於 **turbostat** 工具程式上的相關資訊，請參考 **turbostat**(8) man page，或參閱《效能微調指南》。

2.3. DISKDEVSTAT 與 NETDEVSTAT

Diskdevstat 與 **netdevstat** 都屬於 **SystemTap** 工具組，用來蒐集系統上所有應用程式存取磁碟與網路的詳盡資料。這些工具的靈感來自 **PowerTOP**，顯示了每個應用程式對 CPU 的喚醒次數（詳情請參閱節 2.2, “PowerTOP”）。這些工具所蒐集的統計數據能讓您找出最耗電的應用程式，其中以大量的小筆 I/O 運作為最，而不是少量大筆的運作。其它只能衡量傳輸速度的監控工具，並無法分辨這方面的用量。

以 **root** 身份透過下列指令來以 **SystemTap** 安裝這些工具：

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

請以此指令執行這些工具：

```
diskdevstat
```

或是：

```
netdevstat
```

兩個指令都可以接受三個參數，分別是：

diskdevstat *update_interval total_duration display_histogram*

netdevstat *update_interval total_duration display_histogram*

update_interval

每次更新畫面的時間間隔，單位為秒。預設值為 **5**

total_duration

整個執行時間，單位為秒。預設值為 **86400**（一天）

display_histogram

旗標值，表示在執行結束時，是不是要將所有資料以直方圖列出。

螢幕列出的資料來自 **PowerTOP**。以下是在 Fedora 10 搭配 KDE 4.2 的系統上，執行 **diskdevstat** 的結果：

```
  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
```

欄位為：

PID

應用程式的程序 ID

UID

執行應用程式的使用者 ID

DEV

發生 I/O 的裝置

WRITE_CNT

寫入的總數

WRITE_MIN

兩次連續寫入的最短時間（單位為秒）

WRITE_MAX

兩次連續寫入的最長時間（單位為秒）

WRITE_AVG

兩次連續寫入的平均時間（單位為秒）

READ_CNT

讀取的總數

READ_MIN

兩次連續讀取的最短時間（單位為秒）

READ_MAX

兩次連續讀取的最長時間（單位為秒）

READ_AVG

兩次連續讀取的平均時間（單位為秒）

COMMAND

程序的名稱

在這範例中，有三個應用程式特別突出：

```
  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
```

這三個應用程式的 **WRITE_CNT** 都大於 **0**，表示在評量期間，它們都有寫入資料。其中 **plasma** 的數值特別高：它進行了大部分的寫入動作，當然寫入之間的平均時間也最低。如果您想找出最不省電的應用程式，自然 **Plasma** 就成為最值得觀察的對象。

請藉由追蹤特定程序 ID 的系統呼叫，使用 **strance** 與 **ltrace** 指令來進一步檢視應用程式。就以目前的範例來說，您可以執行：

```
strace -p 2789
```

-

在這範例中，**strace** 的運作模式為每 45 秒開啟 KDE 的圖示快取檔案，寫入資料後又馬上關閉檔案。這導致不必要的硬碟寫入，因為檔案的 meta 資訊（更準確的說，是修改時間）有所改變。最終的修正是為了避免這些不必要的呼叫，因為圖示沒有任何改變。

2.4. 電池壽命工具組

現在 Red Hat Enterprise Linux 6 有了 **Battery Life Tool Kit** （電池壽命工具組，簡稱 BLTK），用來模擬、分析電池的壽命與效能。BLTK 會模擬特定的使用者群組，例如程式設計或一般辦公性事務，進行多種任務的測試，然後回報其結果。雖然 BLTK 是設計用來測試筆記型電腦的，但它也可以在啟動時使用 **-a** 選項，回報桌上型電腦的效能。

BLTK 能產生如真實電腦的工作負載量。舉例來說，**office** 負載會寫入文字再加以校正，並將同樣程序用於試算表裡。若與 **PowerTOP** 或其它稽核或分析工具一起執行 BLTK，就可以測試機器啟用、而不是閒置時的最佳化是否有效果。使用者可以在不同的設定中，用同樣的負載來進行測試，進而比較不同設定所帶來的結果。

請用以下指令安裝 BLTK：

```
yum install bltk
```

請用以下指令執行 BLTK：

```
bltk workload options
```

舉例來說，要執行 **idle**（閒置）負載 120 秒：

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

預設上可用的負載是：

-I, --idle

系統處於閒置狀態，用來作為與其他負載比較的基準值

-R, --reader

模擬讀取文件（預設上使用 **Firefox**）

-P, --player

模擬從光碟機觀看多媒體檔案（預設上使用 **mplayer**）

-O, --office

使用 **OpenOffice.org** 來模擬編輯文件

其他選項可以讓使用者指定：

-a, --ac-ignore

不管 AC 電源是否存在，都予以忽略（桌上型電腦必用）

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

測試時間（單位為秒）；請與 **idle** 負載合用此選項

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

指定檔案用於特定的負載，例如不存取光碟機，而改播放別的檔案給 **player** 負載使用

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

指定應用程式，用於特定的負載上，例如不用 **Firefox** 而改用其它瀏覽器，測試 **reader** 負載。

BLTK 支援大量的選項。欲知詳情，請參閱 **bltk** 的 man page。

BLTK 會將產生的結果儲存在 **/etc/bltk.conf** 配置檔所指定的目錄，預設上是 **~/.bltk/*workload*.results.*number*/**。舉例來說，**~/.bltk/reader.results.002/** 目錄會儲存 **reader** 負載的第三次測試（第一次測試不會有編號）。所有結果會散見於多個檔案中。要把這些結果濃縮到單一、易讀的格式，請執行：

```
bltk_report path_to_results_directory
```

這會在所有結果的目錄中，產生名為 **Report** 的文字檔案。要在終端機模擬程式中顯示結果，請使用 **-o** 選項。

```
bltk_report -o path_to_results_directory
```

2.5. TUNED 與 KTUNE

Tuned daemon 會監控系統元件，並根據監控結果，動態調整系統設定。動態微調負責多種系統元件在系統各種執行時間的行為模式。舉例來說，硬碟在開機與登入時的用量最高，但稍後在使用者使用 OpenOffice 或電子郵件軟體時，就幾乎沒有用到。相同地，CPU 與網路裝置會在不同的時間，有不同的用量。**Tuned** 會監控這些元件的活動，並對這些活動的改變做出回應。

我們以典型的辦公室環境為例：大部分時間乙太網路卡都處於閒置狀態。偶爾只有幾封電子郵件進出網路，或載入網頁。這種負載並不需要網路卡以預設的全速運行。**Tuned** 有網路裝置的監控與微調外掛程式，偵測網路的低活動時期，自動降低網路卡的速度，這可以節省能源。如果網路卡的活動突然增加了一段時間，例如下載 DVD 映像檔或一封附加了大型檔案的電子郵件，那麼 **tuned** 會偵測出來，並將網路卡的速度設為最大值，好為這些活動提供最高效能。這個原則也適用於 CPU 與硬碟等其它外掛程式。

預設上，網路裝置並沒有配置成這個樣子，因為改變速度需要幾秒鐘的時間，因此會對使用者造成直接、可見的衝擊。類似的考量也及於 CPU 與硬碟的外掛程式。當硬碟的轉速下降時，要再全速運轉需要幾秒的時間，在這其間系統會出現顯著的反應時間不足。對 CPU 外掛程式來說，延遲時間的影響是最小的，雖然使用者可能不會察覺，但還是在衡量得到的範圍內。

除了 **tuned** 以外，我們還提供了 **ktune**。**Ktune** 首次出現於 Red Hat Enterprise Linux 5.3，這架構與服務可以在特定的使用情況下，對電腦的效能進行最佳化。之後 **ktune** 更為精進，可以用作一般微調架構的固定班底。如 節 2.5.2, “Tuned-adm” 所述，它主要是用在多種事先定義的 profile 上。

請以此指令安裝 tuned 套件、以及相關的 **systemtap** 程序檔：

```
yum install tuned
```

安裝 tuned 套件時，也會在 **/etc/tuned.conf** 中設定一組範例配置檔，並啟動預設的 profile。

執行以下指令以啟動 **tuned**：

```
service tuned start
```

要在每次開機時啟動 **tuned**，請執行：

```
chkconfig tuned on
```

Tuned 本身有額外的選項，讓使用者手動執行時使用。可用的選項有：

-d, --daemon

以 daemon 方式啟動 **tuned**，而不是在前景執行。

-c, --conffile

使用指定了路徑與檔案的配置檔，例如 **--conffile=/etc/tuned2.conf**。預設值是 **/etc/tuned.conf**。

-D, --debug

使用最高等級的日誌紀錄。

2.5.1. tuned.conf 檔案

tuned.conf 檔案包含了 **tuned** 所使用的配置設定。預設上，這檔案位於 **/etc/tuned.conf**，但使用者可以指定不同的位置與名稱，方法是使用 **--conffile** 選項啟動 **tuned.conf**。

這個配置檔案一定得包含 **[main]** 一節，定義 **tuned** 的通用參數。這檔案的每個外掛程式都會有自己專用的一節。

[main] 一節包含了以下選項：

interval

tuned 監控、微調系統的間隔，單位為秒。預設值為 **10**。

verbose

是否提供詳盡的輸出結果。預設值為 **False**（否）。

logging

指定寫入日誌的訊息之最低優先順序。以降冪排列，可用的值有 **critical**、**error**、**warning**、**info**、與 **debug**。預設值是 **info**。

logging_disable

指定寫入日誌的訊息之最高優先順序，低於這個優先順序的訊息都不會被紀錄。以降冪排列，可用的值有 **critical**、**error**、**warning**、**info**、與 **debug**。**notset** 會停用這個選項。

每個外掛程式都有自己的一節，以中括號括住外掛程式的名稱；例如 **[CPUTuning]**。每個外掛程式都可以使用自己的選項，但以下會套用至所有外掛程式：

enabled

啟用外掛程式與否。預設值為 **True**（是）。

verbose

訊息輸出是否詳盡與否。如果此外掛程式沒有設定，這個值會引用自 **[main]**。

logging

指定寫入日誌的訊息之最低優先順序。如果此外掛程式沒有設定，這個值會引用自 **[main]**。

配置檔案的範例如下：

```
[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

通常閱讀詳盡的稽核與分析資料會很花時間，而且閱讀本身還可能會多耗電，所以這樣作並不值得。之前的作法是直接使用預設值。因此，Red Hat Enterprise Linux 6 針對兩種極端的不同使用方式，提供了各式各樣的 profile，再加上 **tuned-adm** 工具能讓使用者透過命令列，在這些 profile 之間切換。Red Hat Enterprise Linux 6 包括多種事先定義的 profile，適用於多種典型的情境，任使用者透過 **tuned-adm** 指令選擇、啟動；但您也可以自行建立、修改或刪除 profile。

要列出所有的 profile，以及目前啟用中的 profile，請執行：

```
tuned-adm list
```

僅顯示目前啟用中的 prfile，請執行：

```
tuned-adm active
```

要在可用的 profile 之間切換，請執行：

```
tuned-adm profile profile_name
```

例如：

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

要停用所有微調功能：

```
tuned-adm off
```

在首次安裝 **tuned** 時，「**default**」（預設的）profile 會被啟動。Red Hat Enterprise Linux 6 也包括以下事先定義好的 profile：

default

預設的節能設定檔。在所有可用設定檔中，這對節省能源有著最低的衝擊，它只啟用 **tuned** 的 CPU 與磁碟外掛程式。

desktop-powersave

可以直接用於桌上型電腦的節能設定檔。它會為 SATA 主機控制卡啟用 ALPM 省電模式（詳情請參閱 節 3.8, “積極性連結電源管理”），以及 **tuned** 的 CPU、乙太網路與磁碟外掛程式。

server-powersave

可直接用於伺服器系統的節能設定檔。它會為 SATA 主機控制卡啟用 ALPM 省電模式、透過 **HAL** 停用光碟機的輪詢功能（詳情請參閱 hal-disable-polling 的 man page），並啟用 **tuned** 的 CPU 與磁碟外掛程式。

laptop-ac-powersave

這是個中度衝擊的節能設定檔，可直接用於接上電源的筆記型電腦。它會為 SATA 主機控制卡啟用 ALPM 省電模式、WiFi 省電模式，以及 **tuned** 的 CPU、乙太網路與磁碟外掛程式。

laptop-battery-powersave

這是個高度衝擊的節能設定檔，可直接用於使用電池的筆記型電腦。它會啟用之前使用的所有省電機制，再加上給低喚醒系統的多核心省電排程程式，並確定 **ondemand** 調速程式是啟用的，而 AC97 音

效省電功能同時也是啟用的。您可以在任何系統上使用此設定檔，以節省最多的電力；這並不限於以電池運作的筆記型電腦。但使用此設定檔會對效能造成顯著的衝擊，尤其是磁碟與網路的延遲時間會變長。

spindown-disk

這是個強大的節能設定檔，用於使用傳統硬碟的機器。它增加磁碟寫入的值，降低磁碟的 swap ，並停用同步日誌功能。所有分割區都會以 **noatime** 選項重新掛載。所有 **tuned** 嵌入程式都會被停用。

throughput-performance

供伺服器使用的設定檔，可微調 I/O 的效能。它會停用 **tuned** 與 **ktune** 的省電機制，啟用 **sysctl** 設定，改進硬碟與網路 I/O 的吞吐效能，並切換到 **deadline scheduler**。

latency-performance

這是個伺服器設定檔，用於典型的延遲效能微調。這個設定檔會停用動態式的微調機制與透明的巨型頁面。它使用 **performance** 調速程式來透過 **cpuspeed** 進行 p-state，並將 I/O 排程器設為 **deadline**。此外，在 Red Hat Enterprise Linux 6.5 和更新版本中，此設定檔會請求值為 **1** 的 ***cpu_dma_latency*** 參數。在 Red Hat Enterprise Linux 6.4 和更舊版本中， **cpu_dma_latency** 則會請求 **0** 這個值。

enterprise-storage

用來為企業級伺服器配置改善整體效能的伺服器設定檔。這會切換至 **deadline scheduler**，並停用一些 I/O 限制，大幅增進吞吐量。

virtual-guest

這個設定檔已針對虛擬機器進行優化。它基於企業儲存設定檔，不過卻也降低了虛擬記憶體的 swap。此設定檔能使用於 Red Hat Enterprise Linux 6.3 和更新版本中。

virtual-host

基於 enterprise-storage 設定檔，virtual-host 會降低虛擬記憶體的 swap 並且更加積極地回寫中途分頁（dirty page）。非 root 和非 boot 的檔案系統將以 **barrier=0** 掛載。此外，由 Red Hat Enterprise Linux 6.5 起，***kernel.sched_migration_cost*** 參數將會被設為 5 毫秒。在 Red Hat Enterprise Linux 6.5 以前，***kernel.sched_migration_cost*** 使用了 0.5 毫秒作為預設值。

sap

一個為 SAP 軟體效能優化的設定檔。它基於 enterprise-storage 設定檔。Sap 設定檔會額外微調有關於共享記憶體與信號的 sysctl 設定，以及一項程序能擁有的最大記憶體映射數量。

所有 profile 都儲存在 **/etc/tune-profiles** 的子目錄下。所以 **/etc/tune-profiles/desktop-powersave** 包含了本 profile 所需的檔案與設定。每個檔案都包含最多四個檔案：

tuned.conf

這個 profile 所使用的 tuned 服務配置

sysctl.ktune

ktune 所使用的 **sysctl**。它的格式與 **/etc/sysconfig/sysctl** 檔案無異（詳情請參閱 sysctl 與 sysctl.conf 的 man page）。

ktune.sysconfig

ktune 自己的配置檔，通常是 **/etc/sysconfig/ktune**。

ktune.sh

ktune 服務所使用、類似 **init** 的 shell script，可以在開機階段執行特定的指令，以微調系統。

要建立全新 profile 的最好方法，是從修改現有的 profile 開始。**laptop-battery-powersave** profile 已經包含了非常豐富的微調選項，是很好的起點。您只要複製整個目錄，並賦予新名稱即可，如下：

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

請修改新 profile 的所有檔名，以符合個人需求。舉例來說，如果您需要偵測 CD 已經被置換，只要把 ktune.sh script 裡的一行加上註解，就可以停用此最佳化選項：

```
# 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 與 DEVKIT-POWER

Red Hat Enterprise Linux 6 的 **DeviceKit-power** 會假設電源管理功能是 **HAL** 的一部分，其它功能是舊版 Red Hat Enterprise Linux「**GNOME 電源管理程式**」的一部分（詳情請參閱 節 2.7, “GNOME 電源管理程式”）。**DeviceKit-power** 提供了一組 daemon，一組 API，以及命令列工具。系統上的每一組電力來源，不管是不是實際的裝置，都會以裝置方式來呈現。舉例來說，筆記型電腦的電池與 AC 電源，都以裝置的方式來呈現。

您可以使用 **devkit-power** 命令與以下選項，來存取命令列工具：

--enumerate, -e

顯示系統上每個電力來源的物件路徑，例如：

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

--dump, -d

顯示系統上所有電力裝置的參數。

--wakeups, -w

顯示系統的 CPU 喚醒次數。

--monitor, -m

監控系統上電力裝置的改變，例如連上或移除 AC 電源，或電池的電力耗盡。請按下 **Ctrl+C**，停止監控系統。

--monitor-detail

監控系統上電力裝置的改變，例如連上或移除 AC 電源，或電池的電力耗盡。**--monitor-detail** 選項所顯示的內容會比 **--monitor** 選項更為詳盡。請按下 **Ctrl+C**，停止監控系統。

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

顯示特定物件路徑的所有資訊。舉例來說，要取得系統上特定電池（**/org/freedesktop/UPower/DeviceKit/power/battery_BAT0**）的資訊，請執行：

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

2.7. GNOME 電源管理程式

「**GNOME 電源管理程式**」（GNOME Power Manager）是 GNOME 桌面環境所附的一個 daemon。之前在 Red Hat Enterprise Linux 裡，「**GNOME 電源管理程式**」所提供的大部分電源管理功能，現在都已成為 Red Hat Enterprise Linux 6「**DeviceKit-power**」的一部分（詳情請參閱 節 2.6, “DeviceKit-power 與 devkit-power”）。然而，「**GNOME 電源管理程式**」依舊是這功能的前端程式。它會透過系統的工具列程式，將系統電源狀態的任何改變告訴使用者；例如，原本使用電池的筆記型電腦，現在插上了電源。這程式也會回報電池的狀態，並在電力存量不足時提出警告。

「**GNOME 電源管理程式**」也能讓您配置一些基本的電源管理設定。要存取這些設定，請按下系統工具列的「**GNOME 電源管理程式**」圖示，然後點選「**偏好設定**」。

「**電源管理偏好設定**」畫面包含了兩個分頁：

- **使用 AC 電源**
- **一般**

在筆記型電腦上，「**電源管理偏好設定**」畫面包含了第三個分頁：

- **使用電池電源**

「**使用 AC 電源**」與「**使用電池電源**」分頁可以用來指定電腦在一定時間不使用之後，就關閉螢幕；一段時間之後，就進入睡眠模式；甚至要求硬碟降低轉速。「**使用電池電源**」分頁也能用來設定顯示器的亮度，並選擇電池的電量過低時該如何處理。例如「**GNOME 電源管理程式**」預設上會在電池的電量過低時，自動進入休眠狀態。請使用「**一般**」分頁來設定按下（硬體）電源按鈕與休眠按鈕時該如何處理，並指定「**GNOME 電源管理程式**」的圖示該在哪種情況下，出現在系統工具列上。

2.8. 其它稽核方式

Red Hat Enterprise Linux 6 提供了多種工具，對電腦系統進行稽核與分析。這些軟體大多可以拿來當作輔助工具，用來驗證您發現的現象，或是您想要更進一步了解某些元件。這些工具也大多可以拿來當作效能微調工具。這些工具有：

vmstat

vmstat 提供了程序、記憶體、分頁、區塊 I/O、trap、以及 CPU 的詳細資訊。請使用這工具來更進一步檢視系統的整體表現，看看哪些部份處於忙碌狀態。

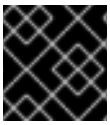
iostat

iostat 跟 **vmstat** 非常類似，但用於磁區裝置的 I/O。它也提供了非常詳盡的輸出與統計資料。

blktrace

blktrace 提供了非常詳盡的區塊 I/O 追蹤資料。它會把資訊依照應用程式，切成獨立區塊。這與 **diskdevstat** 合用時非常有用。

章 3. 核心架構與機制



重要

要使用本章所描述的 **cpupower** 指令，請確定您已經安裝 cpupowerutils 套件。

3.1. CPU 的閒置狀態

x86 架構的 CPU 支援多種狀態，讓 CPU 可以被停用或以低效能方式運行。這些狀態稱之為 *C-state*，能讓系統停用非使用中的 CPU，藉以節省電源。C-state 從 C0 開始向上累進，數字越高表示 CPU 的功能越少，也越省電。雖然 C-state 的各種數字在各種處理器上都類似，但特定 C-state 在特定處理器（或同類型處理器）上的意義，會與另一種處理器的意義不同；也就是說，一種處理器的 C3 狀態與另一種處理器的 C3 不同。C-State 0-3 的定義如下：

C0

運作中或執行中。在此狀態下，CPU 處於工作狀態，完全不閒置。

C1, Halt

這狀態表示處理器並不處於執行狀態，但通常也不處於低耗電狀態。CPU 能在完全不延遲的情形下，繼續進行處理。所有提供 C-State 的處理器都必須支援此狀態。Pentium 4 處理器支援更精進的 C1 狀態，稱之為 C1E，是處於低耗電狀態的。

C2, Stop-Clock

這狀態表示此處理器的時脈是停止的，但它仍然為其註冊碼與快取保持完整的狀態，因此在時脈重新開始時，就可以立即開始處理、運算。這是選用的狀態。

C3, Sleep

這表示處理器的確進入睡眠狀態，而且不會保留最新的快取版本。有鑑於此，從這狀態喚醒處理器，會比 C2 狀態要來得長。同樣的，這是選用的狀態。

要檢視 CPUidle 驅動程式的可用閒置狀態與其它統計資料，請執行以下指令：

```
cpupower idle-info
```

擁有「Nehalem」微架構功能的 Intel CPU 有新的 C-state：C6，可以將 CPU 所消耗的瓦數降到 0；但一般來說，實際降低的瓦數大約介於 80% 到 90% 之間。Red Hat Enterprise Linux 6 的 kernel 對這新的 C-state 已經做了最佳化。

3.2. 使用 CPUFREQ 調速程式

降低電力消耗量與熱能輸出的最有效方法，是在系統上使用 CPUfreq。CPUfreq 也被稱為 CPU 速度調節，能動態調整處理器的時脈速度。這能讓系統以較低的時脈運作，以節省電力。變化頻率（不管是把時脈速度變快或變慢）的規則，以及何時變化頻率，都由 CPUfreq 調速程式（governor）來定義。

調速程式會定義系統 CPU 的電力特性，進而影響 CPU 的效能。對負載來說，每個調速程式都有其獨特的行為、目的、與適用性。本節描述了如何選擇與配置 CPUfreq 調速程式、各種調速程式的特性、以及每種調速程式適用於哪些負載。

3.2.1. CPUfreq 調速程式的類型

本節列出並描述了 Red Hat Enterprise Linux 6 的 CPUfreq 調速程式類型

cpufreq_performance

這個 Performance（效能）調速程式會強迫 CPU 使用最高的時脈頻率運作。這個頻率會以靜態方式設定，而且不會改變。因此，這個特定的調速程式「*不會提供任何省電功能*」。這只適用於高工作負載時段，在這期間 CPU 鮮有（或從沒有）閒置的時候。

cpufreq_powersave

相反地，Powersave（省電）調速程式會強迫 CPU 使用最低的時脈頻率運作。這頻率會以靜態方式設定，而且不會改變。因此，這個調速程式會以最節省電力的方式運作，但會導致「*最低的 CPU 效能*」。

但「Powersave」這個名詞有時候會引起混淆，因為基本上，較慢的 CPU 在滿載的情況下，會比快速而不滿載的 CPU 耗電。因此，如果您預期某段時間內 CPU 的負荷應該不會太重，就可以將 CPU 設在 Powersave 調速程式；但這段時間內的無預期高負載就可能導致系統消耗更多電力。

簡單來說，Powersave 調速程式是 CPU 的「限制速度程式」，而不是「節省電力程式」。它在可能會過熱的系統與環境中，非常有用。

cpufreq_ondemand

Ondemand（視需要）調速程式能在系統負載高的時候，使用 CPU 的最高時脈運行；並在系統閒置時，使用最低的時脈頻率。這讓系統根據系統負載，動態調整電力消耗；但代價是「*切換時脈時會導致延遲*」。因此，如果負載的變動太過頻繁，Ondemand 調速程式會因為切換時脈的次數過多，而降低 Performace 或 Powersave 所帶來的好處。

對於大部分系統來說，Ondemand 調速程式可以達到熱能、電力、效能、與管理的最佳平衡。當系統只在一天的某個時段忙碌的情形下，Ondemand 調速程式會根據負載，自動在最高與最低頻率之間切換，而不需要進一步干預。

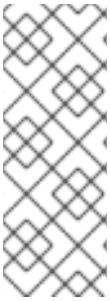
cpufreq_userspace

Userspace 調速程式能讓 userspace（使用者空間）的程式（或任何以 root 身份執行的程序）來設定頻率。此調速程式通常會與 **cpuspeed** daemon 一起使用。在所有調速程式中，Userspace 是最高度客製化的；而且根據配置方式，它可以提供效能與電力消耗的最佳平衡。

cpufreq_conservative

跟 Ondemand 調速程式類似，Conservative（保守）調速程式也可以根據使用量調整時脈頻率。然而，Ondemand 會以更積極的方式來進行（最高頻率或最低頻率二選一）；而 Conservation 調速程式會以漸進方式調整頻率。

這表示 Conservative 調速程式會調整到適合負載的時脈頻率，而不是選擇最高頻率或最低頻率。雖然這可能可以省下可觀的電力，但與 Ondemand 調速程式比起來，「*延遲情況會更嚴重*」。



注意

您可以使用 **cron** job 來啟動調速程式。這能讓您在每日的特定時間，自動設定不同的調速程式。因此，您可以在閒置時段（例如下班後）使用低頻率的調速程式，並在高負載時調回高頻率的調速程式。

欲知啟用調速程式的方法，請參閱 節 3.2.2, “設定 CPUfreq” 的 過程 3.2, “啟用 CPUfreq 調速程式”。

3.2.2. 設定 CPUfreq

在選擇與配置 CPUfreq 調速程式之前，請先加入適當的 CPUfreq 驅動程式。

過程 3.1. 如何新增 CPUfreq 驅動程式

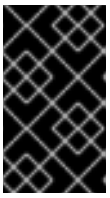
1. 請使用以下指令來檢視系統上有哪些可用的 CPUfreq：

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

2. 使用 **modprobe** 來新增正確的 CPUfreq 驅動程式。

```
modprobe [CPUfreq driver]
```

使用以上指令時，請記得移除 **.ko** 延伸檔名。



重要

在選擇 CPUfreq 驅動程式時，請選擇 **acpi-cpufreq** 而不是 **p4-clockmod**。後者會降低 CPU 的時脈，但不會降低電壓。而前者會降低電壓與時脈，使用更少電力並消耗更少的熱能。

您也可以用以下指令，檢視某個特定 CPU 的可用調速程式：

```
cpupower frequency-info --governors
```

有些 CPUfreq 調速程式可能無法使用。在這種情況下，請使用 **modprobe** 來新增所需的 kernel 模組，以啟用您想要使用的特定 CPUfreq 調速程式。這些 kernel 模組可以在 **/lib/modules/*[kernel version]*/kernel/drivers/cpufreq/** 中找到。

過程 3.2. 啟用 CPUfreq 調速程式

1. 如果有個調速程式並沒有列在 CPU 的可用清單上，請使用 **modprobe** 來啟用您想要使用的調速程式。舉例來說，如果您的 CPU 還無法使用 **ondemand** 調速程式，請執行以下指令：

```
modprobe cpufreq_ondemand
```

2. 一旦調速程式列在 CPU 的可用清單上，您可以用以下指令啟用：

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

3.2.3. 微調 CPUfreq 政策與速度

一旦您選擇了適合的 CPUfreq 調速程式之後，就可以使用 **cpupower frequency-info** 指令檢視 CPU 速度與政策資訊，並進一步透過 **cpupower frequency-set** 選項微調每個 CPU 的速度。

cpupower frequency-info 指令有以下選項：

- **--freq** – 根據 CPUfreq 和新的速度，顯示目前的 CPU 速度，單位為 KHz。
- **--hwfreq** – 根據硬體，顯示 CPU 目前的時脈，單位為 KHz（僅有 root 可使用）。
- **--driver** – 顯示設定此 CPU 的 CPUfreq 驅動程式為何。
- **--governors** – 顯示 kernel 裡可用的 CPUfreq 調速程式。如果您想要使用沒有列在這個檔案中

的 CPUfreq 調速程式，請參閱 節 3.2.2, “設定 CPUfreq” 裡的 過程 3.2, “啟用 CPUfreq 調速程式” 操作指引。

- **--affected-cpus** – 列出需要頻率協調程式之 CPU。
- **--policy** – 顯示目前 CPUfreq 政策的範圍，單位為 KHz，以及目前使用的調速程式。
- **--hwlimits** – 列出 CPU 的可用頻率，單位為 KHz。

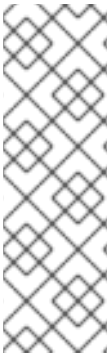
cpupower frequency-set 指令有以下選項：

- **--min *<freq>*** 與 **--max *<freq>*** – 設定 CPU 的「*政策限制*」（policy limits），單位為 KHz。

重要

設定政策限制時，請在 **--min** 之前設定 **--max**。

- **--freq *<freq>*** – 設定 CPU 的特定時脈，單位為 KHz。您只能設定 CPU 政策限制之間的速度（一如 **--min** 與 **--max** 之間的設定）。
- **--governor *<gov>*** – 設定新的 CPUfreq 調速程式。



注意

如果您尚未安裝 cpupowerutils 套件，CPUfreq 設定可以在 **/sys/devices/system/cpu/*[cpuid]*/cpufreq/** 中的可調整參數中找到。設定與數值可以透過撰寫這些可調整參數而改變。舉例來說，要設定 cpu0 的最低時脈為 360 KHz，請執行：

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

3.3. CPU 監控

cpupower 有多種監控內容，提供閒置與睡眠狀態的統計資料與時脈資訊，並在處理器拓樸上回報。一些監控內容與處理器有關，而其它的則與任何處理器相容。關於每種監控內容所評量的詳細資料與相容的系統，請參閱 **cpupower-monitor** man page。

請使用 **cpupower monitor** 指令的以下選項：

- **-l** – 列出系統上的所有監控。
- **-m *<monitor1>, <monitor2>*** – 顯示特定監控。識別子可透過執行 **-l** 找到。
- **command** – 顯示閒置的統計數據，以及特定指令的處理器需求。

3.4. CPU 的省電政策

cpupower 提供了規範處理器之省電政策的方法。

請使用 **cpupower set** 指令的以下選項：

--perf-bias *<0-15>*

允許受支援的 Intel 處理器上的軟體更主動地回應，以決定最佳效能與省電之間的平衡。這並不會覆寫其它省電政策。可指定的值從 0 到 15，0 是最佳效能，而 15 是最省電。

預設上，這選項可套用到所有處理器核心上。要僅僅套用至一個核心上， 請新增 **--cpu *<cpulist>*** 選項。

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

在其它 CPU 套件使用電力前，限制單一 CPU 套件中系統處理器對處理器核心所使用的電力。0 表示沒有限制；1 表示一開始只使用單一 CPU 套件；而 2 在處理任務喚醒時，會額外著重於半閒置的 CPU 套件。

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

限制系統程序在其它核心上使用電力前，對一個處理器核心之同類型執行績所使用的電力。0 表示沒有限制；1 表示一開始只使用單一 CPU 套件；而 2 在處理任務喚醒時，會額外著重於半閒置的 CPU 套件。

3.5. SUSPEND（暫停）與 RESUME（重新開始）

當系統進入暫停狀態時，kernel 會要求驅動程式儲存狀態，然後卸載驅動程式。當系統重新開始時，kernel 會重新載入這些驅動程式，而驅動程式會對裝置重新套用程式。驅動程式完成這項工作的能力，決定了系統是否可以成功地重新開始。

就這一點來說，顯示卡驅動程式問題較多，因為 ACPI（*Advanced Configuration and Power Interface*，進階設定與電源介面）規格並不需要系統韌體來重新套用顯示卡的程式。因此，除非顯示卡驅動程式能從完全未初始的狀態對硬體套用程式，否則驅動程式可能會讓系統無法重新開始。

Red Hat Enterprise Linux 6 為新的顯示卡晶片提供了更廣泛的支援，這能確保使用者在更多平台上能暫停、重新開始無誤。對於 NVIDIA 晶片的支援更是大大改進，尤其是 GeForce 8800 系列。

3.6. 無計時 KERNEL

之前 Linux kernel 會以事先定義的頻率（100 Hz、250 Hz、或 1,000 Hz，視平台而定），定期中斷系統上的每個處理器。kernel 會查詢 CPU 正在執行什麼程序，然後以此根據紀錄與負載來進行處理。kernel 不會理會 CPU 的電源狀態，進行此中斷，這稱為 *timer tick*（計時器的滴答）。因此，即使是閒置中的 CPU，每秒也至少會回應 1,000 次。在套用了省電功能的系統上，計時器的滴答需求會讓 CPU 無法處於閒置狀態，因此無法省下任何電力。

Red Hat Enterprise Linux 6 的 kernel 使用了 *tickless*（無計時）功能：也就是說，它會以「視需求發出中斷」的計時器，取代舊有「定期發出中斷」的計時器。這樣一來，閒置的 CPU 就會保持在閒置狀態，直到新的任務進入佇列為止；同時處於低耗電的 CPU 就可以更長久地處於這些狀態下。

3.7. 主動狀態電源管理

主動狀態電源管理（ASPM，Active-State Power Management）能節省 *PCI Express*（或稱 PCIe，Peripheral Component Interconnect Express）子系統的電力，方法是在 PCIe 裝置不處於使用狀態時，降低 PCIe 連線的電源狀態。ASPM 會控制連線兩端的電源狀態，甚至在某一端的裝置仍處於全力電源狀態下，仍然可以節省連線的電源。

啟用 ASPM 時，連線在切換電源狀態時需要時間，因此延遲時間會增加。ASPM 有三種決定電源狀態的政策。

預設值

根據系統韌體（例如 BIOS）來設定 PCIe 連線的電源狀態。這是 ASPM 的預設狀態。

powersave（省電模式）

設定 ASPM 盡可能的節省電源，不理會效能因素。

performance（效能）

停用 ASPM 讓 PCIe 連線能以最高效能運作。

ASPM 政策可以透過開機時以 ***pcie_aspm*** kernel 參數來啟用或停用，其中 **pcie_aspm=off** 會關閉 ASPM，而 **pcie_aspm=force** 會啟動 ASPM，即使裝置不支援 ASPM 亦然。

ASPM 政策的設定位於 **/sys/module/pcie_aspm/parameters/policy**，但可以在開機時以 ***pcie_aspm*** kernel 參數來設定，舉例來說，其中 **pcie_aspm.policy=performance** 會設定 ASPM 的效能政策。

警告

如果使用者設定了 **pcie_aspm=force**，不支援 ASPM 的硬體可能會導致系統停止回應。在設定 **pcie_aspm=force** 之前，請確定系統上所有的 PCIe 硬體都支援 ASPM。

3.8. 積極性連結電源管理

ALPM（*Aggressive Link Power Management*，積極性連結電源管理）是一種硬碟的省電技術，方法是讓一組 SATA 連線在閒置時，使用低度用電的設定。如果這組連線的佇列中有 I/O 需求進來，ALPM 會自動設定 SATA 連線回到主動式電源狀態。

ALPM 所引介的省電功能會造成存取磁碟延遲。因此，只有在系統會有長期 I/O 閒置時間時，才使用 ALPM。

ALPM 只能用在使用 AHCI（*Advanced Host Controller Interface*，進階主控制介面）的 SATA 控制晶片上。欲知更多 AHCI 的詳細資訊，請參閱 http://www.intel.com/technology/serialata/ahci.htm。

如果您的電腦支援 ALPM，預設上就會啟用。ALPM 有三種模式：

min_power

這個模式會在磁碟沒有 I/O 的情形下，以最低電源狀態（SLUMBER）連線。這模式在閒置時間較長的情形下會很有幫助。

medium_power

這個模式會在磁碟沒有 I/O 的情形下，以次低電源狀態（PARTIAL）連線。這個模式適用於間歇性的電源狀態（例如一段時間有大量 I/O，然後是一段閒置時間），對效能的影響較小。

medium_power 模式能讓連線根據負載，在 PARTIAL 與完全供電（ACTIVE）狀態下游移。請注意，連線無法在 PARTIAL 與 SLUMBER 之間直接游移；電源狀態必須先切換到 ACTIVE，再游移至另一種狀態。

max_performance

停用 ALPM；磁碟沒有任何 I/O 動作時，連線也不會進入任何低電源狀態。

要檢查 SATA 主機介面是否支援 ALPM，請查看 **/sys/class/scsi_host/host*/link_power_management_policy** 是否存在。要改變設定，請把本節所描述的值寫入這些檔案；或請打開這些檔案，檢查這些數值。

重要

將 ALPM 設定為 **min_power** 或 **medium_power** 會自動停用「熱插拔」（Hot Plug）功能。

3.9. RELATIME 磁碟存取最佳化

POSIX 標準需要作業系統維護檔案系統的 metadata，紀錄每個檔案最後被存取的時間。這個時間戳記稱為 **atime**，而維護這資料需要不斷地寫入硬碟。這會讓儲存裝置一直處於忙碌與開機狀態。因為一些應用程式會使用 **atime** 資料，所以儲存裝置會浪費電力。更有甚者，即使檔案不是從存儲裝置直接讀取，而是從快取記憶體中讀取，一樣會有寫入的動作。在有些時候，Linux kernel 支援 **mount** 的 **noatime** 選項，使用這選項掛載的檔案系統就不會寫入 **aime** 資料。然而，僅關掉這項功能還是會有問題，因為有些應用程式仰賴 **atime** 的資料，如果找不到這資料，就會失敗。

Red Hat Enterprise Linux 6 的 kernel 支援另一種方案：**relatime**。**Relatime** 會維護 **atime** 資料，但不會在每次檔案被存取時運作。啟用了這選項之後，**atime** 資料只有在 **atime** 資料上一次被修改（**mtime**）之後有所更動時，或在檔案被存取之後又過了一定時間（預設值為一天）時，檔案才會被寫入。

預設上，所有檔案系統都會以啟用 **relatime** 的方式掛載。您可以使用 **norelatime** 選項掛載任何檔案系統，而不啟用此方式。

3.10. 用電控制

Red Hat Enterprise Linux 6 支援許多新硬體所使用的用電控制功能，例如 HP 的「動態用電控制」（DPC，*Dynamic Power Capping*）以及 Intel 的 Node Manager（簡稱 NM）動態電源管理技術。用電控制功能可讓系統管理者限制伺服器所消耗的電源；但它也能讓管理者更有效地規劃資料中心，因為電力超過負載的風險就可以大幅降低。系統管理者可以在同樣的機房裡建置更多的伺服器，搭配用電控制功能，這樣重運量時的電力負荷也不會超過整體電源供應量。

HP 動態用電控制

動態用電控制（Dynamic Power Capping）可以在幾款 ProLiant 與 BladeSystem 伺服器上找到，它能讓系統管理者設定一或多台伺服器的電力使用上限。不管伺服器的負載為何，其電力消耗量都不會超過這個上限。在到達電力消耗上限之前，這機制不會有任何動作。一旦到了上限，管理處理器會調整 CPU 的 P-state，並限制其時脈，以降低電源消耗量。

動態用電控制會修改 CPU 的行為，這與作業系統無關；然而 HP 的 iLO2（*integrated Lights-Out 2*）韌體能讓作業系統存取管理處理器（management processor），因此使用者所執行的應用程式可以查詢管理處理器。Red Hat Enterprise Linux 6 的 kernel 已包括 HP iLO 與 iLO2 韌體，讓程式得以查詢位於 **/dev/hpilo/d*X*ccb*N*** 的管理處理器。kernel 也包括 **hwmon sysfs** 延伸介面，以支援動態用電控制功能，外加 ACPI 4.0 電力計所使用的 **hwmon** 驅動程式，用於 **sysfs** 介面。有了這些功能，作業系統與使用者空間的工具可以讀取用電控制的配置值，以及系統目前的用電量。

欲知 HP 動態用電控制的進一步詳情，請參閱《*HP Power Capping and HP Dynamic Power Capping for ProLiant Servers*》一文，網址為 http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01549455/c01549455.pdf。

Intel Node Manager

Intel 的 Node Manager 在系統上加了用電控制功能，使用的是處理器的 P-state 與 T-state 來限制 CPU 的效能，進而達成節省電力的目的。設定電源管理政策後，系統管理者可以配置系統，使其在低負載時（例如夜間或週末）消耗更少電力。

Intel Node Manager 會使用透過 ACPI（*Advanced Configuration and Power Interface*，進階配置與電源介面）來使用 OSPM（*Operating System-directed configuration and Power Management*，作業系統導向的配置與電源管理），進而調整 CPU 效能。當 Intel Node Manger 提示 OSPM 驅動程式改變 T-state 時，OSPM 會做出相對應的 P-state；反之亦然。這些變化都會自動發生，不需要作業系統進一步干預。管理者使用 DCM（*Intel Data Center Manager*，Intel 資料中心管理員）來配置、監控 Intel Node Manager。

欲知更多關於 Intel Node Manager 的詳情，請參閱《*Node Manager – A Dynamic Approach To Managing Power In The Data Center*》一文，網址為 http://communities.intel.com/docs/DOC-4766。

3.11. 加強的圖形電源管理

Red Hat Enterprise Linux 6 會減少幾個不必要的電力消耗來源，以節省圖形與顯示裝置的電源。

重新設定 LVDS 的時脈

LVDS（低電壓差動訊號傳輸，Low Voltage Differential Signaling）是一種在銅線上傳送電子訊號的系統。這系統著名的應用之一，是傳送像素資訊到筆記型電腦的 LCD（*liquid crystal display*，液晶螢幕）上。所有顯示器都有「*更新頻率*」（refresh rate）：從顯示卡收到訊號，並在螢幕上重新繪製的速率。一般來說，螢幕每秒鐘會收到 60 次更新資料（也就是 60 Hz）。當螢幕與顯示卡以 LVDS 連接時，LVDS 系統會在每一個更新週期消耗電力。閒置時，許多 LCD 螢幕的更新頻率會降低到 30 Hz，但不會有任何顯著的影響（與 CRT (*cathode ray tube*) 陰極射線管螢幕不同，降低更新頻率會導致螢幕上的字元閃爍）。內建於 Red Hat Enterprise Linux 6 的 Intel 顯示晶片驅動程式會在螢幕閒置時自動降低時脈（*downclocking*），節省約 0.5 W 的電力。

啟用記憶體自我更新

SDRAM（*Synchronous dynamic random access memory*，同步動態存取記憶體）用作顯示卡的記憶體，每秒會充電成千上萬次，好讓每個記憶體單元能保存資料。記憶體控制晶片除了管理資料流進流出記憶體以外，還負責啟動這些更新週期。然而 SDRAM 也有低電壓的自我更新（*self-refresh*）模式。在這模式中，記憶體會使用內部時脈來產生自己的更新週期，這讓系統可以在不損及記憶體現有資料的情形下，關閉記憶體控制晶片。Red Hat Enterprise Linux 6 的 kernel 會在 Intel 顯示晶片閒置時，啟動記憶體自我更新功能，這樣可以節省約 0.8 W。

降低 GPU 時脈

通常 GPU（圖形處理器，Graphical Processing Unit）包含了內部時脈，管理 GPU 的多種內部線路。Red Hat Enterprise Linux 6 的 kernel 可以降低一些 Intel 與 ATI 的 GPU 之頻率。在一定時間內降低 GPU 元件的週期數，可以節省電力。kernel 會在 GPU 閒置時，自動降低時脈速度，並在 GPU 活動增加時調昇速度。降低 GPU 時脈週期可以節省多至 5 W 的電力。

降低 GPU 的電力

Red Hat Enterprise Linux 6 的 Intel 與 ATI 圖形驅動程式可以偵測電腦是否連接了螢幕；若否，則完全關閉 GPU。這功能對於伺服器尤其好用，因為通常伺服器沒有螢幕。

3.12. RFKILL

許多電腦都有發出無線電的裝置，包括 Wi-Fi、藍牙、以及 3G 裝置。這些裝置都會消耗電力，如果裝置不在使用中的話，那會造成無謂的浪費。

RFKill 是 Linux kernel 的一個子系統，它提供的介面可以讓使用者查詢、啟動或停用無線電裝置。無線電裝置停用時，這些裝置會處於可以由軟體重新啟用的狀態（*soft block*），或無法由軟體重新啟用的狀態（*hard block*）。

RFKill 的核心提供了 API 存取子系統。支援 RFkill 的 kernel 驅動程式可以使用此 API 與 kernel 溝通，並納入啟用或停用裝置的方法。除此之外，RFKill 核心提供了提示功能，讓使用者的應用程式解譯傳輸狀態的方式。

RFKill 介面位於 **/dev/rfkill**，其中包含了系統無線電傳輸的現有狀態。每個裝置都有其現有的 RFKill 狀態，註冊於 **sysfs** 裡。除此之外，RFKill 會對能使用 RFKill 的裝置之每個狀態的改變，發出 *uevents*。

Rfkill 命令列工具能讓使用者查詢、改變系統上支援 RFKill 的裝置。要取得這工具，請安裝 **rfkill** 套件。

請執行 **rfkill list** 指令，以取得裝置清單；每個裝置都有 *index number*（索引編號），從 **0** 開始。使用者可以使用這個 index number 來告知 **rfkill** 停用或啟用某一項裝置。

```
rfkill block 0
```

這會停用系統上的第一個支援 RFKill 的裝置。

使用者也可以使用 **rfkill** 來停用某些種類的裝置，或所有支援 RFKill 的裝置。舉例來說：

```
rfkill block wifi
```

這會停用系統上的所有 Wi-Fi 裝置。要停用支援 RFKill 的裝置，請執行：

```
rfkill block all
```

要重新啟用裝置，請執行 **rfkill unblock** 而不是 **rfkill block**。要取得 **rfkill** 可以停用的裝置類別之完整清單，請執行 **rfkill help**

3.13. 使用者空間的最佳化

降低系統硬體的工作量，是節省電源的基石。因此，雖然 章 3, *核心架構與機制* 所描述的改變允許系統在多種節能省電的狀態下運行，但使用者空間（user space）所執行的應用程式可能會對系統硬體進行不必要的存取，讓硬體無法進入這些省電狀態。Red Hat Enterprise Linux 6 的開發過程中，在以下領域加入了稽核程式，以減少不必要的硬體需求：

更低的喚醒需求

Red Hat Enterprise Linux 6 使用了「*無計時 kernel*」（Tickless Kernel，詳情請參閱 節 3.6, “無計時 kernel”），能讓 CPU 處於深層的閒置狀態。然而，*timer tick*（計時器的計時功能）並不是喚醒 CPU 的唯一理由，應用程式的函數呼叫也可能會讓 CPU 無法進入（或保持）閒置狀態。我們降低了超過 50 個應用程式的不必要函數呼叫。

降低儲存裝置與網路的 I/O

對於儲存裝置與網路介面的 I/O（輸入與輸出，Input/Output）會迫使裝置消耗電力。有些儲存裝置與網路裝置可以在閒置時降低電力狀態（例如 ALPM 或 ASPM），但 I/O 會讓裝置無法進入（或保持）閒置狀態，或讓非使用中的硬碟仍然轉個不停。之前幾個應用程式會對儲存裝置發出過量、或沒有必要的需求，這些需求已經被降至最少，尤其以讓硬碟轉個不停的應用程式為最。

稽核 initscript

不管需不需要都會啟動的服務，很有可能會浪費系統資源。請盡可能地把服務的預設值設為「**off**」（關閉）或「**on demand**」（視需求啟動）。舉例來說，之前不管系統有沒有藍牙硬體，**BlueZ** 服務都會在開機時，自動啟動藍牙功能。現在 **BlueZ** initscript 會在啟動藍牙服務前，先檢查藍牙硬體是否存在。

章 4. 使用的範例

本章描述了兩種使用範例，藉以描述本指南所使用到的分析與配置方法。第一個範例是典型的伺服器，第二個則是筆記型電腦。

4.1. 範例 – 伺服器

基本上，Red Hat Enterprise Linux 6 支援現今所有的標準伺服器。首要考量是伺服器的工作量要用於何種用途。根據這項資訊，使用者就可以決定要節省哪個元件的電力消耗。

不管是哪一種伺服器，圖形效能通常都不是必要的。因此，您可以開啟 GPU 的節能功能。

網站伺服器

網站伺服器仰賴網路與磁碟 I/O。只依靠 100 Mbit/s 的網路速度應該就足夠了。如果這台伺服器提供的多半是靜態網頁，那麼 CPU 效能就不甚重要。因此電源管理的選項就包括了：

- **tuned** 不使用磁碟或網路外掛程式。
- 開啟 ALPM。
- 開啟 **ondemand** 調速程式。
- 把網路卡的速度上限設為 100 Mbit/s。

運算伺服器

運算伺服器非常需要 CPU 的運算能力。節能的選項包括：

- 根據工作內容與資料儲存的地方而定，使用 **tuned** 的磁碟或網路外掛程式；或在批次處理的系統上，完全開啟 **tuned**。
- 根據使用率，使用 **performance** 調速程式。

郵件伺服器

郵件伺服器需要磁碟 I/O 與 CPU 的運算能力。電源管理的選項包括：

- 開啟 **ondemand** 調速程式，因為 CPU 運算能力差上幾個百分點無關宏旨。
- **tuned** 不使用磁碟或網路外掛程式。
- 因為郵件多半是由內部網路的電腦所存取，可以獲益於 1 Gbit/s 或 10 Gbit/s 的網路連線，所以網路速度不應該設限。

檔案伺服器

檔案伺服器跟郵件伺服器的需求類似，但依據所使用的通訊協定，可能更需要 CPU 效能。通常 Samba 伺服器會比 NFS 更需要 CPU 資源；而 NFS 又大於 iSCSI。但即使如此，您還是應該使用 **ondemand** 調速程式。

目錄伺服器

通常目錄伺服器對磁碟 I/O 的需求較低，尤其是記憶體足夠時。網路延遲時間的因素會高於 I/O。您不妨使用較低的網路速度，但延遲時間較低的網路；請針對您自己的網路實際測試過，以達最高效能。

4.2. 範例 – 筆記型電腦

另一個電源管理與節能功能可以發揮所長的地方，是筆記型電腦。一般來說，筆記型電腦的耗電量在設計時就已經比工作站、伺服器還要低。在電池模式下，任何節能方式都可以從電池再榨出幾分鐘的電力。雖然本節專注在筆電的電池模式，但您還是可以把這裡的一些微調技巧用在連接 AC 電源上。

通常在筆電上節省單一元件的電力，會比在工作站上這樣作更有效。舉例來說，1 Gbit/s 的網路介面如果只以 100 Mbits/s 的速度執行，可以省下大約 3-4 瓦的電力。對於整體消耗 400 瓦的伺服器來說，這大約省下了 1% 的電力；對於整體消耗 40 瓦的筆記型電腦來說，單是此單一元件所省下的電力就佔了總額的 10%。

一般筆記型電腦的電源最佳化方式有：

- 在系統 BIOS 裡，停用所有不使用的硬體。舉例來說，平行連接埠或序列埠、讀卡機、攝影機、WiFi 無線網路、藍牙等等。
- 在較暗的環境裡使用筆電時，螢幕不需要非常亮也可以閱讀無礙，所以不妨把螢幕的亮度設暗一點。您可以在 GNOME 桌面環境下，透過 **系統+偏好設定 → 電源管理程式**，或 KDE 桌面的 **Kickoff Application Launcher+電腦+系統設定+進階 → 電源管理**，抑或使用命令列執行 **gnome-power-manager** 或 **xbacklight**，又或是筆記型電腦的功能鍵來達到此一效果。
- 使用 **tuned-adm** 應用程式的 **laptop-battery-powersave** 來啟用全套的節能機制。請注意硬碟與網路卡的效能與延遲時間，有決定性的影響。

除此之外，您也可以對多種系統設定進行微調：

- 使用 **ondemand** 調速程式（預設上，Red Hat Enterprise Linux 6 會啟用這個功能）
- 啟用筆電模式（**laptop-battery-powersave** 的一部分）：

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

- 增加磁碟的 flush 時間（**laptop-battery-powersave** 的一部分）：

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

- 停用 nmi watchdog（**laptop-battery-powersave** 的一部分）：

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

- 啟用 AC97 音效的節能模式（預設上，Red Hat Enterprise Linux 6 會啟用這個功能）：

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

- 啟用多核心的節能功能（**laptop-battery-powersave** 的一部份）：

```
echo 1 > /sys/devices/system/cpu/sched_mc_power_savings
```

- 啟用 USB 的自動暫停功能：

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

  請注意：不是所有的 USB 裝置都支援此功能。

- 啟用 ALPM 的最低耗電設定（**laptop-battery-powersave** 的一部分）：

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

- 使用 relatime 參數來掛載檔案系統（預設上 Red Hat Enterprise Linux 6 會啟用此功能）：

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

- 啟用硬碟的最佳節能模式（**laptop-battery-powersave** 的一部分）

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

- 停用光碟機的輪詢功能（**laptop-battery-powersave** 的一部分）：

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

- 把螢幕亮度降至 **50** 以下，例如：

```
xbacklight -set 50
```

- 啟用螢幕閒置時的 DPMS 功能：

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

- 降低 Wi-Fi 的電力（**laptop-battery-powersave** 的一部分）：

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

- 停用 Wi-Fi：

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

- 把有線網路的速度設為 100 Mbit/s（**laptop-battery-powersave** 的一部分）：

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

附錄 A. 給程式設計師的提示

每本程式設計教科書都會提到記憶體分配的問題，以及一些函數的效能問題。因此在您撰寫軟體時，也請注意軟體執行時會增加的系統電力消耗量。雖然這考量並不及於每一行程式，但您還是可以在效能瓶頸之處多所著墨。

通常會出問題的技巧包括：

- 使用執行續。
- 不必要或沒效率地喚醒 CPU。如果真的必要喚醒 CPU，請一次（從 race 到 idle）、儘快完成。
- 不必要地使用 **[f]sync()** 函數。
- 不必要地啟用輪詢（polling）功能，或使用短暫、一定頻率的 timeout。（請只對事件做出回應。）
- 沒效率地使用喚醒功能。
- 沒效率地存取磁碟。請使用較大的緩衝區空間，以避免過度頻繁存取硬碟。一次寫入大空間的磁區。
- 沒效率地使用計時器。可能的話，請跨應用程式（甚至跨系統）使用計時器。
- 過度頻繁的 I/O、消耗電力、或使用記憶體（包括記憶體逸漏）
- 進行不必要的運算。

以下章節會更進一步檢視這些內容。

A.1. 使用執行續

一般來說，使用執行續會讓應用程式更好、更快；但這不是每次都靈光。

Python

Python 使用了 Global Lock Interpreter[1]，因此只有在大規模的 I/O 運作時，執行續才有其效用。Python 的 **Unladen-swallow** [2] 較快，更適用於程式碼最佳化。

Perl

Perl 的執行續一開始是給沒有 fork 功能的系統（例如 32 位元的 Windows 作業系統）之應用程式使用的。在 Perl 執行續裡，資料會複製給每個執行續（Copy On Write）。預設上，資料不會共享，因為使用者應該可以定義資料共享的等級。為了資料共享，**threads::shared** 模組已經包括在 Perl 裡面。然而，資料不但會被複製（Copy On Write），這模組還會為資料建立相關的變數，耗費更多時間，變得更慢。[3]

C

C 的執行續會共享同樣的記憶體，每個執行續都有自己的堆疊，kernel 不需要建立新的檔案描述或分配新的記憶體空間。C 可以善用多 CPU，執行多執行續。因此，要讓執行續的效能最大化，請使用較低階的語言，如 C 或 C++。如果您使用的是描述式語言，請考慮使用與 C 相關的語言。並使用剖析程式來找出程式中效能不彰的地方。[4]

A.2. 喚醒功能

許多應用程式會掃描配置檔，看有沒有變化。在許多情形下，這掃描會在固定的時間間隔下進行，例如每

分鐘執行一次。這可能會造成問題，因為它會喚醒已經降速的硬碟。最好的方法是找到最適合的時間間隔、更好的檢查機制，或使用 **inotify** 來檢查變更並重新啟動事件。**inotify** 可以檢查檔案或目錄的多種變化。

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();
}
...
```

這方法的好處是可以進行多樣化的檢查。

主要的限制是一台系統上只會有限定的觀察數量。這數量可以從 **/proc/sys/fs/inotify/max_user_watches** 得知；雖然這個值可以改變，但不建議您這樣作。同時，如果 **inotify** 失效的話，程式碼就會回到不同檢查方式，這多半意味著原始碼的 **#if #define** 會常常發生。

欲知 **inotify** 的詳情，請參閱 inotify 的 man page。

A.3. FSYNC

通常 **Fsync** 是 I/O 密集的運作，但這並不完全正確。

Firefox 會在使用者按下網頁連結，以連上新的頁面時，呼叫 **sqlite** 函式庫。**Sqlite** 會呼叫 **fsync**，同時因為檔案系統的設定（主要是 **ext3** 與資料的排序模式），在什麼都沒發生時，會產生很長的延遲時間。如果同時間有另一個程序正在複製大檔案時，這延遲時間會多達 30 秒。

然而在其它不使用 **fsync** 的情形下，轉換到 **ext4** 檔案系統上問題就會接踵而來。Ext3 的設定是資料順序模式，每幾秒鐘就會清空記憶體，寫入磁碟去。但在 **ext4** 與 **laptop_mode**（筆電模式）下，儲存的間隔較長，系統無預期關機時，資料就會流失。雖然現在 **ext4** 已經更新，但撰寫應用程式時，還是要注意，並適當地使用 **fsync**。

以下這個簡單的範例會讀、寫一個配置檔，顯示了如何備份檔案，以及資料是怎麼流失的：

```
/* 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);
```

較好的方法是：

```
/* 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

附錄 B. 修訂記錄

修訂 1.0-35.2 **Mon Apr 20 2015** **Terry Chuang**
翻譯完成

修訂 1.0-35.1 **Mon Apr 20 2015** **Terry Chuang**
讓翻譯檔案與 XML 來源 1.0-35 同步

修訂 1.0-35 **Fri Oct 10 2014** **Yoana Ruseva**
6.6 GA 發行版本。

修訂 1.0-34 **Fri Aug 8 2014** **Yoana Ruseva**
6.6 Beta 發行版本。

修訂 1.0-33 **Wed Sep 25 2013** **Yoana Ruseva**
6.5 GA 發行版本

修訂 1.0-25 **Tue Feb 19 2013** **Jack Reed**
6.4 GA 第二發行版本

修訂 1.0-22 **Mon Feb 18 2013** **Jack Reed**
6.4 GA 發行版本

修訂 1.0-21 **Wed Nov 7 2012** **Jack Reed**
修正小錯誤

修訂 1.0-20 **Wed Oct 31 2012** **Jack Reed**
移除兩個警告標題

修訂 1.0-19 **Wed Oct 31 2012** **Jack Reed**
修正 cpupower 替代方案中的小錯誤

修訂 1.0-18 **Tue Oct 30 2012** **Jack Reed**
新增 cpupower 相關的警告，並更新 --policy 選項的功能

修訂 1.0-15 **Thu Oct 18 2012** **Jack Reed**
新增 CPU 監控與 CPU 省電政策二節，並更新一些指令以反應新的 cpupower 工具 - BZ#860874

修訂 1.0-14 **Fri Feb 10 2012** **Jack Reed**
額外修正 ASPM 參數的格式 - BZ#732859
新增 tuned 設定檔 - BZ#701924

修訂 1.0-12 **Fri Dec 16 2011** **Jack Reed**
進一步修正 ASPM 參數格式 - BZ#732859

修訂 1.0-11 **Thu Dec 15 2011** **Jack Reed**
移除 fsync 附錄中的錯誤連結 - BZ#706928

修訂 1.0-10 **Thu Dec 15 2011** **Jack Reed**
進一步編輯 fsync 配置檔的範例 - BZ#706928

修訂 1.0-9 **Wed Dec 14 2011** **Jack Reed**
移除連至開機參數的錯誤參照 - BZ#692859

修訂 1.0-8 **Mon Dec 12 2011** **Jack Reed**

修正打字錯誤 - BZ#722798
編輯 fsync 一節中的範例設定檔 - BZ#706928
釐清電池參照資料的可用性 - BZ#740794
修正 ASPM 政策參數的格式 - BZ#732859

修訂 1.0-7 **Thu Sep 29 2011** **Jack Reed**
從 GNOME 電源管理一節中，移除「使用電池電源」分頁的資訊 - BZ#740794

修訂 1.0-6 **Tue May 24 2011** **Rüdiger Landmann**
重新建立

修訂 1.0-2 **Fri Oct 22 2010** **Rüdiger Landmann**
修正本文中的小錯誤

修訂 1.0-1 **Thu Oct 7 2010** **Rüdiger Landmann**
移除「草稿」的標記

修訂 1.0-0 **Thu Oct 7 2010** **Rüdiger Landmann**
GA 發行版