# Red Hat Enterprise Linux 7
# 效能微調指南

---

最佳化 Red Hat Enterprise Linux 7 子系統生產能力

作者：Laura Bailey
翻譯、校對：陳郁棠 － Yutang (Prima) Chen
翻譯、校對：羅詩婷 － Shih-Ting (Judy) Lo
校對、責任編輯：鄭中 － Chester Cheng

# Red Hat Enterprise Linux 7 效能微調指南

## 最佳化 Red Hat Enterprise Linux 7 子系統生產能力

作者：Laura Bailey
紅帽公司　出版部
lbailey@redhat.com

翻譯、校對：陳郁棠 － Yutang (Prima) Chen
澳大利亞昆士蘭大學　筆譯暨口譯研究所
prima0727@hotmail.com

翻譯、校對：羅詩婷 － Shih-Ting (Judy) Lo
澳大利亞昆士蘭大學　筆譯暨口譯研究所
outstanding.lo@gmail.com

校對、責任編輯：鄭中 － Chester Cheng
紅帽全球服務部 & 澳洲昆士蘭大學筆譯暨口譯研究所
ccheng@redhat.com, uqcchun1@uq.edu.au

## 法律聲明



## 摘要

Red Hat Enterprise Linux 7《效能微調指南 》說明如何最佳化 Red Hat Enterprise Linux 7 的效能。本文件也記載有關 Red Hat Enterprise Linux 7 效能的升級。 《效能微調指南》僅提供實地驗證過以及經過證實的步驟。儘管如此，所有潛在配置應該在運用於產品系統之前，設置好並於試驗環境測試。 同時，建議您在微調之前備份所有資料和配置設定。

# 內容目錄

# 章 1. Red Hat Enterprise Linux 7 效能特色

請閱讀本章節了解 Red Hat Enterprise Linux 7 所包含有關效能更動的簡要概述。

## 1.1. 7.1 的新增項目

- 目前的 kernel 閒置平衡機制對於在同一個 CPU 的最近排定任務比較不易造成延誤，也減少了使用閒置平衡時的處理延遲。
- **swapon** 指令有新增的 ***--discard*** 參數，讓管理員在 swap 時能夠選擇使用哪一個捨棄原則。這不僅提供更好的彈性，也能改善固態硬碟的效能。預設情況下，**swapon** 運作時，***--discard*** 參數會捨棄整個 swap 區域，並且在重新使用前捨棄自由 swap 分頁。更多詳情，請參閱 **swapon** 的 man page，指令為：**man swapon**。
- **tuned** 設定檔現在能夠透過設定指定 ***cmdline*** 參數以及一系列在已微調的設定檔的 kernel 參數方式，達成更改 kernel 命令列參數。任何由 ***cmdline*** 指定的 kernel 參數會在重新啓動後生效。

## 1.2. 7.0 的新增項目

- 本指南已為 Red Hat Enterprise Linux 7 完整地重寫與重新調整架構。
- Red Hat Enterprise Linux 7 中的 **deadline** 取代了 **cfq** 做為預設 I/O 排程器。這項改變在大部分情況下提供了更好的效能。
- XFS 檔案系統取代了 ext4 做為預設檔案系統，並支援 500 TB 的最大檔案系統大小，和 8 EB 的最大檔案位移（疏鬆檔案）。為了讓讀者更清楚了解，本指南對 XFS 已經做出更新。
- ext4 檔案系統可支援最大 50 TB 的檔案系統，以及最大 16 TB 的檔案大小。微調建議已相應更新。此外，現在 ext2 和 ext3 檔案系統的支援是由 ext4 驅動程式提供。
- btrfs 檔案系統現在列為技術預覽。
- Red Hat Enterprise Linux 7 包含一些 GFS2 的微小效能改良。
- **Tuna** 升級至能夠包含配置檔支援以及增添和儲存 **tuned** 檔案。此升級版本使用基於事件的採樣去消耗更少的處理器資源。圖形化版本也已更新，使用者現在能夠進行即時監測。**Tuna** 記錄在〈節 2.4, “Tuna”〉、〈節 3.3.8, “用 Tuna 來配置 CPU、執行緒，以及插斷親和性”〉和〈節 A.2, “Tuna”〉中。
- **tuned** 的預設檔案文件是 **throughput-performance**。這取代了被移除的 **enterprise-storage** 檔案，也新增了許多網路和虛擬化的新檔案。此外，**tuned** 提供 shell script callout 以及 ***includes*** 功能。
- **tuned-adm**工具提供 **recommend** 子指令能夠推薦一項適合您系統的的微調設定檔。在安裝的同時，此工具也為您的系統設定了預設設定檔，可以用於返回預設設定檔。
- Red Hat Enterprise Linux 7 支援自動 NUMA 負載平衡。kernel 會自動偵測哪一個是正在使用的記憶體分頁執行續，並且將執行續和其記憶體分類為或分類至 NUMA 節點。kernel 重新規劃執行續與轉移記憶體以平衡系統，進而達到最佳 NUMA 對齊方式和效能。
- 啓用檔案系統屏障導致的效能損失是可以忽略不記的（少於百分之三）。因此， **tuned** 檔案不再停用檔案系統屏障。
- OProfile 以新的 **operf** 工具為程式碼剖析新增支援，而此剖析是以 Linux 效能事件子系統作為根據。這項新工具取代 **opcontrol** daemon，能夠用於收集數據。
- 在您的系統上，控制群組仍可以分配資源至特定處理續群組。更多關於 Red Hat Enterprise Linux 7 執行情況的資訊，詳見《*Red Hat Enterprise Linux 7 資源管理指引*》，可自

http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 取得。

# 章 2. 效能監控工具

本章節簡約地描述一些可以用於 Red Hat Enterprise Linux 7 的效能監控和設定工具。在可能情況下，本章節將指引讀者更多有關如何使用此工具的資訊，以及此工具能夠應用於實際生活中的例子。

以下知識庫文章提供適用於 Red Hat Enterprise Linux 的更完整效能監控工具列表：https://access.redhat.com/site/solutions/173863。

## 2.1. /proc

**/proc**「檔案系統」是一個包含檔案層級的目錄，代表 Linux kernel 目前的狀態。此「檔案系統」讓使用者和應用程式能夠以 kernel 的角度檢視系統。

**/proc** 目錄也包含關於系統硬體以及目前正在運行的程序之訊息。多數在 **/proc** 檔案系統的檔案僅供參閱，但有一些檔案（主要是在 / proc/sys 的檔案）可以由使用者和應用程式操作，進而傳達配置變更訊息至 kernel。

有關檢閱和編輯 **/proc** 目錄檔案的更多資訊，請參閱《Red Hat Enterprise Linux 7 系統管理員參考指南》，網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

## 2.2. GNOME 系統監視器

GNOME 桌面環境包含 graphical tool，即系統監視器，協助您監控和修改系統行為。系統監視器顯示基本系統資訊，以及允許您監控系統流程和資源，或者檔案系統使用量。

系統監視器有四個索引標籤，每一個都顯示關於系統的不同訊息。

**系統**

此索引標籤顯示關於系統硬體和軟體的基本訊息。

**程序**

此索引標籤顯示有關使用中的流程，和這些流程之間關係的詳細資訊。顯示的流程可以經由篩選，使特定流程更容易尋得。此索引標籤亦能讓您執行一些顯示於流程的動作，例如：啓動、停止、刪除和更改優先順序。

**資源**

此索引標籤顯示目前 CPU 的時間使用量、記憶體和 swap 空間使用量，以及網路使用量。

**檔案系統**

此索引標籤列出所有掛載的檔案系統，並且提供一些關於各個檔案系統的基本訊息，例如：檔案系統類別、掛載點和記憶體使用量。

如需啓動系統監視器，請按 Super 鍵進入活動概覽，輸入「*System Monitor*」，然後按 Enter。

更多關於系統監視器的訊息，詳見應用程式中的求助選單或至《Red Hat Enterprise Linux 7 *系統管理員指南*》，網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

## 2.3. Performance Co-Pilot

Red Hat Enterprise Linux 7 支援 Performance Co-Pilot（PCP）工具套件、服務，和擷取、儲存與分析系統層級效能度量的程式庫。PCP 的輕質與和分散式結構格外適用於複雜系統的集中分析。效能指標能夠在使用 Python、Perl、C++ 和 C 介面時添加。分析工具可以直接使用用戶端 API（Python、 C++、 C），且 rich web 應用程式能夠查看正在使用 JSON 介面的所有效能數據。

*pcp* 套件提供命令列工具以及基礎功能。圖形化工具亦需要「*pcp-gui*」套件。

Red Hat 客戶入口網站有一些實用的文章，能夠幫助您了解 PCP。文章的索引的網址為：https://access.redhat.com/articles/1145953。

此外，*pcp-doc* 套件提供完整的文件。預設上，此文件是安裝於 **/usr/share/doc/pcp-doc**。PCP 也為每個工具提供 man page；欲閱覽指定工具的 man page，請在命令列輸入 **man *toolname***。

## 2.4. Tuna

Tuna 會調整配置細節，比如排程器原則、執行續優先順序，以及 CPU 和同質中斷。*tuna* 套件提供命令列工具與同等功能的圖形化介面。

〈節 3.3.8, “用 Tuna 來配置 CPU、執行緒，以及插斷親和性”〉 描述如何在命令列使用 Tuna 來配置您的系統。有關如何使用 Tuna 的詳情，請參閱〈節 A.2, “Tuna”〉或 man page：

```
$ man tuna
```

## 2.5. 內建命令列工具

Red Hat Enterprise Linux 7 提供許多工具，能夠從命令列監視系統，並讓您得以在 run level 5 外監視您的系統。本章節簡要談論每個工具，並且提供更多關於每個工具應該用於何處的連結，以及如何使用這些工具。

### 2.5.1. top

由 *procps-ng* 提供的 top 工具套件會在運行中的系統產生一個處理程序的動態視圖。top 工具套件可以呈現各式各樣的訊息，包括系統摘要以及一系列目前由 Linux kernel 管理的任務。此工具套件能有限度地操作處理程序，以及使配置更動在系統重啓後持續存在。

預設情況下，顯示的處理程序是根據 CPU 使用量比率做排序，因此您可以輕易地看見消耗最多資源的處理程序。top 所顯示和操作的訊息皆為高度可配置的，讓您能根據所需，專注於不同的使用量統計資料。

更多有關使用 top 的詳細資訊，請參閱 man page：

```
$ man top
```

### 2.5.2. ps

由 *procps-ng* 提供的 ps 工具展現一個使用中程序的選取群組情況。在預設情況下，被檢查的群組受限於目前使用者所擁有的處理序，以及 ps 相關的運行終端。

相較於 top ， ps 能夠提供關於處理序的更詳細訊息。然而在預設情況下，是由識別子命令 ps 提供資料的單一快照。

更多有關使用 ps 的詳情，請參閱 man page：

```
$ man ps
```

### 2.5.3. 虛擬記憶體統計資料（vmstat）

虛擬記憶體統計資料工具（vmstat - Virtual Memory Statistics tool）提供您系統程序、記憶體、分頁、區塊輸出/入、中斷和 CPU 活動的即時報告。vmstat 可讓您設定取樣間隔，以便您接近即時地觀察系統活動。

vmstat 是由 *procps-ng* 套件提供。更多有關使用 vmstat 的詳情，請參閱 man page：

```
$ man vmstat
```

### 2.5.4. 系統活動報告器

系統活動報告（sar）收集和報告有關今天至此刻的系統活動訊息。預設輸出是從一天的開始（根據系統時鐘的 00:00:00），以十分鐘為間隔，顯示當天的 CPU 使用量。

您也可以使用 **-i** 選項設定間隔時間，以秒為單位。比如：**sar -i 60** 告訴 sar 每分鐘檢查 CPU 使用量。

若使用者欲手動操作以建立系統活動定期報告，sar 能與 top 一起使用，是一個實用的替代選項。sar 是由 *sysstat* 套件提供。更多有關使用 sar 的詳情，請參閱 man page：

```
$ man sar
```

## 2.6. tuned 與 tuned-adm

tuned 是微調 daemon，透過設定微調設定檔調整作業系統，使效能在特定工作負載下執行地更好。tuned-adm 是一個命令列工具，讓使用者在不同微調設定檔之間切換。

一些預設檔案提供一般使用案例，但是 tuned-adm 也讓您定義客戶設定檔，這個設定檔是基於預設設定檔或者重新定義。Red Hat Enterprise Linux 7 中的預設設定檔是「**throughput-performance**」。

配有 tuned-adm 的檔案被分為兩種：節能檔案和效能提升檔案。效能提升檔案包含著重於以下的檔案：

- 儲存量和網路的低延遲
- 儲存量和網路的高處理量
- 虛擬機效能
- 虛擬主機效能

有關如何啓用 tuned 的詳細資訊，詳見〈節 A.5, “tuned”〉。

有關附有 tuned-adm 的效能提升檔案的詳細資訊，詳見〈節 A.6, “tuned-adm”〉。

有關附有 tuned-adm 的節能檔案詳細資訊，詳見《 Red Hat Enterprise Linux 7 能源管理指南 》，網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

有關使用 tuned 和 tuned-adm 的詳細資訊，詳見各 man page：

```
$ man tuned
```

```
$ man tuned-adm
```

## 2.7. perf

perf 工具使用硬體效能計數器和 kernel 追蹤點，追蹤您系統上其他命令和應用程式的影響。各種 perf 子指令會顯示和紀錄命令效能事件的數據，以及對已紀錄的資料進行分析和報告。

有關 perf 和其子指令的詳細資訊，詳見〈節 A.7, “perf”〉。

此外，《Red Hat Enterprise Linux 7 *開發者指南*》中有更多資訊，網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

## 2.8. turbostat

turbostat 是由「*kernel-tools*」套件提供。turbostat 對處理器拓撲、頻率、閒置電源狀態數據、溫度和 Intel® 64 處理器上的電源使用量提出報告 。

turbostat 有助於辨識電源使用量或閒置時間方面低效率的伺服器，也有助於辨識發生在系統上的系統管理中斷（SMI）。此外，turbostat 也能用於核對電源管理微調的效果。

turbostat 的運作需要 root 權限，也需要以下的處理器支援：

- 恆定時間標示計數器
- APERF 特定型號註冊碼
- MPERF 特定型號註冊碼

更多有關 turbostat 訊息輸出以及如何閱讀 turbostat 的詳細資訊，請參閱〈節 A.11, “turbostat”〉。

更多有關 turbostat 的訊息，請參閱 man page：

```
$ man turbostat
```

## 2.9. iostat

「**iostat**」工具是由「*sysstat*」套件提供。iostat 對系統輸入 / 輸出裝置負載進行監測並提出報告，以幫助系統管理員決定有關如何平衡實體硬碟之間輸入 / 輸出的負載。iostat 從最近一次運作或者重新啓動以來，對處理器或裝置運用提出報告。透過使用 iostat man page 中定義的參數，您能專注於這些報告對特定裝置的輸出。

```
$ man iostat
```

## 2.10. irqbalance

「**irqbalance**」是處理器分發硬體中斷的命令列工具，用以改善系統效能。有關「**irqbalance**」的詳細資訊，請參閱〈節 A.1, “irqbalance”〉或 man page：

```
$ man irqbalance
```

## 2.11. ss

ss 是印出有關通訊端統計資訊的命令列工具程式，使系統管理員能夠隨時間評估裝置效能。在預設情況下，ss 列出已連線的開放非聽取性 TCP 通訊端，也提供一些實用的選項，以協助系統管理員篩選有關特定通訊端的統計。

Red Hat 建議於 Red Hat Enterprise Linux 7 中使用 ss 而非 netstat。

「**ss -tmpie**」是一個常見用法，其顯示有關 TCP 通訊端、記憶體使用量和使用通訊端的程序的詳細資訊（包含內部資訊）。

ss 由「*iproute*」套件提供。更多資訊請參閱 man page：

```
$ man ss
```

## 2.12. numastat

「**numastat**」工具以每一 NUMA 節點為基準，對程序與作業系統顯示記憶體數據。

在預設情況下，「**numastat**」顯示自 kernel 記憶體配置器的每一節點 NUMA 點擊與遺漏系統數據。最佳效能是由高「**numa_hit**」數值和低「**numa_miss**」數值顯示。「**numastat**」也提供一些命令列選項，這些選項能夠顯示系統和程序記憶體如何於系統中 NUMA 節點分發。

此工具有助於相互參照每一個 CPU「**top**」輸出與每一個節點「**numastat**」輸出，以確認程序執行緒運作的節點與記憶體所分配至的節點相同。

「**Numastat**」是由「*numactl*」套件提供。有關如何使用 numastat 的詳細資訊，詳見〈節 A.12, “numastat”〉。更多有關 numastat 的資訊，請參閱 man page：

```
$ man numastat
```

## 2.13. numad

numad 是一自動 NUMA 親和性管理 （affinity management）daemon。為了動態改善 NUMA 資源分配和管理（乃至系統效能），numad 於系統內監測 NUMA 拓撲和資源使用量。根據系統工作負載，numad 能夠改善高達百分之五十的效能基準。此外，numad 也提供能夠透過各種工作管理系統取得的設置前建議服務，輔助 CPU 與記憶體資源的最初結合的過程。

numad 監測器定期取得「**/proc** 」檔案系統中資訊，使系統資源能夠以每一節點基準使用。numad 會試圖維持一個特定的資源使用量水平，並在必要時透過移動 NUMA 節點之間的處理序以重新平衡資源分配。numad 透過定位與隔離系統的 NUMA 節點子集上重要的處理序，達到最佳 NUMA 效能。

numad 主要對消耗大量資源，和包含在整體系統資源子集內的長期運行處理序有益。numad 也有益於需要消耗多個 NUMA 節點資源的應用程式。然而，numad 的益處會隨著系統資源消耗百分比的增加而降低。

當處理序僅運行數分鐘或者不消耗很多資源時，numad 不大可能對效能進行改善。持續與不可預測的記憶體存取模式的系統（例如：大量的內存資料庫），也不大可能受益於使用 numad。

更多有關使用 numad 的資訊，詳見〈節 3.3.5, “用 numad 自動管理 NUMA 親和性”〉或〈節 A.14, “numad”〉或參閱 man page：

```
$ man numad
```

## 2.14. SystemTap

SystemTap 是一個追蹤和探測工具，讓您能夠詳細地監測與分析運作中系統活動，特別是 kernel 活動。SystemTap 所提供的訊息與 top、ps、netstat 以及 iostat 工具的輸出訊息相似，但額外包含篩選和分析已收集資料的選項。

SystemTap 提供更深入、更精確的系統活動與應用程式行為分析，使您能夠找出系統與應用程式瓶頸。

更多有關 SystemTap 的詳細訊息，請參閱《 Red Hat Enterprise Linux 7 *SystemTap 初學者指南* 》與《 Red Hat Enterprise Linux 7 *SystemTap TapSet 參考指南* 》。兩本書的網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

## 2.15. OProfile

OProfile 是一個全系統效能監測工具。OProfile 使用處理器的專用效能監測硬體，擷取有關 kernel 和系統可執行程序，以決定特定事件的頻率，例如：記憶體受參照中接收到的第二級快存請求數量，與硬體請求數量。OProfile 也能夠用於決定處理器使用量，與決定哪一應用程式和服務最常使用。

然而，OProfile 有以下限制：

- 效能監測樣本可能不精確，因為處理器可能不依順序執行指令。樣本能夠從附近的指令紀錄下來，而非觸發中斷的指令。
- OProfile 預期程序會啓動與停止數次。藉此累積多個運作樣本。您或許需要清除源於先前運作的範例資料。
- OProfile 著重於辨識受 CPU 存取限制的處理序問題。因此， 當程序等待其他事件解除鎖定而進入睡眠狀態，OProfile 並不適用於辨識睡眠狀態中的程序。

更多有關 OProfile 的詳細資訊，請參閱〈節 A.15, “OProfile”〉或《 *Red Hat Enterprise Linux 7 系統管理員指南* 》，兩者的網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。您亦可參閱位於系統上的文件：「**/usr/share/doc/oprofile-*version***」。

## 2.16. Valgrind

Valgrind 提供數個偵測和程式剖析碼工具，協助改善您應用程式的效能。這些工具能夠偵測記憶體和執行續相關錯誤，以及堆積、堆疊和陣列溢位，使您容易定位和更正您應用程式程式碼中的錯誤。這些工具也剖析快存、堆積與分支預測，以辨識可能增加應用程式速度和減低記憶體使用量的因素。

當您正在執行應用程式，Valgrind 透過在綜合 CPU 運作應用程式和檢測應用程式程式碼，分析您的應用程式。Valgrind 接著印出評論，這些評論清楚辨識各個有關使用者指定檔案、檔案描述元或網路通訊端的應用程式執行程序。請注意，執行檢測程式碼所需時間比一般執行多四至五倍。

Valgrind 不用重新制定便能於您應用程式上使用。然而，因為 Valgrind 使用 debug 訊息找出您程式碼中的問題，如果您的應用程式和支援程式庫並不符合 debug 資訊，Red Hat 建議您重新制定，以包含這項訊息。

Valgrind 也與 GNU Project Debugger（gdb）整合，以改善 debug 效率。

Valgrind 與其子工具都有益於記憶體分析。有關使用 Valgrind 分析系統記憶體的詳細資訊，請參閱〈節 4.2.2, “用 Valgrind 設定應用程式記憶體使用量”〉。

有關 Valgrind 的詳細資訊，請參閱 《Red Hat Enterprise Linux 7 開發者指南》，能夠自 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/ 取得。

有關使用 Valgrind 的詳細資訊，詳閱 man page：

```
$ man valgrind
```

安裝完畢 valgrind 套件時，隨附文件能夠於「**/usr/share/doc/valgrind-*version***」中取得。

# 章 3. CPU

本章節將說明 CPU 硬碟的細節以及影響 Red Hat Enterprise Linux 7 效能的設定選項。〈節 3.1, “注意事項”〉針對與 CPU 相關且影響效能的因素作討論。〈節 3.2, “監視與診斷效能問題”〉教您如何使用 Red Hat Enterprise Linux 7 的工具來診斷與 CPU 有關的效能問題以及設定細節。〈節 3.3, “配置建議”〉討論在 Red Hat Enterprise Linux 7 中有哪些工具和策略可以用來解決因 CPU 引起的效能問題。

## 3.1. 注意事項

請詳讀以便了解以下因素如何影響系統以及應用程式的效能：

- 各處理器之間連接的方式，以及和記憶體這種相關資源連接的方式。
- 處理器如何排程執行緒執行指令。
- 處理器如何處理 Red Hat Enterprise Linux 7 的插斷現象。

### 3.1.1. 系統拓樸

在現代的電腦科學中，「*中央*」處理器的意思很模糊，因為現在大部分的系統都有多個處理器。這些處理器之間，如何連接以及如何與其他系統資源連接，會大大影響系統效能以及微調系統時該注意的事項，而這個相互連接的過程稱為系統「*拓樸*」。

現代電腦科學主要應用的拓樸有兩種：

**SMP 拓樸（對稱多處理器拓樸）**

SMP 拓樸讓各處理器得以在相同時間內存取記憶體。但是，存取共用記憶體本來就會強制存取所有 CPU 內序列化的記憶體，因此現在 SMP 系統的縮放限制普遍不被接受。基於這個原因，所有現代系統的伺服器幾乎都是 NUMA 機型。

**NUMA 拓樸（非一致性記憶體存取拓樸）**

與 SMP 拓樸相比，NUMA 拓樸是近期才發展出來的。NUMA 系統中，多個處理器集結在 CPU 插槽。每個 CPU 插槽皆有一個特定區域用來儲存記憶體，而存取該記憶體的處理器統稱為一個節點。

在同一節點上的處理器可以快速地連接到該節點的記憶體插槽，但是如果要連到其他節點的記憶體插槽，速度會比較慢。因此，取得遠端記憶體時效能會受到影響。

為避免影響效能，使用 NUMA 拓樸的效能相關程式應該取得的是與執行中的處理器同一個節點上的記憶體，並且盡量避免存取遠端記憶體。

當您微調使用 NUMA 拓樸的系統的程式效能，要考慮此程式執行的位置，還要考慮哪一個記憶體區塊離執行點最接近。

在使用 NUMA 拓樸的系統中，「**/sys**」檔案系統包含了關於處理器、記憶體和周邊裝置互連接方式的資訊。「**/sys/devices/system/cpu**」目錄包含系統內的處理器互相連接的方式。「**/sys/devices/system/cpu**」目錄則包含系統內 NUMA 節點的資訊以及這些節點之間的相對距離。

#### 3.1.1.1. 確定系統拓樸

有一些指令可以幫助您了解您所使用的系統拓樸。「**numactl --hardware**」這個指令介紹您的系統拓樸的概觀。

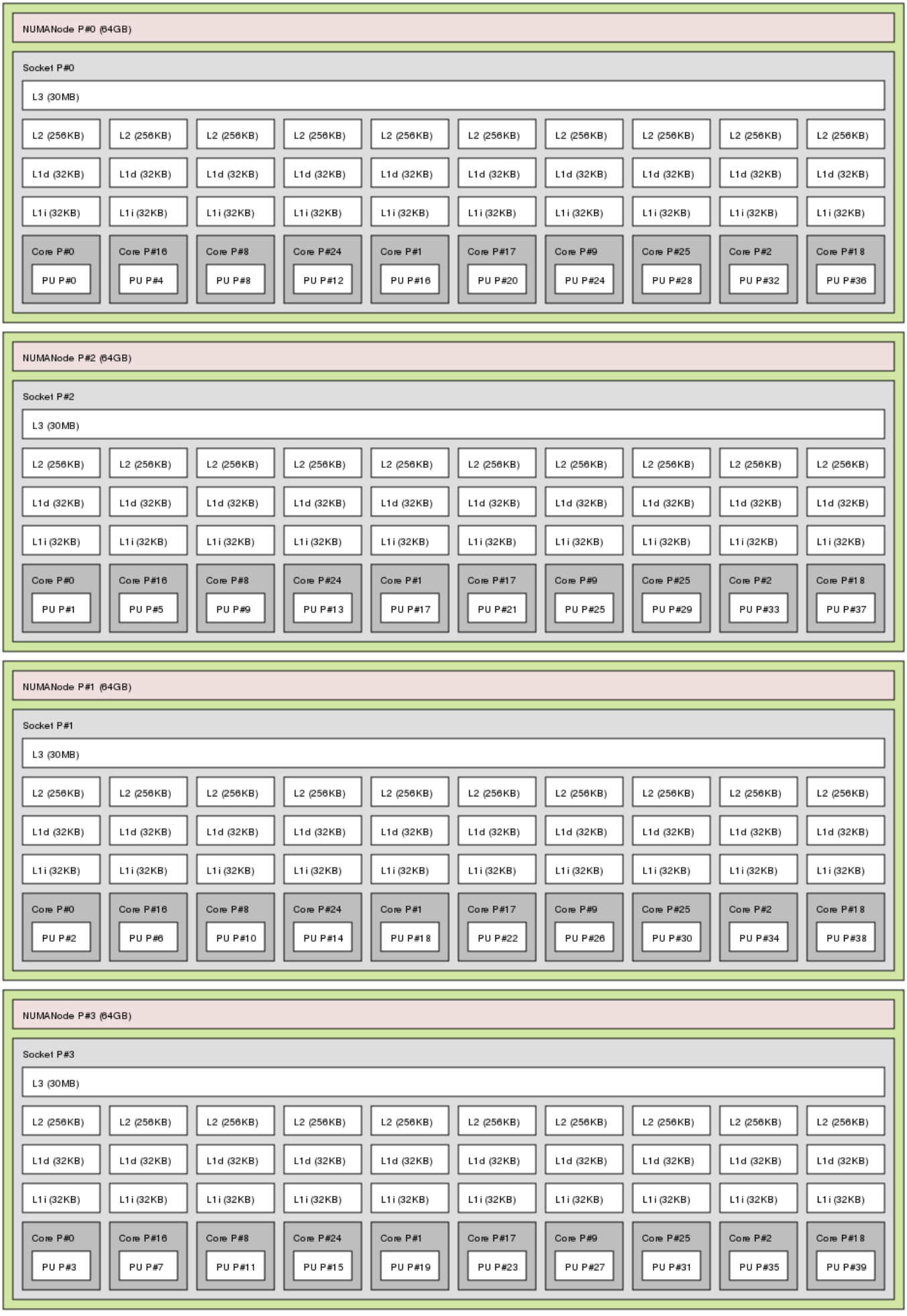
```
$ numactl --hardware
```

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

「**lscpu**」指令是由「*util-linux*」套件提供。此指令收集 CPU 架構的資訊，例如 CPU 的數目、執行緒、核心、CPU 插槽以及 NUMA 節點。

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

「**lstopo**」指令是由「*hwloc*」套件提供。這個指令以圖形方式呈現您的系統。「**lstopo-no-graphics**」指令提供詳細的文字輸出。


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

lstopo 指令輸出

### 3.1.2. 排程

Red Hat Enterprise Linux 裡最小的程序執行單元是「*執行緒*」。系統的排程器決定各處理器執行的執行緒，以及執行執行緒的時間。但是，由於排程器最主要的任務是要讓系統忙碌，所以它可能不會為了應用程式的效能而最佳排程執行緒。

舉例來說，假如一個裝有 NUMA 系統的應用程式在 A 節點上執行，而此時 B 節點的處理器可以被使用，排程器會把其它應用程式上的一個執行緒移到 B 節點，移到 B 節點的執行緒仍然需要取得 A 節點的記憶體。這會花比較久的時間，因為此執行緒現在於 B 節點執行，A 節點的記憶體對它而言屬於遠端的記憶體。與其等 B 節點的執行緒執行完畢，比較省時間的作法是保持原狀，等 A 節點的處理器執行完畢，然後執行原節點上的執行緒，因為它能取得本機的記憶體。

程式設計者或是管理者決定執行緒執行的位置，讓效能敏感的應用程式達到最高效能。欲了解執行緒的排程是否對效能相關程式有所幫助，請見〈節 3.3.6, “微調排程原則”〉

#### 3.1.2.1. kernel tick

在之前的 Red Hat Enterprise Linux 版本中，為了檢查有哪些未完成的工作，Linux kernel 會定期插斷每一個 CPU。kernel 依據檢查的結果排程處理程序以及負載平衡。這個固定的插斷被稱為 kernel「*tick*」。

無論核心有沒有工作要執行，tick 都會發生。意思是，閒置核心為了回應插斷，每隔一段時間就會被迫進入高功率狀態 ( 每秒高達一千次 )。這個情形讓系統無法有效率地使用最近幾代 x86 處理器當中的深層睡眠狀態。

Red Hat Enterprise Linux 6 和 7 將 kernel 預設為不打擾閒置 CPU。這讓系統處在低電源狀態，而這種設定被稱為 tickless kernel 。當系統只有少數核心在工作時，隨選插斷會取代定期插斷，CPU 可以因此省電，保持在低電源或是閒置狀態。

Red Hat Enterprise Linux 7 提供動態 tickless 選項（「**`nohz_full`**」），減少 kernel 干擾使用者空間的工作所造成的影響。這個選項可以在有「***`nohz_full`***」參數的核心中被使用。此選項一旦在核心被啓用，全部計時的活動皆會移到非延遲敏感的核心。這對高效能運算和即時運算的工作負載是很有用的，因為在使用者空間的工作對與 kernel tick 相關的微秒程度延遲特別敏感。

欲了解如何使用 Red Hat Enterprise Linux 7 動態 tickless 行為的方法，請見〈節 3.3.1, “設定 kernel tick 的時間”〉。

### 3.1.3. IRQ 處理

插斷要求（IRQ）是一個從硬體傳送到處理器的即時訊號。系統當中的每一個裝置都被指派至少一個 IRQ 數字來傳送特定插斷。當系統被插斷，接收插斷要求的處理器為了解決插斷要求，會立刻暫停執行當前應用程式的執行緒。

由於一般操作被插斷，這樣的高插斷率會嚴重影響系統效能。透過插斷親和性的設定或者分批傳送較低優先順序的插斷（「*聯合*」 一些插斷）來減少插斷的時間是可行的。

欲了解微調插斷要求的資訊請見 〈節 3.3.7, “設定插斷親和性”〉 或〈節 3.3.8, “用 Tuna 來配置 CPU、執行緒，以及插斷親和性”〉。 欲了解特定網路插斷的資訊請見〈章 6, *網路功能*〉。

## 3.2. 監視與診斷效能問題

Red Hat Enterprise Linux 7 提供了許多有用的工具，可以監視系統效能以及診斷跟處理器和設定有關的效能問題。本章節將介紹這些工具，還有如何用這些工具來監視或者診斷跟處理器相關的效能問題。

### 3.2.1. turbostat

「**turbostat**」 會在特定間隔列印計數器數據，幫助管理員識別伺服器的異常狀況，例如過度使用電源、無法進入深層睡眠狀態，或是不必要的系統管理插斷（SMI）。

「**turbostat**」工具是「*kernel-tools*」套件的一部份，用來支援 AMD64 和 Intel® 64 處理器的系統。它需要 root 權限才能執行，而且需要處理器支援不區分時間戳記計數器，以及 APERF 和 MPERF 型號的特定登錄。

使用範例，詳見 man page：

```
$ man turbostat
```

### 3.2.2. numastat

**重要**

此工具在 Red Hat Enterprise Linux 6 生命週期有持續的更新。雖然預設輸出與 Andi Kleen 寫的原始工具保持相容，供應 numastat 任何選項或參數都會明顯改變輸出的格式。

「**numastat**」工具顯示每一個 NUMA 節點的程序和作業系統的記憶體統計資料，此工具也會告訴系統管理員程序記憶體是否橫跨至整個系統，或者集中於特定節點。

用每一個處理器的「**top**」輸出交叉參考「**numastat**」輸出，這樣一來可以確認程序的執行緒在同一個節點上執行。程序的記憶體也配置於此。

「**numastat**」是由「*numactl*」套件提供。欲進一步了解 「**numastat**」輸出的訊息，請見 man page：

```
$ man numastat
```

### 3.2.3. /proc/interrupts

「**`/proc/interrupts`**」檔案列出從特定 I/O 裝置傳送到每一個處理器的插斷數目。它顯示 IRQ 碼、有多少同樣經由系統內處理器來處理的插斷要求、被傳送的插斷類型以及逗點分隔清單，此清單顯示回應列出的插斷要求的裝置。

如果有一個特定的應用程式或裝置產生大量要由遠端處理器來處理的插斷要求，它的效能很可能會下降。遇到這樣的情況，將在同一節點上的處理器設為處理插斷要求的應用程式或裝置，便可改善低效能的情況。欲了解如何指派插斷處理至一個特定的處理器，請見 〈節 3.3.7, “設定插斷親和性”〉。

## 3.3. 配置建議

Red Hat Enterprise Linux 提供一些工具來輔助系統管理員設定系統。這一段落將介紹有哪一些可用的工具，也會以例子說明這些工具如何被用來解決 Red Hat Enterprise Linux 7 中與處理器有關的效能問題。

### 3.3.1. 設定 kernel tick 的時間

預設中，Red Hat Enterprise Linux 7 使用的是 tickless kernel 。它不會透過插斷閒置的 CPU 來達到省電的效果，也會讓較新的處理器處於深沉睡眠狀態。

Red Hat Enterprise Linux 7 也提供動態 tickless 選項（預設為停用）。這個對延遲敏感的工作負載來說很實用，例如高效能運算或是及時運算。

若要在特定核心啓用動態 tickless 行為，就指定那些在 kernel 命令列上面有「***`nohz_full`***」參數的核心。如果使用 16 核心的系統，指定「**`nohz_full=1-15`**」可以在核心 1 到 15 啓用動態 tickless 行為，並且將所有時程序移到唯一一個未被指定的核心上（core 0）。這個行為可以在開機時被暫時啓用，或是在「**`/etc/default/grub`**」檔案持續地啓用。要持續地啓用，請執行「**`grub2-mkconfig -o /boot/grub2/grub.cfg`**」指令來儲存設定。

要啓用動態 tickless 行為需要一些手動管理。

- 當系統開機時，您必須手動將 rcu 執行緒移動至非延遲敏感核心。在這裡就是指 0 核心。

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

- 在 kernel 命令列使用「***isolcpus***」參數將特定核心與使用者空間的工作隔離。
- 選擇項，將 CPU 對 kernel 的寫回 bdi-flush 執行緒的親和性設定為核心：

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

要確認動態 tickless 設定是否正常運作，請執行以下命令。「*stress*」是一個在 CPU 上空轉一秒的程式。

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

可以取代「*stress*」的指令碼類似「**while :; do d=1; done**」。另外一個可以取代「*stress*」的是以下連結的程式：〈https://dl.fedoraproject.org/pub/epel/6/x86_64/repoview/stress.html〉。

kernel 計時器設定預設為在忙碌的 CPU 上顯示 1000 個 tick：

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

設定動態 tickless kernel 後，您應該只會看見一個 tick。

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

### 3.3.2. 設定硬體效能原則

「**x86_energy_perf_policy**」工具允許系統管理員定義效能與能源效率相對的重要性。選擇 CPU 的高效能或高節能選項時，這項資訊就可以用來對支持此功能的 CPU 造成影響。

在預設情況下，此工具可以在所有處於「**performance**」（效能） 模式的處理器上執行。它需要處理器的支援，可透過「**CPUID.06H.ECX.bit3**」的狀態得知，還必須以 root 使用者權限執行。

「**x86_energy_perf_policy**」是由「*kernel-tools*」套件所提供。欲了解使用「**x86_energy_perf_policy**」的細節請見 〈節 A.10, “x86_energy_perf_policy”〉，或是參考 man page：

```
$ man x86_energy_perf_policy
```

### 3.3.3. 用 taskset 設定程序親和性

「**taskset**」工具是由「*util-linux*」套件所提供。「**taskset**」允許系統管理員取出以及設定執行中程序的處理器親和性，或是用特定的處理器親和性來啓動程序。

**重要**

「**taskset**」不保證本機記憶體的配置。如果您需要其他的效能效益或是本機記憶體配置，Red Hat 建議您使用「**numactl**」來取代「**taskset**」。

欲了解更多關於「**taskset**」的資訊，請見〈節 A.16, “taskset”〉或是 man page：

```
$ man taskset
```

### 3.3.4. 用 numactl 管理 NUMA 親和性

系統管理員可以使用「**numactl**」來執行有特定排程或是記憶體放置原則的程序。「**numactl**」也可以為了共用的記憶體區段或檔案設定常設原則，以及設定程序的處理器親合性和記憶體親和性。

使用 NUMA 拓樸的系統中，當處理器跟記憶體插槽之間的距離變長，處理器的記憶體存取會變慢。因此，將應用程式設定為對效能敏感是很重要的，這樣才能將記憶體配置到最近的記憶體插槽。最好使用位於相同 NUMA 節點中的記憶體以及 CPU。

對效能敏感的多執行緒應用程式來說，與其被設定為在一個特定的處理器上執行，被設定為在特定 NUMA 節點上執行比較有效。這個設定是否合適則要視您的應用程式的系統以及需求而定。如果多重應用程式執行緒存取一樣的快取資料，那麼將這些執行緒設定為在同一個處理器上執行就很合適。但是，如果多重執行緒在同一個處理器上存取以及快取不同的資料，那麼每一個執行緒可以收回由前一個執行緒快取的資料。意思是，每一個執行緒「遺漏」（miss）快取，而且把執行時間浪費在從磁碟擷取資料上以及在快取裡取代它。您可以使用「**perf**」工具。就如同〈節 A.7, “perf”〉記載的一樣，此工具可以被用來檢查過多的快取遺漏。

「**numactl**」提供許多選項來幫助您管理處理器以及記憶體親和性。欲了解細節，請見〈節 A.12, “numastat”〉或是man page：

```
$ man numactl
```

**注意**

「**numactl**」套件包含「**libnuma**」程式庫。這個程式庫提供簡單的程式設計介面給由 kernel 支援的 NUMA 原則使用，這個介面還能被用來操作精密的微調，比「**numactl**」應用程式來的精密。更多資訊請見 man page：

```
$ man numa
```

### 3.3.5. 用 numad 自動管理 NUMA 親和性

「**numad**」 是自動的 NUMA 親和性管理精靈。它透過監視 NUMA 拓樸以及系統內的資源使用方式來動態改善 NUMA 資源分配與管理。

「**numad**」也提供放置前的建議服務來幫助程序的 CPU 以及記憶體的初始繫結。此服務可以透過許多工作管理系統來查詢。不管 numad 是以可執行檔或是服務來執行，這個放置前的建議服務都可使用。

欲了解如何使用「**numad**」，請見〈 節 A.14, “numad”〉或參考 man page：

```
$ man numad
```

## 3.3.6. 微調排程原則

Linux 排程器實施許多排程原則，這些原則用來決定執行緒在哪裡執行以及執行多久。排程原則有兩個主要的類別：標準原則以及及時原則。標準執行緒是用來執行一般優先順序的工作；及時原則是用來處理時間緊迫、不能插斷才能完成的工作。

及時執行緒不是在時間配量的前提之下。這個意思是及時執行緒會持續執行到它們封鎖、結束、自動讓出、或者被較高優先順序的執行緒優先占用。最低優先順序的及時執行緒會在任何應用標準原則的執行緒之前排程。

### 3.3.6.1. 排程原則

#### 3.3.6.1.1. 用 SCHED_FIFO 靜態優先順序排程

「**SCHED_FIFO**」（或者被稱為靜態優先順序排程）是一個定義每個執行緒的固定優先順序的即時原則。這個原則允許系統管理員改善回應事件的時間以及減少延遲。我們建議對時間敏感，且不長時間執行的工作可以應用此原則。

當「**SCHED_FIFO**」在使用中，排程器會依優先順序掃描全部「**SCHED_FIFO**」執行緒的清單，也會排程執行最優先順序的執行緒。「**SCHED_FIFO**」執行緒的優先權層級可以是介於 1 到 99 的任何一個整數（99 被視為是最高優先順序）。Red Hat 建議從較低的數字開始，當您識別延遲問題時再增加優先順序。

**警告**

由於及時執行緒並不是在時間配量的前提之下，Red Hat 不建議將優先順序設為 99。這會將您的程序設為與移轉以及看門狗執行緒相同的優先權層級。如果您的執行緒進入計算迴圈，而且這些執行緒被封鎖了，那麼它們就無法執行。有單一處理器的系統終究會懸置於這樣的狀況中。

系統管理員可以限制「**SCHED_FIFO**」頻寬來防止及時應用程式的程式設計師起始會壟斷處理器的及時工作。

**/proc/sys/kernel/sched_rt_period_us**

這個參數定義微秒的時間長度，它被認為是百分之百的處理序頻寬。預設值為「**1000000** µs」或是一秒。

**/proc/sys/kernel/sched_rt_runtime_us**

這個參數定義微秒的時間長度，它專門執行及時執行緒。預設值為「**950000** µs」或 0.95 秒。

#### 3.3.6.1.2. 用 SCHED_RR 輪詢優先排程法

「**SCHED_RR**」是「**SCHED_FIFO**」的變異輪詢法。當多重執行緒必須在同一個優先權層級上執行時，這個原則就很有幫助。

就像「**SCHED_FIFO**」，「**SCHED_RR**」是一個執行原則，被用來定義每一個執行緒的固定優先順序。排程器依照優先順序掃描所有「**SCHED_RR**」執行緒的清單，然後排程已經準備好要執行的最高優先順序的執行緒。但是，跟「**SCHED_FIFO**」不一樣的是，有同樣優先順序的執行緒會在某些時間配置下以輪詢法排程。

您可以用「***sched_rr_timeslice_ms***」kernel 參數（「**/proc/sys/kernel/sched_rr_timeslice_ms**」）將此時間配置的數值設定為毫秒。最低的數值是一毫秒。

#### 3.3.6.1.3. 用 SCHED_OTHER 標準排程

「**SCHED_OTHER**」是 Red Hat Enterprise Linux 7 的預設排程原則。這個原則使用完全公平排程（CFS）來允許處理器公平連接至所有執行緒。當大量執行緒或是資料輸送為優先時，應用這個原則是最有幫助的，因為它會隨著時間允許更有效率的執行緒排程。

當此原則被使用，排程器會建立一個動態優先順序清單。這是部分根據每個程序執行緒的精密值來建立的。系統管理員可以變更程序的精密值，但是不能直接變更排程器的動態優先順序。

欲了解變更精密程序的細節，請見「*Red Hat Enterprise Linux 7 Deployment Guide*」，此指南位於〈http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/〉。

#### 3.3.6.2. 隔離 CPU

您可以使用「***isolcpus***」開機參數將一個或一個以上的 CPU 從排程器隔離。這樣可以防止排程器將任何使用者空間的執行緒排程置這個 CPU 上。

一旦 CPU 被隔離，您必須手動指派程序至被隔離的 CPU。您可以使用 CPU 親和性系統或是 numactl 指令。

要隔離系統上第三個或是第六個到第八個的 CPU，請將以下加至 kernel 命令列：

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

您也可以使用「**Tuna**」工具來隔離 CPU。「**Tuna**」可以在任何時間隔離 CPU，不是只有開機的時候。但是這種隔離的方法跟 ***isolcpus*** 參數有些微的不同，而且目前無法獲得與「***isolcpus***」相關的效能。請見〈節 3.3.8, “用 Tuna 來配置 CPU、執行緒，以及插斷親和性”〉以便了解更多關於此工具的資訊。

### 3.3.7. 設定插斷親和性

「插斷要求」有相關的親和性屬性。此屬性為「***smp_affinity***」，它定義控制插斷要求的處理器。要改善應用程式效能，請指派插斷親和性以及程序親和性至同一個處理器、或是同一個核心的多個處理器上。這允許被指定的插斷以及應用程式執行緒分享快取的樣式。

特定插斷要求的插斷親和性值儲存於相關的「**/proc/irq/*irq_number*/smp_affinity**」檔案中。「***smp_affinity***」是以十六進位位元遮罩來儲存，它代表系統內所有的處理器。它的預設值為「**f**」，意思是插斷要求可以在系統內任何處理器上被控制。將預設值設為「**1**」代表只有 0 號處理器可以控制插斷。

在內含超過 32 個處理器的系統中，您必須為了分開 32 位元群組分隔「***smp_affinity***」值。舉例來說，如果您只想要 64 處理器的系統中前 32 個處理器來服務插斷要求，您可以執行：

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

或者，如果 BIOS 匯出它的 NUMA 拓樸，「**irqbalance**」服務可以使用此資訊來處理在本機硬體要求的服務節點上的插斷要求。欲了解「**irqbalance**」，請見〈節 A.1, “irqbalance”〉。

**注意**

在支援插斷轉向的系統中，修改插斷要求的「***smp_affinity***」可以建立硬碟。所以說，在沒有 kernel 的插斷之下，硬碟可以決定用哪些特定的處理器來處理插斷。欲了解更多關於插斷轉向的資訊，請見〈章 6, *網路功能*〉。

### 3.3.8. 用 Tuna 來配置 CPU、執行緒，以及插斷親和性

「**Tuna**」可以控制 CPU、執行緒、插斷親和性，以及提供它可以控制的每一個型別的實體許多動作。欲見「**Tuna**」功能的完整清單，請見〈節 A.2, “Tuna”〉。

為了要將所有執行緒從一個或多個被指定的 CPU 移走，請執行以下指令，並用您想要隔離的 CPU 編號取代「*CPU*」。

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

要將一個 CPU 包含在一列可以執行特定執行緒的 CPU 清單，請執行以下指令，並且用您想要包含的 CPU 編號來取代「*CPUs*」。

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

要將插斷要求移至一個被指定的 CPU，請執行以下指令，並且用 CPU 的編號取代「*CPU*」。另外，用您想要移動的插斷要求的逗點分隔清單來取代「*IRQs*」。

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

或者，您可以使用以下指令來找出所有「**sfc1***」模式的插斷要求。

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

要變更原則以及一個執行緒的優先順序，請執行以下指令，用您想變更的執行緒來取代「*thread*」；將您想要用來執行執行緒的原則來取代「*policy*」；用 0（最低優先順序）到 99（最高優先順序）之間的整數來取代「*level*」。

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

# 章 4. 記憶體

本章節將概述 Red Hat Enterprise Linux 7 管理記憶體的能力。〈節 4.1, “考量”〉討論與記憶體相關影響效能的因素。〈節 4.2, “監視與診斷效能問題”〉教您如何使用 Red Hat Enterprise Linux 7 工具來診斷與記憶體使用率或設定細節相關的效能問題。〈節 4.3, “配置工具”〉則討論可以被用來解決 Red Hat Enterprise Linux 7 內與記憶體相關的效能問題的工具與策略。

## 4.1. 考量

在預設情況下，系統會最佳化 Red Hat Enterprise Linux 7 來控管工作負載。如果您的系統管理員或使用案例需要大量的記憶體，更改系統管理虛擬記憶體的方式可以改善應用程式的效能。

### 4.1.1. 分頁大小

實體記憶體是由稱為分頁的區塊來管理。每一分頁的實體位置會對應至一個虛擬的位置，好讓處理器可以存取記憶體。此對應儲存於資料結構中，稱為分頁表。

在預設情況下，一頁大概是 4 KB 大小。因為預設分頁很小，您需要很多分頁來管理大量的記憶體。然而，分頁表只能儲存有限的位址對應，增加它可以儲存的位址對應不僅很昂貴，對將效能等級維持在記憶體需求等級來說也很困難。

Red Hat Enterprise Linux 也讓每一個分頁用靜態大分頁管理大量記憶體。靜態大分頁最大可以被設為 1 GB。但是它們很難用手動管理，而且必須在開機時就被指派。至於 Red Hat Enterprise Linux 7.1，它們可以透過每個節點為基準被指派。

透明大分頁是靜態大分頁大量自動化的替代品。透明大分頁的大小是 2 MB，而且可以在預設情況中啓用。它們有時候可能會影響延遲敏感的應用程式，所以當延遲有可能發生時，它們會被停用。

欲了解設定大分頁來改善應用程式效能的細節，請見〈節 4.3.1, “設定巨大分頁”〉。

### 4.1.2. 轉譯對應緩衝區大小

從分頁表讀位置對應是很浪費時間而且消耗資源的，所以 Linux 執行系統為近期使用的位置提供快取：轉譯對應緩衝區（TLB）。但是在預設情況下 TLB 只能快取一定數量的位置對應。如果要求的位置對應不在 TLB 裡面（意思是這個 TLB 是「*未猜中*」— miss 的），那麼系統仍然需要讀取分頁表來決定虛擬位置對應的實體。

因為應用程式記憶體需求與用來快取位置對應的分頁大小之間的關係，對有大量記憶體需求的應用程式來說，跟有最小記憶體需求的應用程式比起來，其效能較有可能因為 TLB 未猜中而下降。因此要盡量避免 TLB 未猜中。

Red Hat Enterprise Linux 提供超大轉譯緩衝器（HugeTLB）。它讓記憶體可以在非常大的區段被管理。這讓系統可以一次快取較多的位置對應，也減少 TLB 未猜中的可能性，進而改善有大量記憶體需求的應用程式的效能。

欲了解設定 HugeTLB 的細節，請見〈節 4.3.1, “設定巨大分頁”〉。

## 4.2. 監視與診斷效能問題

Red Hat Enterprise Linux 7 提供一些對系統效能與診斷跟系統記憶體相關的效能問題來說很實用的工具。本小節將概述有哪些可用的工具，以及舉例說明如何使用它們來監視與診斷跟效能問題相關的記憶體。

### 4.2.1. 用 vmstat 監視記憶體使用量

「**Vmstat**」是由「*procps-ng*」套件所提供。它輸出系統的程序、記憶體、分頁、區塊輸入/輸出、插斷以及 CPU 活動的報告。它也提供這台機器從上次開機後這些事件的平均報告，或是從前一次報告後開始記錄。

以下指令顯示各種事件計數器以及記憶體數據

```
$ vmstat -s
```

欲進一步了解使用 「**vmstat**」的細節請見〈節 A.9, “vmstat”〉或是參考 man page：

```
$ man vmstat
```

### 4.2.2. 用 Valgrind 設定應用程式記憶體使用量

「**Valgrind**」是一個提供使用者空間二進位檔檢測的架構。這個架構之下有一些可以被用來設定以及分析程式效能的工具。此小節中概述的「**valgrind**」工具可以幫助您偵測記憶體錯誤，例如未初始化的記憶體使用以及不正確的記憶體配置或是解除配置。

要使用「**valgrind**」或是它的其它工具，請安裝「*valgrind*」套件：

```
# yum install valgrind
```

#### 4.2.2.1. 用 Memcheck 設定記憶體使用量

「**Memcheck**」是「**valgrind**」預設情況下的工具。它偵測以及對一些難以被偵測或診斷出來的記憶體錯誤做出報告，例如：

- 不應該發生的記憶體存取
- 使用未被定義或未被使用的數值
- 不正確的釋出堆積記憶體
- 指標重疊
- 記憶體流失

**注意**

「**Memcheck**」只能報告這些錯誤，它不能阻止錯誤發生。如果您的程式存取記憶體的方式通常都會導致離散錯誤，那麼這個離散錯誤仍然會發生。然而，「**memcheck**」會及時在錯誤發生前記錄錯誤訊息。

由於「**memcheck**」會使用檢測，所以用「**memcheck**」執行的應用程式比平常慢十到二十倍。

欲在應用程式上執行「**memcheck**」，請執行以下指令：

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

您也可以使用以下選項來控制「**memcheck**」特定類別的問題輸出。

**--leak-check**

在應用程式結束執行時，「**memcheck**」會搜尋記憶體流失。預設值為「**--leak-check=summary**」，它會列印記憶體流失的數量。您可以指定「**--leak-check=yes**」或者「**--leak-check=full**」來輸出每一個記憶體流失的細節。欲停用此選項，請指定「**--leak-check=no**」。

**--undef-value-errors**

其預設值為「**--undef-value-errors=yes**」。若未被定義的數值被使用，它會對此提出報告。您也可以指定「**--undef-value-errors=no**」，它會停用此報告也會略微加速 Memcheck。

**--ignore-ranges**

在檢查記憶體定址能力時指定一個或一個以上「**memcheck**」應該忽略的範圍，例如「**--ignore-ranges=0xPP-0xQQ,0xRR-0xSS**」。

欲見「**memcheck**」 選項的完整清單，請見包含在「**/usr/share/doc/valgrind-version/valgrind_manual.pdf**」的文件。

#### 4.2.2.2. 用 Cachegrind 分析快取使用方式

「**Cachegrind**」用系統的快取階層以及分支預測器模擬應用程式的互動。它會追蹤第一層級指令以及資料快取所模擬的使用量，進而偵測此層級的快取中較差的代碼互動。它也會追蹤最後一個層級的快取（第二或是第三層級）來追蹤記憶體的存取。因此用「**cachegrind**」執行的應用程式的執行速度會比平常慢二十到一百倍。

「**Cachegrind**」收集應用程式執行期間的統計資料，並輸出摘要至主控台。要在應用程式上執行「**cachegrind**」，請執行以下指令：

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

您也可以使用以下選項來將「**cachegrind**」輸出集中於一個特定的問題上。

**--I1**

它指定第一層級指令快取的大小、關聯性以及行大小：「**--I1=*size*,*associativity*,*line_size***」。

**--D1**

它指定第一層級資料快取的大小、關聯性以及線條大小：「**--D1=*size*,*associativity*,*line_size***」

**--LL**

它指定最後一個層級快取的大小、關聯性以及線條大小：「**--LL=*size*,*associativity*,*line_size***」。

**--cache-sim**

它將啓用或者停用快取存取的集合與遺失記數。這在預設情況下是經由（「**--cache-sim=yes**」）被啓用。若停用這個指令與「**--branch-sim**」，則會讓「**cachegrind**」收集不到資料。

**--branch-sim**

它啓用或者停用分支指令的收集以及不正確的預測計數。它在預設情況下是經由（「**--branch-sim=yes**」）被啓用。若停用這個指令與「**--cache-sim**」則會讓「**cachegrind**」收集不到資訊。

「**Cachegrind**」將詳細的分析資訊寫入「**cachegrind.out.*pid***」檔案的每一個處理序，其中「*pid*」為處理序的識別碼。此資訊可以透過小幫手工具「**cg_annotate**」來進一步處理：

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

「**Cachegrind**」也提供 cg_diff 工具給使用者，這讓代碼改變前與改變後繪製程式效能圖表更加容易。欲比較輸出的檔案，請執行以下指令，並且以初始設定檔的輸出檔案來取代第一個檔案；用後續設定檔的輸出檔案來取代第二個檔案。

```
# cg_diff first second
```

其產生的輸出檔案可以用「**cg_annotate**」工具來檢視更多細節。

欲了解「**cachegrind**」選項的完整清單，請參考包含於「**/usr/share/doc/valgrind-version/valgrind_manual.pdf**」的文件。

#### 4.2.2.3. 使用 Massif 來分析堆積與堆疊空間

「**Massif**」會測量某一個特定的應用程式使用的堆疊空間。它會測量實用的空間以及任何其它配置於簿記以及對齊目的的其它空間。「**massif**」可以幫助您了解如何透過減少應用程式的記憶體使用量，來加快執行速度、減少應用程式耗盡系統切換空間的可能性。用「**massif**」執行的應用程式會比平常慢大約二十倍。

欲在應用程式上執行「**massif**」，請執行以下指令：

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

您也可以使用以下選項來將「**massif**」集中輸出於一個特定的問題上。

**--heap**

它指定「**massif**」是否設定堆積。其預設值為「**--heap=yes**」。若將這個設定值設為「**--heap=no**」，堆積分析就會被停用。

**--heap-admin**

它指定每一個區塊在堆積分析被啓用時系統管理員可以使用的位元組。其預設值為「**8**」位元組。

**--stacks**

它指定「**massif**」是否設定堆積。其預設值為「**--stack=no**」，因為堆積分析會大大降低「**massif**」的速度。請將此選項設為「**--stack=yes**」來啓用堆積分析。要注意的是，「**massif**」採用的設定是主堆積於大小為零時開始，以便更清楚的表示與被設定的應用程式相關的堆積大小的改變。

**--time-unit**

它指定「**massif**」收集分析資料的間隔。其預設值為「**i**」（執行指令）。您也可以指定「**ms**」（毫秒，或及時）以及「**B**」（配置的位元組或是在堆積與堆疊上被取消配置的位元組）。測試配置的位元組對簡短執行的應用程式以及測試目的是很實用的，因為它可重現於不同的硬體上。

「**Massif**」輸出分析資料至「**massif.out.*pid***」檔案。「*pid*」是這個檔案的處理序識別碼。「**ms_print**」工具將此分析資料製成圖表來顯示記憶體在應用程式執行時的消耗量，同時也會顯示負責指派記憶體高峰點配置的網站的詳細資料。要將「**massif.out.*pid***」檔案的資料製成圖表，請執行以下指令：

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

欲了解「**Massif**」選項的完整清單，請參考包含於「**/usr/share/doc/valgrind-version/valgrind_manual.pdf**」的文件。

## 4.3. 配置工具

記憶體使用量通常是透過設定一個或是一個以上的 kernel 參數來設定的。這些參數可以透過改變 /proc 檔案系統的內容暫時設定，或者它們可以透過 sysctl 工具永久設定。此工具是由「*procps-ng*」套件所提供。

舉例來說，要將 overcommit_memory 參數暫時設為 1，請執行以下指令：

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

欲持續設定此數值，請執行以下指令：

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

暫時設定一個參數是很實用的，它可以確定此參數在您系統上的效果。之後當您確定此參數值有達到想要的效果時便可以持續設定此參數。

### 4.3.1. 設定巨大分頁

大分頁依賴連續的記憶體區域，所以在開機時（在記憶體被分割為片段之前）就定義巨大分頁是最佳方法。欲這麼做，請將以下參數加至 kernel 開機命令列：

**hugepages**

它會定義在開機時設定於 kernel 中巨大分頁的數量。其預設值為 0。只有在系統內有足夠的連續實體空間分頁時才有辦法配置（或是解除配置）巨大分頁。保留於此參數的分頁將不能被用作其它用途。

您可以在開機後透過變更「**/proc/sys/vm/nr_hugepages**」檔案的數值來調整此數值。

在 NUMA 系統內，用此參數指派的巨大分頁在不同節點之間被平均分割。您可以在執行階段透過變更節點的「**/sys/devices/system/node/*node_id*/hugepages/hugepages-1048576kB/nr_hugepages**」檔案數值，將巨大分頁指派至特定節點。

欲了解更多資訊，請閱讀與 kernel 相關的文件。這些文件在預設情況下被安裝於「**/usr/share/doc/kernel-doc-*kernel_version*/Documentation/vm/hugetlbpage.txt**」。

**hugepagesz**

它在開機時定義設定在 kernel 上的持續巨大分頁的大小。有效值為 2 MB 或 1 GB，預設值為 2 MB。

**default_hugepagesz**

它在開機時定義設定在 kernel 中持續巨大頁面的預設大小。有效值為 2 MB 或 1 GB，預設值為 2 MB。

您也可以使用以下參數來影響巨大分頁在執行階段的行為：

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

它定義被指定到特定 NUMA 節點上，有特定大小的巨大分頁的數量。Red Hat Enterprise Linux 7.1 有支援此功能。以下例子將二十個 2048 kB 巨大分頁新增至「**node2**」。

```
# numastat -cm | egrep 'Node|Huge'
                 Node 0 Node 1 Node 2 Node 3  Total
AnonHugePages         0      2      0      8     10
HugePages_Total       0      0      0      0      0
HugePages_Free        0      0      0      0      0
HugePages_Surp        0      0      0      0      0
# echo 20 > /sys/devices/system/node/node2/hugepages/hugepages-
2048kB/nr_hugepages
# numastat -cm | egrep 'Node|Huge'
                 Node 0 Node 1 Node 2 Node 3  Total
AnonHugePages         0      2      0      8     10
HugePages_Total       0      0     40      0     40
HugePages_Free        0      0     40      0     40
HugePages_Surp        0      0      0      0      0
```

**/proc/sys/vm/nr_overcommit_hugepages**

它定義過度使用記憶體時，可以建立以及系統可以使用的其它大分頁的最大數目。如果將非零值寫進此檔案，這就表示系統在永續性大分頁耗盡時，能從 kernel 的標準分頁集區取得的大分頁數目。這些剩餘大分頁在未被使用的情況下，會處於空閒狀態而且會被傳回 kernel 的標準分頁集區。

### 4.3.2. 設定系統記憶體容量

本小節將討論可能對改善系統的記憶體使用率有幫助，並且與記憶體有關的 kernel 參數。這些參數可以透過改變「**/proc**」檔案系統內的對應檔案而被暫時設定以做測試之用。一旦您確定產生最佳效能的數值，您可以使用「**sysctl**」指令來持續設定參數。

#### 4.3.2.1. 虛擬記憶體參數

除非另外設定，否則本小節列出的參數的位置為「**/proc/sys/vm**」。

**dirty_ratio**

它是一個百分比值。當系統總記憶體的百分比值被修改，那麼系統會開始用「**pdflush**」運算將修改後的數值寫進磁碟裡。其預設值為百分之 20。

**dirty_background_ratio**

它是一個百分比值。當系統的總記憶體百分比值被修改，那麼系統會開始將修改後的數值寫進磁碟背景裡。其預設值為百分之 10。

**overcommit_memory**

它定義大量的記憶體需求是否被接受或拒絕的條件。

其預設值為「**0**」。在預設情況下，kernel 透過評估可用的記憶體以及拒絕需要過大記憶體的要求，來執行啓發式記憶體過量使用。然而，因為記憶體是使用啓發式學習（不是使用精確演算法）被配置的，用此設定多載記憶體是行的通的。

當此參數被設為「**1**」，那麼 kernel 會執行無記憶體過量使用的處理。這將增加記憶體多載的可能性，但是會改善記憶體高用量工作的效能。

當此參數被設為「**2**」時，kernel 會拒絕與被指定在「***overcommit_ratio***」內相同或是大於可用的總切換空間以及實體 RAM 的百分比的記憶體要求。這會減少過量使用記憶體的風險，但是較適用於切換區域大於實體記憶體的系統。

**overcommit_ratio**

它在「***overcommit_memory***」被設為「**2**」 指定被考慮的實體 RAM 的百分比。其預設值為「**50**」。

**max_map_count**

它定義程序可以使用的最大記憶體對應區域的數目。其預設值（「**65530**」）適合大部分的情況。若您的應用程式需要對應比這個數目更多的檔案，請增加此預設值。

**min_free_kbytes**

它指定最小的 KB 為多少來讓系統保持空閒。它被用來確定每一個低記憶體區域適合的數值，而每一個數值都會被指派一些與它的大小成比例的空閒頁面。

**警告**

極高的數值會損壞您的系統。將「***min_free_kbytes***」設在一個極低的數值可以防止您的系統回收記憶體。系統回收記憶體可能會造成系統當機以及 OOM-killing 的程序。但是若將「***min_free_kbytes***」設太高（例如將其設為系統總記憶體的 5–10%）將會導致系統立即進入記憶體不足的狀態，進而造成系統花太多時間來回收記憶體。

**oom_adj**

在系統的記憶體用罄，以及「***panic_on_oom***」參數被設為「**0**」的情況下，「**oom_killer**」函數會一直刪除程序直到系統復原。它將從有最高「**oom_score**」的程序開始刪除。

「***oom_adj***」參數可以幫助確定程序的「***oom_score***」。此參數是依每一個程序的識別碼而設定的。若數值為「**-17**」則會為了那個程序停用「**oom_killer**」。其它有效的數值介於「**-16**」到「**15**」之間。

**注意**

從調整過的程序繁衍而來的程序會繼承原來程序的「**oom_score**」。

**swappiness**

它是一個從「**0**」 到「 **100**」的數值。它控制系統協助匿名記憶體或是分頁快取的程度。高數值可以改善檔案系統的效能，同時主動在 RAM 內切換較少被使用的程序。低數值可以避免將程序切換出記憶體，這通常會降低延遲，並且消耗 I/O 效能。其預設值為「**60**」。

**警告**

設定「**swappiness==0**」將會讓系統不停的避免切換，這會增加系統在強式記憶體以及 I/O 壓力之下 OOM 的刪除現象。

#### 4.3.2.2. 系統檔案參數

除非另外指派，否則本小節內列出的參數皆位於「**/proc/sys/fs**」中。

**aio-max-nr**

它定義全部使用中的非同步輸入 / 輸出內容可允許的事件數量。其預設值為「**65536**」。修改此數值不會預先配置或者調整任何 kernel 資料結構的大小。

**file-max**

它定義由 kernel 配置的檔案控點的最大數目。其預設值與 kernel 內的「**files_stat.max_files**」數值相符。此數值被設為最大值，其源自於「***NR_FILE***」（在 Red Hat Enterprise Linux 內為 8192）或者源自於以下指令的結果：

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

提高此數值可以解決因為缺少可用的檔案控點而造成的錯誤。

### 4.3.2.3. kernel 參數

除非另外指派，否則本小節列出的參數皆位於「**/proc/sys/kernel**」中。

**msgmax**

它定義在訊息佇列內任何單一訊息可允許的位元組大小上限。此數值不能大於佇列（「***msgmnb***」）的大小。其預設值為「**65536**」。

**msgmnb**

它定義在單一訊息佇列內的位元組大小上限。其預設值為「**65536**」。

**msgmni**

它定義訊息佇列識別碼的數量上限（意思是佇列的數量上限）。在 64 位元架構的系統內，其預設值為「**1985**」。

**shmall**

它定義系統內的頁面可分享記憶體的數量上限。在 Red Hat Enterprise Linux 系統內，其預設值為「**1<<24**」或是 33554432 頁。

**shmmax**

它定義 kernel 允許的單一可分享的記憶體區段頁面的大小上限。在 Red Hat Enterprise Linux 系統內其預設值為「**1<<24**」或是 33554432 位元組。

**shmmni**

它定義系統內可分享記憶體區段的數量上限。在所有系統內其預設值皆為「**4096**」。

**threads-max**

它定義整個系統內 kernel 一次可允許的執行緒數量上限。其預設值與 kernel 參數「***max_threads***」的數值是一樣的，或者也有可能是以下指令的結果：

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

最小值為「**20**」。

# 章 5. 儲存空間與檔案系統

本章節將概述支援的檔案系統，以及在 Red Hat Enterprise Linux 7 內 I/O 與檔案系統中會影響應用程式效能的設定選項。〈節 5.1, “考量”〉將討論與 I/O 和檔案系統有關的因素，這些因素會影響效能。〈節 5.2, “監視以及診斷效能問題”〉將教您如何使用 Red Hat Enterprise Linux 7 的工具來診斷與 I/O 以及檔案系統設定細節有關的效能問題。〈節 5.3, “設定工具”〉將討論您可以用來解決 Red Hat Enterprise Linux 7 內與 I/O 以及檔案系統相關的效能問題的工具以及策略。

## 5.1. 考量

適當的儲存空間設定以及檔案系統效能設定，與儲存空間的目有很大的關係。I/O 以及系統效能會受到以下的因素影響：

- 資料寫入或讀取模式
- 將軟體的資料分布與硬碟的配置保持一致。
- 區塊大小
- 檔案系統大小
- 日誌大小與位置
- 記錄存取時間
- 確保資料可靠性
- 預先截取資料
- 預先配置磁碟空間
- 分散檔案
- 資源爭用

請閱讀本章節來了解會影響檔案系統的輸送、延展性、回應、資源使用量以及可用性的掛載選項和格式調整選項。

### 5.1.1. 固態磁碟

固態磁碟 ( SSD ) 使用 NAND 的快閃晶片來儲存永續性資料，而不是使用旋轉磁盤。它們在整個邏輯區塊位址的範圍內持續提供存取資料的時間，而且不會像旋轉磁盤一樣造成搜尋的成本。它們每一 GB 的儲存空間較貴，而且有較少的儲存密度，但是跟 HDD 硬碟比起來，它們也有較低的延遲現象以及更大的運輸量。

當 SSD 上被使用的區塊接近磁碟的容量，效能通常會下降。效能下降的程度會隨著廠商而不同，但是所有裝置的效能都會在這樣的情況之下下降。啓用捨棄行為可以減緩效能下降。欲了解更多細節請見〈節 5.1.4.3, “維護”〉。

在預設情況下，I/O 排程器以及虛擬記憶體選項很適合與 SSD 一起使用。

關於 SSD 部署建議的其他資訊請見《*Red Hat Enterprise Linux 7 儲存管理指引*》，以下為連結：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

### 5.1.2. I/O 排程器

I/O 排程器決定 I/O 作業在儲存裝置上執行的時間以及執行時數。它也被稱為 I/O 升降舵。

Red Hat Enterprise Linux 7 提供三個 I/O 排程器。

**deadline**

預設的 I/O 排程器會封鎖 SATA 磁碟以外的裝置。「**deadline**」試圖在要求到達 I/O 排程器時開始提供保證延遲。這一個排程器在大部分使用情況下是很適合的，但是尤其適合那些讀取作業比寫入作業更常發生的情況。

已排入佇列的 I/O 要求會被分類為讀取批次或是寫入批次，然後會被排程來執行增加 LBA 次序。讀取批次在預設情況下比寫入批次優先，因為應用程式在讀取 I/O 時較容易封鎖。當一個批次處理完成，「**deadline**」會檢查寫入作業已經渴望處理器的時間多久了，然後會排程下一個適合的讀取或是寫入批次。每一個批次處理的要求數量、每一個寫入批次發行的讀取批次數量以及要求過期之前有多少時間都是可以設定的。欲了解更多細節，請見〈節 5.3.4, “微調排程器期限”〉。

**cfq**

預設的排程器只適用於被識別為 SATA 磁碟的裝置。「**cfq**」是完全公平隊列（ Completely Fair Queueing）排程器，它將程序分為三個不同的類別：即時程序、盡力程序以及閒置程序。即時程序比盡力程序先執行；而盡力程序比閒置程序先執行。這意味著即時類別的程序急需盡力程序，以及閒置程序的處理器時間。程序可以經由預設，指派至盡力類別。

「**cfq**」使用歷史資料來預測應用程式之後是否會發出更多 I/O 要求。如果會有更多 I/O，「**cfq**」會閒置來等新的 I/O，就算別的程序的 I/O 在等待執行也相同。

由於這個閒置的傾向，cfq 排程器不應該與不會造成大量搜尋罰則的硬體接合使用，除非它被調正過了。它也不應該與其他非工作節省的排程器接合使用，例如主機型硬體 RAID 控制器，因為堆疊這些排程器會導致大量的延遲。

「**cfq**」行為是可以被設定的，欲了解更多細節請見〈節 5.3.5, “微調 cfq 排程器”〉。

**noop**

「**noop**」I/O 排程器實施簡易的先進先出 FIFO（first-in first-out）排程演算法。請求會透過上次使用的快取被合併於一般區塊層。對於使用快速儲存，且極度仰賴 CPU 的系統，這是最好的排程器。

欲了解設定不同的 I/O 排程器，或是給一個特定的裝置指定不同排程器的細節，請見〈節 5.3, “設定工具”〉。

### 5.1.3. 檔案系統

請閱讀本小節來了解 Red Hat Enterprise Linux 7 支援的檔案系統的細節、建議的使用案例以及各檔案系統普遍可以使用的格式和掛接選項。欲了解這些檔案系統詳細的微調建議細節，請見〈節 5.3.7, “為效能設定檔案系統”〉。

**XFS**

XFS 是一個強大而且高度可調式的 64 位元檔案系統。它是 Red Hat Enterprise Linux 7 預設的檔案系統。XFS 使用的是基於分割的檔案分配，而且以許多分配配置為特徵，包含預先分配以及延遲的應用程式。兩者皆會減少分散提升效能。它也會支援中繼資料日誌紀錄，這樣子可以協助毀損復原。XFS 在被掛載以及使用中時可以被重組以及放大。Red Hat Enterprise Linux 7 支援一些特定是 XFS 備份以及還原的公用程式。

對 Red Hat Enterprise Linux 7.0 GA 來說，XFS 會被支援至一個 500 TB 的最大檔案系統以及 8 EB 的最大檔案位移 ( 疏鬆檔案 ) 。欲了解管理 XFS 的細節，請見「*Red Hat Enterprise Linux 7 儲存系統管理員指引*」，連結為 http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。欲尋求微調 XFS 的協助，請見〈節 5.3.7.1, “微調 XFS”〉。

**Ext4**

ext4 是 ext3 檔案系統的可調式新版本。它的預設行為對大部分的工作負載來說是最好的。然而，它只會被支援至 50 TB 大小的最大檔案系統以及 16 TB 大小的最大檔案。欲了解管理 ex4 的細節，請見「*Red Hat Enterprise Linux 7 儲存系統管理員指引*」。請至以下網址取得：〈http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/〉。欲尋求微調 ext4 的協助，請見〈節 5.3.7.2, “微調 ext4”〉。

請注意，ext4 的叢集配置特徵 ( bigalloc ) 目前並沒有經過測試或是在 Red Hat Enterprise Linux 7 被支援。

**Btrfs (技術預覽 )**

Btrfs 是一個寫入時複製的檔案系統，它提供延展性、容錯，以及輕鬆的管理。它包含內建的快照與 RAID 支援。它使用資料以及 metadata 的總和檢查碼來提供資料完整性。它也使用資料壓縮來改善效能以及更有效地使用空間。作為技術預覽（Technology Preview），它被支援至一個 50 TB 大小的最大檔案系統。

Btrfs 最適合桌面以及雲端儲存。當裝置一開始被格式化時，最好將 btrfs 微調來符合使用目的。

Red Hat Enterprise Linux 7 提供 btrfs 作為技術預覽。欲了解 Technology Preview 的特徵，請見〈https://access.redhat.com/site/support/offerings/techpreview/〉。

欲了解管理 btrfs 的細節，請見「*Red Hat Enterprise Linux 7 儲存系統管理員指引*」。您可以從以下連結取得：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。欲尋求微調 btrfs 的協助，請見〈節 5.3.7.3, “微調 btrfs”〉。

**GFS2**

GFS2 是 High Availability Add-On ( 高可用性附加元件）的一部分。它提供叢集檔案系統支援 Red Hat Enterprise Linux 7。GFS2 在全部的伺服器提供一個一致的檔案系統映像，允許伺服器從單一共享的檔案系統閱讀以及寫入。

GFS2 支援至 250 TB 大小的最大檔案系統。

欲了解管理 GFS2 的細節，請見「*Red Hat Enterprise Linux 7 儲存系統管理員指引*」。請從以下連結取得：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。欲尋求微調 GFS2 的協助，請見〈節 5.3.7.4, “微調 GFS2”〉。

## 5.1.4. 檔案系統的一般微調設定

本小節涵蓋所有檔案系統常見的微調設定。欲了解針對您的檔案系統的微調建議，請見〈節 5.3.7, “為效能設定檔案系統”〉。

### 5.1.4.1. 格式化時間的考量

有一些檔案系統的設定決定不能在裝置被格式化之後被改變。本章節涵蓋一些可以使用的選項，這些選項是您在格式化儲存裝置之前必須做的決定。

**大小**

若要為您的工作負載建立適當大小的檔案系統，要知道的是，較小的檔案系統依比例來說需要較短的備份時間，而且需要較少時間以及記憶體來執行檔案系統檢查。然而，若您的檔案系統過小，那麼它的效能將會因為高度分散而減低。

**區塊大小**

區塊為檔案系統工作的單位。區塊大小將決定有多少資料可以被儲存於單一區塊之中，因此可以決定同一時間最小量可以被寫入或讀取的資料。

在預設情況之下，區塊的大小是很適合大多數使用案例的。但是如果區塊大小 ( 或者是多重區塊的

大小 ) 與同一時間內讀取或寫入的資料數量一樣，或者比它稍為大一些的話，您的檔案系統將有更好的效能以及更有效地儲存資料。小的檔案將會占用一整個區塊。檔案可以被分散至多重區塊，但是這將建立其他執行階段的額外負荷。另外，有一些檔案系統會受限於特定區塊的數目，而這將會限制檔案系統的大小上限。

當您使用「**mkfs**」指令格式化一個裝置時，區塊大小會被指定為檔案系統選項的一部分。指定區塊大小的參數會隨著檔案系統有所不同。欲了解細節，請見您的檔案系統的「**mkfs**」man page。舉例來說，欲了解格式化 XFS 檔案系統時可用的選項，請執行以下指令。

```
$ man mkfs.xfs
```

**幾何分佈**

檔案系統幾何分佈與跨列檔案系統的資料發佈有關。如果您的系統使用等量儲存空間，例如 RAID，那麼您可以透過以下方式改善效能：在格式化裝置時使用基礎儲存空間幾何分布來對齊資料和中繼資料。

許多裝置會匯出推薦的幾何分佈，然後會在裝置與特定的檔案系統被格式化時自動被設定。如果您的裝置不匯出這些推薦項目，或者您想要變更推薦的設定，那麼您必須在用「**mkfs**」格式化裝置時手動指定幾何分佈。

指定檔案系統幾何的參數會隨著檔案系統有所不同。欲了解細節，請見您的檔案系統的「**mkfs**」man page。舉例來說，欲了解在格式化 ext4 檔案系統時可用的選項，請執行以下指令。

```
$ man mkfs.ext4
```

**外部日誌**

以日誌記錄檔案系統可以記錄在日誌檔案內執行寫入作業（在作業被執行前）會做的變更。這減少了儲存裝置在系統當機或電源失敗時毀損的可能性，並且也會加速復原。

metadata高用量的工作負載與日誌的頻繁更新有關。較大的日誌使用較多記憶體，但是減少寫入作業的頻率。另外，您可以透過以下方法用中繼資料加護的工作負載改善裝置的搜尋時間：將工作負載的日誌放置在跟主要儲存空間一樣快，或者比儲存間更快的專用儲存空間。

**警告**

請確保外部日誌是穩定的。失去一個外部日誌裝置會導致檔案系統毀損。

外部日誌一定要建立於格式化時，而且也要在掛接時指定日誌裝置。欲了解細節，請見**mkfs** 和 **掛載** 的 man page。

```
$ man mkfs
```

```
$ man mount
```

### 5.1.4.2. 掛載時間設定

本小節包含可套用於大部分檔案系統的微調決策，以及在裝置被掛載（mount）時可以被指定的微調決策。

**屏障**

檔案系統屏障可以確保檔案系統的中繼資料被正確寫入以及被安置於持續性儲存區，以 以確保「**fsync**」傳輸的資料即使電源中斷了也會持續。之前的 Red Hat Enterprise Linux 版本中，啓動檔案系統屏障會明顯減慢大幅度依賴「**fsync**」的應用程式，或者建立、刪除許多較小的檔案。

在 Red Hat Enterprise Linux 7 中，檔案系統屏障的效能已獲得明顯改善，停用檔案系統屏障的效能效果非常少（少於 3 %）。

欲了解更多資訊，請見「*Red Hat Enterprise Linux 7 儲存系統管理員指引*」。請從以下網址取得：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

**存取時間**

每當一個檔案被讀取時，它的 metadata 會在被存取時更新（「**atime**」）。這包含了其它 I/O 寫入。大部分的案例中，這種額外負荷是很小的。在預設情況下，Red Hat Enterprise Linux 7 只有在之前的存取時間比上一次修改次數（「**mtime**」）更早時，或是狀態變更時（「**ctime**」）會更新「**atime**」。

然而，如果更新 metadata 很耗時間，而且並沒有要求準確的存取時間，您可以用「**noatime**」掛接選項來掛接檔案系統。在一個檔案被讀取時，這將停用更新至 metadata。它也將啓用「**nodiratime**」行為。此行為在一檔案被讀取時會停用更新至 metadata。

**預先讀取**

預先讀取行為會透過以下方式加快檔案存取：預先提取有可能會被需要的資料以及將資料載入頁面快取；和我們的磁碟相比，資料在這裡可以更快的被取出。預先讀取的數值越高，系統就可以越快預先提取資料。

Red Hat Enterprise Linux 嘗試設立一個適當的預先讀取數值，是以它在您的檔案系統偵測到什麼為基礎。但是不太可能每次都能準確的偵測。舉例來說，如果一個儲存陣列將自己以單一 LUN 呈現給系統，系統偵測到單一 LUN 而且不為陣列設定適當的預先讀取數值。

與循序 I/O 的重負荷串流有關的工作負載通常從高的預先讀取數值得到效益。Red Hat Enterprise Linux 7 提供的與儲存空間相關的微調檔案會提高預先讀取的數值；使用 LVM 分散讀寫也會有一樣的效果。但是這些調整對所有工作負載來說不總是完全足夠。

定義預先讀取行為的參數會隨著檔案系統有所不同；欲了解細節請見 mount man page。

```
$ man mount
```

### 5.1.4.3. 維護

我們建議您定期捨棄沒有被檔案系統使用的區塊，這對固態磁碟以及精簡佈建的儲存空間來說都是好的。有兩種捨棄沒有被使用的區塊的方法：批次捨棄與線上捨棄。

**批次捨棄**

批次捨棄是「**fstrim**」指令的一部分。它捨棄符合系統管理員指定準則的檔案系統內所有沒有被使用的區塊。

Red Hat Enterprise Linux 7 在 XFS 和 ext4 支援實體捨棄作業而且格式化的裝置上支援批次捨棄（意思是，在 HDD 裝置上「**/sys/block/*devname*/queue/discard_max_bytes**」的數值不是零，而且 SSD 裝置上，「**/sys/block/sda/queue/discard_granularity**」的數值不是「**0**」）。

**線上捨棄**

線上捨棄作業是在掛接時和「**discard**」選項一起被設定的，而且會在沒有使用者中斷的狀況下即時執行。但是，線上捨棄只會捨棄從使用中切換至空閒狀態的區塊。Red Hat Enterprise Linux 7 在 XFS 以及 ext4 格式化的裝置上支援線上捨棄。

Red Hat 建議使用批次捨棄，除了需要線上捨棄來維持效能，或者批次捨棄不適合系統的工作負載時。

**預先配置**

預先配置會在沒有將任何資料寫入檔案空間的情況下將磁碟空間標示為已被配置到某個檔案。這對於限制資料分散以及較差讀取效能來說是很有幫助的。Red Hat Enterprise Linux 7 在 XFS，ext4 和 GFS2 裝置的掛接時間時支援預先配置。請見「**mount**」man page 來了解適合您檔案系統的參數。應用程式也可以從使用「**fallocate(2) glibc**」呼叫預先配置空間而得到效益。

## 5.2. 監視以及診斷效能問題

Red Hat Enterprise Linux 7 提供許多工具，這些工具對監視系統效能，以及診斷與 I/O 、檔案系統和它們的設定相關的效能問題來說是很實用的。本章節概述您可以使用的工具，以及舉例說明如何使用它們來監視，並診斷 I/O 和與檔案系統相關的效能問題。

### 5.2.1. 用 vmstat 監視系統效能

vmstat 報告整個系統的處理序、記憶體、分頁、I/O 區塊、插斷以及 CPU 活動。它可以幫助系統管理員來決定 I/O 子系統是否要為任何效能問題負責。

以下欄位顯示的是與 I/O 效能相關程度最大的資訊：

**si**

swap in 的簡寫，即寫入 swap 空間；單位為 KB。

**so**

swap out 的簡寫，即取讀自 swap 空間；單位為 KB。

**bi**

block in 的簡寫，即 block 寫入作業；單位為 KB。

**bo**

block out 的簡寫，即 block 取讀作業；單位為 KB。

**wa**

等待 I/O 作業完成的佇列比例。

當您的 swap 空間和資料位於一樣的裝置上，swap in 以及swap out 就特別有用，可以視為記憶體使用方式的指標。

此外，free，buff，與 cache 欄位可以幫助識別寫回頻率。快取數值突然的下降以及空閒數值的增加表示寫回以及頁面快取無效判定已經發生。

如果用 **vmstat** 分析，會顯示 I/O 子系統跟效能的下降是有關係的。系統管理員可以使用 **iostat** 來決定該負責的 I/O 裝置。

「**vmstat**」 是由 「*procps-ng*」提供的套件。欲了解使用「**vmstat**」的細節，請見 man page：

```
$ man vmstat
```

### 5.2.2. 用 iostat 監視 I/O 效能

「**Iostat**」是由「*sysstat*」 套件提供。它報告您的系統的 I/O 裝置載入。如果用「**vmstat**」的分析顯示 I/O 子系統和效能減低有關係，那麼您可以使用「**iostat**」來決定要負責的 I/O 裝置。

您可以專注在「**iostat**」對特定裝置的報告輸出上，方法是使用「**iostat**」man page 中定義的參數。

```
$ man iostat
```

#### 5.2.2.1. 用 blktrace 執行詳細的 I/O 分析

「**Blktrace**」提供在 I/O 子系統是如何分配時間的詳細資料。小幫手公用程式「**blkparse**」讀取「**blktrace**」的原始輸出，而且產生人們可讀取的輸入摘要以及被「**blktrace**」記錄的輸出作業。

欲了解更多使用此工具的資訊，請見 man page：

```
$ man blktrace
```

```
$ man blkparse
```

#### 5.2.2.2. 用 btt 分析 blktrace 輸出

「**Btt**」是「*blktrace*」套件的一部分。它可以分析「**blktrace**」輸出以及顯示資料在每一個 I/O 堆疊空間花了多少時間。這會讓 I/O 子系統的瓶頸較容易被發現。

舉例來說，如果「**btt**」顯示被傳送至區塊層（「**Q2Q**」）的要求與要求之間的時間比要求在區塊層（「**Q2C**」）所花的總時間還要長，那麼效能問題就與 I/O 子系統無關。如果一個裝置花很長的時間服務要求（「**D2C**」）， 那麼這個裝置的負載會過重，或者傳送至裝置的工作負載可能不會是最佳狀況。如果 I/O 區塊在被指派要求（「**Q2G**」）之前排隊了很長的時間，這可能表示使用中的儲存空間無法服務 I/O 負載。

欲了解更多使用此工具的資訊，請見 man page：

```
$ man btt
```

#### 5.2.2.3. 用 seekwatcher 分析 blktrace 輸出

「**seekwatcher**」工具可以使用「**blktrace**」輸出來製圖 I/O 逾時。它聚焦於 I/O 磁碟的 Logical Block Address（LBA, 邏輯區塊位置 ）， 每秒輸送 MB、每秒搜尋的數量以及每秒 I/O 作業。這可以幫助您識別您的裝置何時達到每秒作業上限。

欲了解更多使用此工具的資訊，請見 man page：

```
$ man seekwatcher
```

### 5.2.3. 用 SystemTap 監視儲存空間

「*Red Hat Enterprise Linux 7 系統 tap 初學者指南*」包含幾個範例指令碼。這些指令碼對設定檔以及監視儲存空間效能來說是很有用的。

以下「**SystemTap**」範例指令碼和儲存空間效能有關聯，而且在診斷儲存空間或是檔案系統效能問題方面可能會有幫助。在預設情況下，它們被安裝在「**/usr/share/doc/systemtap-client/examples/io**」目錄裡。

**disktop.stp**

每隔五秒它會檢查磁碟的讀取或是寫入狀態，以及輸出在那段期間的前十個輸入。

**iotime.stp**

它會列印花在讀取和寫入作業的時間總量，以及讀取、寫入的位元組數量。

**traceio.stp**

每秒它會列印前十個可執行的檔案，是依據發現的累計 I/O 流量。

**traceio2.stp**

在讀取和寫入至一個指定的裝置發生時，它會列印可執行檔的名字以及處理識別碼。

**inodewatch.stp**

每一次一個讀取或寫入發生在指定的主要 / 次要裝置上的 inode 時，它會列印可執行檔的名字以及處理識別碼。

**inodewatch2.stp**

它會列印可執行檔名字、處理識別碼以及在每一次指定的主要 / 次要裝置上的 inode 的屬性變更時列印屬性。

「**latencytap.stp**」指令碼記錄不同種類的延遲對一個或多個處理序的效果。它每三十秒會列印一個延遲種類的列表，依一個處理序或多個處理序花在等待的總時間的遞減順序來排序。這對於識別儲存空間以及網路延遲的原因是很有用的。Red Hat 建議您使用「**--all-modules**」選項與這個指令碼來啓用延遲事件的對應，這樣效果會比較好。在預設狀況下，此指令碼是安裝在「**/usr/share/doc/systemtap-client-*version*/examples/profiling**」目錄。

「*Red Hat Enterprise Linux 7 系統 tap 初學者指引*」可以從以下連結取得：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

## 5.3. 設定工具

Red Hat Enterprise Linux 提供一些工具來輔助系統管理員設定儲存空間以及檔案系統。本小節將概述可以使用的工具以及提供例子說明要怎麼使用它們來解決 I/O 和與 Red Hat Enterprise Linux 7 內效能問題相關的檔案系統。

### 5.3.1. 設定微調檔案增強儲存空間效能

「**Tuned**」以及「**tuned-adm**」提供一些為了特定使用案例設計來改善效能的檔案。以下檔案對改善儲存空間效能特別有用。

- 延遲效能
- 吞吐效能 ( 預設狀況下 )

要在您的系統上設定一個設定檔，請執行以下指令，用您想使用的設定檔檔名來取代「*name*」。

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

「**tuned-adm recommend**」指令為您的系統建議適合的設定檔。這也會在開機時為您的系統設定預設設定檔，所以它可以被用來傳回至預設設定檔。

欲了解這些設定檔或是附加設定的選項的細節，請見〈節 A.6, “tuned-adm”〉。

### 5.3.2. 設定預設 I/O 排程器

預設的 I/O 排程器是在一個裝置的掛接選項裡，沒有指定排程器時所使用的排程器。

要設定預設的 I/O 排程器，請透過以下方式指定您想使用的排程器：附加電梯參數至 kernel 命令列，無論是在開機時或是透過編輯「**/etc/grub2.conf**」檔案。

```
elevator=scheduler_name
```

### 5.3.3. 裝置設定 I/O 排程器

要設定排程器或是設定一個特定儲存空間裝置的排程器喜好順序，請編輯「**/sys/block/*devname*/queue/scheduler**」檔案，而「*devname*」是您想要設定的裝置名稱。

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

### 5.3.4. 微調排程器期限

當「**deadline**」在使用中，排入佇列的 I/O 要求會被分類為讀取或寫入批次，然後被排程執行以增加 LBA 順序。在預設狀況下，讀取批次比寫入批次優先，因為應用程式在讀取 I/O 時較有可能被阻擋。在一個批次處理完成後，「**deadline**」會檢查寫入作業已被占用了多久的處理器時間，然後排程下一個適合的讀取或是寫入批次。

以下參數會影響「**deadline**」排程器的行為。

**fifo_batch**

在單一批次發行的讀取或寫入作業數量。預設值為「**16**」。較高的數值可以增加輸送，但是也會增加延遲。

**front_merges**

如果您的工作負載不會產生前端合併，那麼這個可調數值可以被設為「**0**」。然而，除非您測量過這個檢查的額外負荷，否則 Red Hat 建議您將預設值設為「**1**」。

**read_expire**

讀取要求應該要排程給服務使用的毫秒數量。預設值為「**500**」（0.5 秒）。

**write_expire**

寫入要求應該要排程給服務使用的毫秒數量。預設值為「**500**」（0.5 秒）。

**writes_starved**

在處理寫入批次之前可以被處理的讀取批次數量。數值設定的越高，讀取批次得到的偏好設定就越大。

### 5.3.5. 微調 cfq 排程器

當「**cfq**」在使用中，處理序被放置在三個類別裡：即時、 盡力以及閒置。所有即時程序都比盡力程序先排程；它也比任何閒置程序先被排程。在預設情況下，程序被分類為盡力類別，您可以用「**ionice**」命令手動調整處理序的分類。

您可以使用以下參數進一步調整「**cfq**」排程器的行為。這些參數是透過調整在「**/sys/block/*devname*/queue/iosched**」目錄之下指定的檔案而設立於每一個裝置上的。

**back_seek_max**

「**cfq**」將會執行反向搜尋的最大距離，單位為 KB。預設值為 「**16** KB」。反向搜尋通常會損害效能，所以我們不建議設定大數值。

**back_seek_penalty**

這是當磁碟前端正在決定要向前還是向後移動時，套用於反向搜尋的乘數。預設值為「**2**」。如果磁碟前端位置是在「1024 KB」，而且系統內（舉例來說，「1008 KB and 1040 KB」）有等距離的要求，「***back_seek_penalty***」會套用於反向搜尋，磁碟會向前移動。

**fifo_expire_async**

非同步（已緩衝寫入）要求可以保持無服務的時間長度，單位為 KB。逾期時，單一急需非同步要求會移到分派的清單上。預設值為「**250**」毫秒。

**fifo_expire_sync**

同步（讀取或「**O_DIRECT**」寫入）要求可以保持無服務的時間長度，單位為 KB。逾期時，單一 starved 同步要求會移到分派清單。預設值為「**125**」毫秒。

**group_idle**

在預設狀況下，這個參數被設為「**0**」（停用）。當被設為「**1**」（啓用），「**cfq**」排程器會在上一個在控制群組裡發行 I/O 的處理序閒置。這個在您使用等比例權數 I/O 控制群組時，以及「***slice_idle***」被設為「**0**」（快速儲存空間上）的時候是很有用的。

**group_isolation**

這個參數在預設情況下被設為「**0**」（停用）。當被設為「**1**」（啓用），它會在不同群組之間提供較強的隔離，但是會減少輸送，因為 fairness 套用於隨機以及循序工作負載。當「***group_isolation***」被停用（被設為「**0**」），fairness 只會套用於循序工作負載。欲了解更多資訊，請見安裝於「**/usr/share/doc/kernel-doc-version/Documentation/cgroups/blkio-controller.txt**」的文件。

**low_latency**

這個參數在預設情況下被被設為「**1**」（啓用）。啓用時，「**cfq**」藉由以下方式偏好 fairness 多過輸送：為每一個在裝置上發行 I/O 的處理序提供最大等待時間「**300** ms」。當這個參數被設為「**0**」（停用），目標延遲會被忽略，而且每一個處理序會接收完整時間配量。

**quantum**

這個參數決定「**cfq**」一次傳送到一個裝置的 I/O 要求數量，基本上會限制佇列深度。預設值為「**8**」個要求。使用中的裝置可以支援更大的佇列深度，但是增加配量數值也會增加延遲，尤其是對於大的循序寫入工作負載。

**slice_async**

這個參數決定分配到每一個發出非同步 I/O 要求的處理序的時間配量長度 ( 以毫秒計算 )。預設值為「**40**」毫秒。

**slice_idle**

此參數指定毫秒內 cfq 在等待其他要求時閒置的時間長短。預設值為「**0**」（佇列或是服務樹狀等級沒有閒置）。預設值對外部 RAID 儲存空間輸出來說是很理想的，但是它會降低內部非 RAID 儲存空間的輸出，因為它會增加搜尋作業的整體數量。

**slice_sync**

此參數決定分配到每一個發出同步 I/O 要求的處理序時間配量（以毫秒計算）。預設值為「**100** ms」。

#### 5.3.5.1. 為了快速儲存空間微調 cfq

我們不建議在不受大型搜尋罰則影響的硬體上使用「**cfq**」排程器，例如快速外部儲存空間陣列或是固態硬碟。如果您要求「**cfq**」在這個儲存空間被使用，您必須編輯以下設定檔案：

- 請設定「**/sys/block/*devname*/queue/ionice/slice_idle**」至「**0**」
- 請設定「**/sys/block/*devname*/queue/ionice/quantum**」至「**64**」
- 請設定「**/sys/block/*devname*/queue/ionice/group_idle**」至「**1**」

### 5.3.6. 微調 noop 排程器

「**noop**」I/O 排程器主要有助於極度仰賴 CPU，且使用快速儲存量的系統。請求會合併在攔截層，因此「**noop**」行為的更改是藉由編輯「**/sys/block/sdX/queue/**」目錄中的攔截層參數進行。

**add_random**

一些 I/O 事件 對於 entropy 集區的「**/dev/random**」有所幫助。此參數能夠設定至「**0**」，若這些幫助的額外負荷變得可以計算。

**max_sectors_kb**

指定 I/O 請求的最大位元組（kilobytes）大小。預設值為「**512**」 KB。此參數的最小值是由儲存裝置的攔截大小來決定。此參數的最大值是由「***max_hw_sectors_kb***」的值來決定。

當 I/O 請求比內部消除攔截大小來的大，一些固態硬碟將執行不佳。在此情況下，Red Hat 建議降低「***max_hw_sectors_kb***」至內部消除攔截大小。

**nomerges**

請求的合併將有益於大多數的工作載量。然而，停用合併能夠有助於調適（除錯）目的。請設定此參數至「**0**」以停用合併。 在預設情況下，參數是啓動的（設定至「**1**」）。

**nr_requests**

指定讀取與寫入請求一次能夠排入佇列的最大數量。預設值為「**128**」；亦即 128 個讀取請求與 128 個寫入請求能夠在下一個程序之前被排入佇列，以請求將一個讀取或寫入進入睡眠狀態。

對於延遲敏感的應用程式，請降低此參數值並限制指令佇列在儲存量的深度，以便返寫 I/O 不會用寫入請求佔用裝置佇列。裝置佇列佔滿時，其他程序嘗試去執行 I/O 的作業將會被設定進入睡眠狀態，直至有佇列空間。接著，請求會以循環的方式被分配，以避免程序持續消耗所有佇列空間。

**optimal_io_size**

一些儲存裝置通過此參數報告一個最佳 I/O 大小。若此值被提出，Red Hat 建議應用程式盡早配發 I/O，此 I/O 是相應和最佳 I/O 大小的倍數。

**read_ahead_kb**

決定作業系統在一個循序讀取的作業中將預先讀取的 KB（kilobyte）數量，以儲存短期內在頁面快取可能需要的資訊。通常，一個高的「***read_ahead_kb***」值將有助於裝置對應程式；若各個裝置都對應到 128 KB 將會是好的起始點。

**輪調**

一些固態硬碟不會正確地告知其固體狀態，且以傳統輪調硬碟掛載。若您的硬體裝置沒有自動將此設定至「**0**」，請自行設定，以停止在排程器中非必要的搜尋減低邏輯。

**rq_affinity**

在預設情況下，I/O 完成的項目能夠在不同的處理器接受處理，並不一定需要在發行 I/O 指示的處理器中接受處理。請設定「***rq_affinity***」至「**1**」以停用此功能，並且僅在發出 I/O指示的處理器中執行完成項目。此項動作能夠改善處理器資料快存的效率。

## 5.3.7. 為效能設定檔案系統

本章節涵蓋各檔案系統特定的微調參數，以支援 Red Hat Enterprise Linux 7。參數的分類是依照其值是否應該在您格式化儲存裝置時設定，或者當您掛載格式化裝置時設定。

當效能降低是由檔案分散或資源爭用而造成，效能通常能夠透過重新設定檔案系統而有所改善。然而在某些情況下，應用程式可能需要更改。在此情況下，Red Hat 建議您與客戶支援部聯繫，請求協助。

### 5.3.7.1. 微調 XFS

本章節涵蓋一些在 XFS 檔案系統格式化，和掛載時能夠使用的微調參數。

XFS 的預設格式化以及掛載設定，適用於大部分的工作載量。Red Hat 建議您僅在特定設定更改有益於您的工作載量時才做出更動。

#### 5.3.7.1.1. 格式化選項

更多有關以上格式化選項的詳細資訊，請參閱 man page：

```
$ man mkfs.xfs
```

**目錄區塊大小**

目錄區塊大小會影響每 I/O 作業能夠擷取或更改的目錄資訊量。目錄區塊大小的最小值為檔案系統區塊大小（在預設情況下為 4 KB）。目錄區塊大小的最大值為「**64**」 KB。

在既定目錄區塊大小之下，一個較大的目錄比較小的目錄需要更多的 I/O。相較於有較小的目錄區塊大小的系統，有較大的目錄區塊大小的系統也會消耗更多每 I/O 作業的處理能源。因此，我們建議您盡可能地為您的工作載量使用較小的目錄和目錄區塊大小。

當檔案系統的寫入密集與讀取密集工作載量不超過以下列出的輸入項目數量，Red Hat 建議以下的目錄攔截大小：〈表格 5.1, “目錄區塊大小的建議最大目錄輸入項目”〉。

**表格 5.1. 目錄區塊大小的建議最大目錄輸入項目**

| 目錄區塊大小 | 最大輸入項目（讀取密集） | 最大輸入項目（寫入密集） |
|---|---|---|
| 4 KB | 100000–200000 | 1000000–2000000 |
| 16 KB | 100000–1000000 | 1000000–10000000 |
| 64 KB | >1000000 | >10000000 |

如需關於在不同大小的文件系統內目錄區塊大小對讀取和寫入工作負載的影響的詳細信息，請參考 XFS 文件。

請使用 **mkfs.xfs -l** 選項來配置目錄區塊大小。如需詳細信息，請參考 **mkfs.xfs** 主頁。

**分配組**

一個分配組是一個獨立結構，為文件系統的一區塊的自由空間編索引和分配內節點。每個分配組可以被獨立修改，從而允許 XFS 運行分配和接觸配置的操作，只要，與此同時，同時發生的操作影響不同的分配組。因此，在文件系統中能被運行的同時發生的操作數量等於分配組的數量。然而，既然運行同時發生的操作的能力受限於能運行這些操作的處理程序的數量，Red Hat 推薦分配組的數量大於或等於系統內處理程序的數量。

一個單一目錄不能同時被多個分配組修改。因此 Red Hat 建議創建和移除大量文件的應用程序不要將所有文件儲存至一個目錄中。

請使用 **mkfs.xfs -d** 選項配置分配組。如需詳細信息，請參考 **mkfs.xfs** man page。

**擴展限制**

在設置格式後，您可以擴大您文件系統的大小 （可以透過增加硬件，或通過自動精簡配置）， 您必須仔細地考慮初始文件佈局，因為在格式設定之後，分配組的大小將不能更改。

分配組的大小必須按照文件系統最終容量設置。在一個完全擴展開來的文件系統中，分配組的數量不應超過數萬，除非分配組是在最大值（1 TB）。因此，對於大多數文件系統，對於一個文件系統，推薦的允許最大擴展量為初始大小的 10 倍。

擴展以 RAID 陳列的文件系統，需要格外小心，因為裝置大小必須和多個分配組的大小一致，這樣新的分配組標頭才能正確地與新加的儲存一致。新儲存必需於現有的儲存的幾何一樣，因為在格式設置後，幾何是不能被更改的，因此，在同一區塊裝置上，不能優化不同幾何結構的儲存。

**inode 大小和行內屬性**

如果 inode 有足夠可用的空間，那麼 XFS 可以直接寫入屬性名稱和值到內節點。這些行內屬性能夠被擷取和修改至一個比個別屬性區域快的數量級，因為不需要其他的 I/O。

inode 大小的預設值為 256 bytes（位元組）。只有在此值的 100 bytes （位元組）內才能使用屬性儲存，這取決於存在 inode 內的數據程度指示器。在您設置能增加儲存屬性可使用的空間量的文件系統格式時，增加 inode 大小。

屬性名稱和屬性值都受限於最大值 254 bytes。如果名稱或值長度上超過了254 bytes，屬性將會被推回到單獨的屬性區塊，而不是儲存到內聯裡。

請使用 **mkfs.xfs -i** 選項配置內節點參數。如需詳細信息，請參考 **mkfs.xfs** 主頁。

**RAID**

如果軟件 RAID 在使用中，**mkfs.xfs** 會自動為配置底層硬件合適的磁條單元和寬度。但是，如果硬件 RAID 在使用中，條形單元和寬度可能需要手動配置，因為不是所有的硬件 RAID 設備都輸出此信息。請使用 **mkfs.xfs -d** 選項配置條形單元和寬度。如需詳細信息，請參考 **mkfs.xfs** 主頁。

**日誌大小**

擱置更改是彙總至記憶體中，直至同步處理事件被啓動，在此時間點擱置更改會被編寫至日誌。日誌大小決定一次可以處理的並行修改數量。日誌大小也決定可以被彙總至記憶體的最大更改量，以及日誌資料被編寫至硬碟的頻率。相較於一個較大的日誌，較小的日誌將使資料更頻繁地被編寫回硬碟。然而，一個較大的日誌使用較多的記憶體去紀錄擱置修改，因此有著有限的記憶體的系統將不會受益於較大的日誌。

當日誌調整至與基本等量硬碟區單位一致，日誌會有較佳的效能。亦即，日誌在等量硬碟區單位界線開始和結束。欲調整等量硬碟區單位，請使用「**mkfs.xfs -d**」選項。更多詳情，請參閱「**mkfs.xfs**」man page 。

如想配置日誌大小，請使用以下 **mkfs.xfs** 選項，用日誌的大小替換 *logsize*：

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

欲知詳情，請參考 **mkfs.xfs** 主頁：

```
$ man mkfs.xfs
```

**將條帶單元寫入日誌**

寫在使用 RAID5 或 RAID6 佈局的儲存裝置的日誌可能運行的更好，如果儲存裝置起始並終結與條帶單元界（ 與基本條帶單元對齊。**mkfs.xfs** 試圖設置自動設置適當的日誌條帶單元，但這取決於 RAID 裝置輸出此信息。

如果您的工作負載引起同步活動非常頻繁，設置大的條帶單元可能會損害性能，因為較小的寫入需要填補日誌條帶單元的大小，而這會增加等待時間。如果您的工作負載受日誌寫入延遲的約束，Red Hat 推薦設置條帶單元為 1 區塊，如此能盡可能引起未調準的日誌寫入。

支持的日誌條帶單元最大大小為日誌緩衝區的最大值。因此， 有可能底層存儲可能有較大條帶單元可被配置到日誌上。此情況下，**mkfs.xfs** 發出警告並設置日誌條帶單元為 32 KB。

使用以下選項中的一個來配置日誌條帶單元，*N* 是區塊數值，作為條帶單元來使用， *size*是以 KB 為單元的條帶單元大小。

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

欲知詳情，請參考 **mkfs.xfs** 主頁：

```
$ man mkfs.xfs
```

#### 5.3.7.1.2. 掛載選項

**inode 配置**

對於大小大於 1 TB 的文件系統，極力推薦。***inode64*** 參數配置 XFS 以分配整個文件系統內的 inode 和資料。這確保大部分 inode 不被分配到文件系統的開端，大部分數據不被分配到文件系統的末端，從而提高大文件系統的性能。

**日誌緩衝區的大小和數量**

日誌緩衝區越大，寫入日誌所有改變所用的 I/O 的操作越少。較大的日誌緩衝區可以改善系統性能，並且 I/O 密集的工作負載沒有不穩定的寫入緩存。

日誌緩衝區的大小是用 ***logbsize*** 掛載選項來配置的，並決定能被存入日誌緩衝區的最大資訊量；如果日誌條帶單元為被設定，緩衝區寫入可能比最大的短，因此，無需減少為同步大工作量而減少日誌緩衝區大小。日誌緩衝區預設大小為 32 KB，最大為 256 KB，其他支持的大小為 64 KB、128 KB，或在 32 KB 和 256 KB 之間日誌條形單元的二倍數。

日誌緩衝區的數量由 ***logbufs*** 掛載選項來設定。預設值為 8 個日誌緩衝區（最多），但是，最少要配置兩個日誌緩衝區。通常不需要減少日誌緩衝區的數量，除非是在無法承擔將內存分配到另外的日誌緩衝區的有內存限度的系統。減少日誌緩衝區的數量通常會減少日誌性能，特別是對於日誌 I/O 時延敏感的工作負載。

**延遲紀錄更改**

XFS 有選項可以在改變寫入內存前，合計內存改變。***delaylog*** 參數允許經常定期地更改被寫入到日誌中的 metadata，而不用每次更改時修改。此選項增加系統當機時失去作業的潛在數量，也增加用於追蹤 metadata 的記憶體使用量。然而，透過範圍順序，此選項亦增加 matadata 修改速度與可擴展性，且在使用**fsync**、**fdatasync**或 **sync** 來確保資料和 metadata 被寫入硬盤時，不會降低數據或 metadata 的完整性。

### 5.3.7.2. 微調 ext4

本節包含設定ext4 文件系統的格式和掛載時可參考的調節參數。

#### 5.3.7.2.1. 格式化選項

**inode 表初始化**

在非常大的文件系統上，初始化文件系統中的所有 inode 會需要很長時間。在預設情況下，初始化過程會被推遲 （懶惰 inode 表 - lazy inode table - 被啓動）。然而，如果您的系統沒有 ext4 驅動程序，懶惰內節點表初始化將被預設為禁用。可以通過設置 ***lazy_itable_init*** 為 **1** 來啓用。在此情況下，內核進程繼續在文件系統掛載後初始化文件系統。

本節僅描述了一些設置格式時可使用的一些選項。如需瞭解更多格式設置參數，請參考 **mkfs.ext4** 主頁：

```
$ man mkfs.ext4
```

#### 5.3.7.2.2. 掛載選項

**inode 初始化速度**

如果懶惰內節點表初始化已啓用，您可以通過指定 ***init_itable*** 參數的值來控制初始化的速度。運行背景初始化的所用的時間量大約等於 1，由此參數值劃分。預設值為**10**。

**自動文件同步**

一些應用程序在重命名、或刪減、重寫現存文件後不會正確進行**fsync**。ext4 預設情況下會自動在每個操作後同步文件。但是，這可能會耗費時間。

如果不需要此等級同步，您可以在掛載時指定 **noauto_da_alloc** 選項以停用此操作。如果 **noauto_da_alloc** 已設定，應用程序必須明確使用 fsync 以保證數據持久性。

**日誌 I/O 優先級**

在預設情況下，日誌 I/O 優先級為 **3**，比正規 I/O 的優先級稍高。您可以在掛載時用 ***journal_ioprio*** 參數來控制日誌 I/O 的優先級。***journal_ioprio*** 的有效值範圍為 **0** 到 **7**， **0** 為 I/O 最高優先級。

本節僅描述在掛載時刻可使用的一些選項。如需更多掛載選項，請參考 **mount** man page：

```
$ man mount
```

### 5.3.7.3. 微調 btrfs

關於 Red Hat Enterprise Linux 7.0, btrfs 僅為技術預覽。如果將來完全支持 btrfs， 本節將在未來版本中更新。

#### 5.3.7.4. 微調 GFS2

本節為 GFS2 文件系統的格式和掛載介紹一些可用的微調參數。

**目錄間隔**

GFS2 掛載點最高級目錄中所創建的目錄都會被自動以規定距離排列，以減少儲存殘片並增加這些目錄中的寫入速度。如想像最高級目錄一樣間隔另一目錄用 ***T*** 屬性標出目錄，如同所示，用你想間隔的目錄的目錄路徑替換 *dirname* ：

```
# chattr +T dirname
```

**chattr** 只是 *e2fsprogs* 程序包的一部分。

**減少競逐**

GFS2 使用能要求集群節點間溝通的全局鎖定機制。在多個節點間的文件和目錄衝突會降低性能。您可通過最小化多個節點間共享的文件系統空間將交叉緩存無效的風險降到最低。

# 章 6. 網路功能

網路功能子系統是由數個不同，且具有感應式連接的組件組成。Red Hat Enterprise Linux 7 網路功能的設計，是為最佳化大部份工作負載的效能和自動最佳化其效能。因此，網路效能並不一定需要手動微調。 本章節討論更多可以新增至功能性網路功能系統的最佳化。

有時，網路效能問題是硬體功能故障或基礎構造故障的結果。解決這些問題已超出本文討論的範圍。

## 6.1. 考量

為作出好的微調決定，您需要對 Red Hat Enterprise Linux 中的封包收發有全面的了解。本章節解釋網路封包如何被接收和產生，和潛在瓶頸可能發生的時候。

傳送至 Red Hat Enterprise Linux 系統的 packet 是由網路介面卡（NIC）接收，並置於內部硬體緩衝區或環形緩衝區。NIC 隨後傳送一硬體中斷請求，並提示軟體中斷運作的產生，以處理中斷請求。

身為軟體中斷運作的一部分，packet 是從緩衝區轉送至網路堆。根據 packet 和您網路的設定，packet 隨後會為應用程式而被轉寄、捨棄或傳遞至一 socket 接收佇列，或者從網路堆移除。此程序將會持續，直至 NIC 硬體緩衝區中沒有 packet，或者一定數量的被轉移 packet（指定於 **/proc/sys/net/core/dev_weight**）。

### 6.1.1. 在您微調之前

網路效能問題大多為硬體故障或基礎架構故障的結果。Red Hat 強烈建議在開始微調網路堆疊前，確認硬體和基礎架構皆如預期運作。

### 6.1.2. packet reception 中的瓶頸

雖然網路堆大多是自動最佳化， 但處理網路封包時有一些點將成為瓶頸並降低效能。

**NIC 硬體緩衝區或環形緩衝區**

若大量封包被卸除則硬體緩衝區可能成為瓶頸。更多有關監控您的系統所卸除的封包，請參閱〈節 6.2.4, “ethtool”〉。

**硬體或軟體中斷佇列**

中斷延遲延長且增加處理器爭用。更多有關處理器如何處理中斷的資訊，請參閱〈節 3.1.3, “IRQ 處理”〉。更多資訊關於如何於系統中監控中斷處理，請參閱〈節 3.2.3, “/proc/interrupts”〉。有關影響中斷處理的設定選項，請參閱〈節 3.3.7, “設定插斷親和性”〉。

**應用程式的通訊端接收佇列**

應用程式接收列佇列中的瓶頸是由大量的 packet 指出。這些 packet 並不會被複製至要求的應用程式，或透過增加**/proc/net/snmp** 中的 UDP 輸入錯誤（**InErrors**）。更多有關監控您系統上的這些錯誤，請參閱〈節 6.2.1, “ss”〉和〈 節 6.2.5, “/proc/net/snmp”〉。

## 6.2. 監控和診斷效能問題

Red Hat Enterprise Linux 7 提供許多工具，這些工具有助於監控系統效能和診斷有關網路系統的效能問題。本章節簡述可使用的工具和提供如何使用這些工具的範例，以監控和診斷與網路有關的效能問題。

### 6.2.1. ss

「**ss**」 是一個命令列公用程式，可以顯示有關通訊端的數據資訊，使系統管理員能夠隨時間取得裝置效能。在預設情況下，「**ss**」列出開放非聽取性 TCP 通訊端。這些通訊端建立連結，而有些提供的實用選項有助於系統管理員篩選出有關特定通訊端的數據。

Red Hat 建議使用「**ss**」，而不是使用 Red Hat Enterprise Linux 7 中的「**netstat**」。

「**ss**」是由「*iproute*」套件提供。 更多相關資訊，請參閱 man page：

```
$ man ss
```

### 6.2.2. ip

**ip** 公用程式使系統管理員能夠管理和監控路由、設置、路由原則和通道。**`ip monitor`** 命令能夠持續監控設置狀態、位址和路由。

**ip** 由 *iproute* 套件提供。更多有關使用 ip 的資訊，請參閱 man page：

```
$ man ip
```

### 6.2.3. dropwatch

**Dropwatch** 為一監控和紀錄由 kernel 棄置的 packet 的互動工具。

更多詳細資訊，請參閱 **dropwatch** 的 man page。

```
$ man dropwatch
```

### 6.2.4. ethtool

**ethtool** 公用程式使系統管理員能夠查閱並編輯網路介面卡設定。此工具有益於觀察特定設置的數據，例如：由設置所卸除的 packet 數量。

您可以查閱特定設置相對應的數據，透過使用 **`ethtool -S`**，和您想監控的裝置名稱。

```
$ ethtool -S devname
```

更多詳細資訊，請參閱 man page：

```
$ man ethtool
```

### 6.2.5. /proc/net/snmp

**`/proc/net/snmp`** 檔案為 IP、ICMP、TCP 和 UDP 的監控與管理，顯示 snmp 代理程式使用的數據。定期檢查此檔案能夠幫助系統管理員辨識異常值，因此辨識潛在效能問題。例如：增加的 UDP 輸入錯誤（**`InErrors`**）於 **`/proc/net/snmp`** 中，能夠表示通訊端接收佇列的瓶頸。

### 6.2.6. 以 SystemTap 監控網路

《*Red Hat Enterprise Linux 7 系統點選初學者指南*》 包含一些範例指令碼，有助於設定和監控網路效能。

以下 **SystemTap** 範例指令碼與網路功能相關，也可能有助於診斷網路效能問題。在預設情況下，這些指令碼是安裝在 **`/usr/share/doc/systemtap-client/examples/network`** 目錄中。

**`nettop.stp`**

此檔案每隔五秒會顯示程序列表（程序辨識器和命令），列表上會顯示寄出的 packet 數量和間隔之間由程序接收與寄出的資料數量。

**`socket-trace.stp`**

檢測 Linux kernel **`net/socket.c`** 檔案中的各個功能，並追蹤資料。

**`dropwatch.stp`**

每隔 5 秒鐘，檔案會顯示在 kernel 位置中釋放的通訊端緩衝。使用「**`--all-modules`**」選項以檢閱象徵性名稱。

**`latencytap.stp`** 指令碼紀錄不同類型的延遲的效果，這些效果發生在一個或更多程序上。此檔案每隔 30 秒會顯示一延遲類型的列表，而此列表是以降冪排序一個或更多程序消耗於等待的總時。此檔案有助於辨識儲存量和網路延遲的原因。Red Hat 建議以此指令碼使用 **`--all-modules`** 選項，以更好地啓用延遲項目的對應。在預設情況下，此指令碼是安裝於 **`/usr/share/doc/systemtap-client-`*`version`*`/examples/profiling`** 目錄中。

更多詳細資訊，請參閱《*Red Hat Enterprise Linux 7 系統點選初學者指南*》。網址為：〈http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/〉。

## 6.3. 設定工具

Red Hat Enterprise Linux 提供一系列工具，以協助系統管理員設定系統。本章節概述可使用的工具，以及提供這些工具可以如何運用，以解決 Red Hat Enterprise Linux 7 中有關網路效能的問題。

然而，重要的是知道網絡性能問題有時是由硬件故障或錯誤基礎結構導致的。Red Hat 極力建議在使用這些工具來調整網路堆疊前，檢查您硬件和基礎結構是正常運行的。

此外，透過更改應用程式比重新設定網路子系統，能夠更有效地解決一些網路效能問題。大體來說，設定您的應用程式以履行頻繁的 posix 呼叫是一個好的方法，因為這使資料能夠更有彈性的儲存，並根據所需在記憶體中調換，即使這意味著需要在應用程式空間佇列資料。

### 6.3.1. 網路效能的 Tuned-adm 設定檔

**tuned-adm** 提供一系列不同的設定檔，以改善一些特定使用案件的效能。以下設定檔有益於改善網路效能。

- 延遲效能
- 網路延遲
- 網路輸送量

更多有關以上設定檔的詳細資訊，請參閱〈 節 A.6, “tuned-adm”〉。

### 6.3.2. 設定硬體緩衝

若大量 packets 由硬體緩衝卸除，系統將有一些可能的解決方案。

**放慢輸入流量**

過濾傳入的流量、減少加入的群播群組或減少播放數量，以降低佇列的投放率。更多有關如何篩選輸入流量的詳細資訊，請參閱《*Red Hat Enterprise Linux 7 安全性指南*》。更多有關群播群組的詳細資訊，請參閱 Red Hat Enterprise Linux 7 叢集文件。更多有關播放流量的詳細資訊，請參閱《*Red Hat Enterprise Linux 7 系統管理員參考指南*》或與您想設定的設置有關的文件。所有

Red Hat Enterprise Linux 7 文件可以自以下網址取得：〈http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/〉。

**重新調整硬體緩衝佇列大小**

透過增加佇列大小，降低卸除的封包數量，使佇列不易溢位。您可以使用 ethtool 命令，修改網路裝置的 rx/tx 參數：

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

**更改佇列的流失率**

設置重量指的是一個設置一次能接收到的封包數量（於單一預定的處理器存取）。透過增加設置重量，佇列流失率會增加，而設置重量是由「***dev_weight***」參數管理。透過改變 **/proc/sys/net/core/dev_weight** 檔案的內容，此參數能夠暫時被更改，或永久地更改為「**sysctl**」，由 *procps-ng* 套件提供。

更改佇列流失率通常是降低不好的網路效能最簡單的方法。然而，增加一個裝置一次能接收的封包數量，會使用額外的處理器時間，且在這期間不能排程其他處理器，因此可能造成其他效能問題。

### 6.3.3. 設定中斷佇列

若分析顯示高度延遲，您的系統將能受益於以輪詢為基礎的封包回條，而非以中斷為基礎的封包回條。

#### 6.3.3.1. 配置忙碌輪詢

忙碌輪詢透過讓通訊端層代碼輪詢接收一網路裝置的佇列與停用網路中斷，以協助降低在網路接收路徑上的延遲。這將移除由中斷造成的延遲，和可能構成的內容切換。然而，這也增加了 CPU 使用量。忙碌輪詢也避免 CPU 處於休眠狀態而導致額外的能源消耗量。

忙碌輪詢在預設情況下是停用的。欲在特定通訊端啓用忙碌輪詢，請按以下步驟操作。

- 設定「***sysctl.net.core.busy_poll***」至一個非「**0**」的值。此參數將控制微秒數量，等待位於裝置佇列的封包用於通訊端輪詢和挑選。 Red Hat 建議「**50**」值。
- 增加「**SO_BUSY_POLL**」通訊端選項至通訊端。

欲整體性啓用忙碌輪詢， 您必須也設定「***sysctl.net.core.busy_read***」至一個非「**0**」的值。此參數將控制微秒數量，以等待位於裝置佇列上的封包用於通訊端取讀。此參數亦設定「**SO_BUSY_POLL**」選項的預設值。Red Hat 建議「**50**」值用於一個較少數量的通訊端，和 「**100**」值用於較多數量的通訊端。若用於極大數量多通訊端（多於數百），請使用「**epoll**」。

忙碌輪詢行為是由以下驅動程式支援。這些驅動程式亦支援 Red Hat Enterprise Linux 7。

- bnx2x
- be2net
- ixgbe
- mlx4
- myri10ge

截至 Red Hat Enterprise Linux 7.1，您也可以操作以下命令以檢查一個特定的裝置是否支援忙碌輪詢。

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

若傳回「**busy-poll: on [fixed]**」，忙碌輪詢就能於裝置上使用。

### 6.3.4. 設定通訊端接收佇列

若分析顯示通訊端因通訊端佇列的流失率過於緩慢而被解除，有一些方法能夠減少因而造成的效能問題。

**降低傳入流量的速度**

透過在封包延伸至佇列前或透過降低裝置重量，篩選或卸除封包以降低佇列填滿的速率。

**增加應用程式通訊端佇列的深度**

若一通訊端佇列在高載時接收有限數量的流量，增加通訊端深度以符合流量的高載大小，能夠避免封包被卸除。

#### 6.3.4.1. 降低傳入流量的速度

篩選傳入流量或降低網路介面卡的裝置重量，以減緩傳入流量。更多有關如何篩選傳入流量的詳細資訊，請參閱《*Red Hat Enterprise Linux 7 安全性指南*》，可自以下網址取得：〈http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/〉。

裝置重量指的是一個裝置一次能接收的封包數量（在一個單一排程處理器存取）。裝置重量是由「***dev_weight***」參數管理。透過修改「**/proc/sys/net/core/dev_weight**」檔案的內容，或與「**sysctl**」永久更動（由 *procps-ng* 套件提供），此參數可以暫時被更動。

#### 6.3.4.2. 增加佇列深度

增加一應用程式通訊端佇列的深度，通常是改善通訊端佇列流失率最簡單的方法，但這不太可能是長期的解決方法。

欲增加佇列深度，請透過以下變更增加通訊端接收緩衝區的大小：

**增加 /proc/sys/net/core/rmem_default 的值**

此參數控制通訊端使用的接收緩衝區的預設大小。此值必須小於**/proc/sys/net/core/rmem_max** 的值。

**使用 setsockopt 以設定一較大的 SO_RCVBUF 值**

此參數以位元組控制通訊端接收緩衝區的最大大小。使用「**getsockopt**」系統呼叫，以決定緩衝區當前的值。 更多有關此參數的詳細資訊，請參閱 man page：

```
$ man 7 socket
```

### 6.3.5. 配置接收端縮放比例（RSS）

接收端縮放比例（RSS），又稱多佇列接收，均分網路接收程序於數個硬體基礎的接收佇列，使多個 CPU 能夠處理傳入的網路流量。RSS 可以減少因單一 CPU 過載而造成的接收中斷處理的瓶頸，也可以降低網路延遲。

欲決定您網路介面卡是否支援 RSS，請檢查多中斷請求佇列是否與「**/proc/interrupts**」聯繫。例如：若您對「**p1p1**」介面有興趣：

```
# egrep 'CPU|p1p1' /proc/interrupts
   CPU0     CPU1     CPU2     CPU3     CPU4     CPU5
89:   40187       0       0       0       0       0   IR-PCI-MSI-edge
```

```
p1p1-0
90:       0     790       0       0       0       0   IR-PCI-MSI-edge
p1p1-1
91:       0       0     959       0       0       0   IR-PCI-MSI-edge
p1p1-2
92:       0       0       0    3310       0       0   IR-PCI-MSI-edge
p1p1-3
93:       0       0       0       0     622       0   IR-PCI-MSI-edge
p1p1-4
94:       0       0       0       0       0    2475   IR-PCI-MSI-edge
p1p1-5
```

此前置結果顯示 NIC 驅動程式「**p1p1**」介面建立了 6 個接收佇列（**p1p1-0** 至 **p1p1-5**）。 這也顯示有多少中斷是由各個佇列處理，以及哪一 CPU 處理了中斷。在此情況下，將會有 6 個佇列，因為在預設情況下，此特定 NIC 驅動程式建立一個 CPU 對一個佇列，且此系統有 6 個 CPU。這是一個在 NIC 驅動程式裏較為常見的模式。

另外，您可以檢查「**ls -1 /sys/devices/*/*/*device_pci_address*/msi_irqs**」的輸出，於網路驅動程式負載之後。例如：若您對一個有 **0000:01:00.0** PCI 位置的裝置有興趣，您可以列出其裝置的中斷請求佇列，用以下命令：

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

在預設情況下，RSS 是啓動的。RSS 的佇列數量（或應該處理網路活動的 CPU ）是設定在一個合適的網路裝置驅動程式。**bnx2x** 驅動程式是設定在「***num_queues***」。**sfc** 驅動程式是設定在「***rss_cpus***」參數。除此之外，RSS 通常是設定在「**/sys/class/net/*device*/queues/*rx-queue*/**」，其中「*device*」為網路裝置的名稱（例如：「**eth1**」）且「*rx-queue*」為合適的接收佇列名稱。

設定 RSS時，Red Hat 建議限制佇列數量至一個佇列對一個實體 CPU 核心。超線程在分析工具中，通常代表分隔的核心，但為所有的核心（包括如超線程的邏輯核心）設定佇列並沒有被證實是對網路效能有益。

RSS 啓動時，RSS 將以各個 CPU 排列的處理數量，在可使用的 CPU 當中均分網路處理，然而，您也可以使用 **ethtool *--show-rxfh-indir*** 與 ***--set-rxfh-indir*** 參數，以修改網路活動應如何分配，以及權衡較其他活動更為重要的特定網路活動。

**irqbalance** daemon 可以連同 RSS 一起用於降低跨節點記憶體轉換的可能性，以及快取線回升（cache line bouncing）。這將降低處理網路封包的延遲。

### 6.3.6. 配置接收封包操控

接收封包操控 （RPS）與 RSS 相似，兩者皆用於引導封包至特定 CPU 處理。然而，RPS 是置入在軟體層面，並協助避免單一網路介面卡的硬體佇列成為網路流量的瓶頸。

RPS 與以硬體為基礎的 RSS 相比，有以下數個好處：

- RPS 可以與任何網路介面卡一起使用。

- 非常容易增加軟體篩選至 RPS 以處理新的通訊協定。
- RPS 並不會增加網路裝置的硬體中斷率。然而，RPS 會引入處理器之間的中斷。

RPS 是按網路裝置和接收佇列設定，位於「**/sys/class/net/*device*/queues/*rx-queue*/rps_cpus**」檔案，其中「*device*」為網路裝置的名稱（例如：「**eth0**」），且「*rx-queue* 」為合適的接收佇列名稱（例如：「**rx-0**」）。

「**rps_cpus**」檔案的預設值為「**0**」。這將停用 RPS，如此，處理網路中斷的 CPU 亦能處理套件。

欲啓動 RPS，請使用應當處理來自特定網路裝置和接收佇列的 CPU，來設定合適的「**rps_cpus**」檔案。

「**rps_cpus**」檔案使用逗號分格 CPU 點陣圖（comma-delimited CPU bitmaps）。因此，欲使 CPU 能夠為介面上的接收佇列處理中斷，請設定點陣圖中 CPU 的位置值至 1。例如：欲使用值為 0、 1、2 和 3 的 CPU 處理中斷，請設定 「**rps_cpus**」值至「**00001111**」 （1+2+4+8），或 「**f**」（十六進位值為15）。

對於單一傳輸佇列的網路裝置，可以透過設定 RPS 使用同一記憶體網域的 CPU，達到最佳效能。在非 NUMA 系統上，這意味著所有現有的 CPU 皆可使用。若網路中斷率非常的高，排除處理網路中斷的 CPU 也可能改善效能。

對於多佇列的網路裝置，設定 RPS 和 RSS 兩者通常沒有益處，因為在預設情況下，RSS 的設定是為對應 CPU 至個接收佇列。然而，若硬體佇列較 CPU 佇列少，且 RPS 是設定去使用同一記憶體網域的 CPU，RPS 可能保持有益。

### 6.3.7. 配置接收封包操控 （RFS）

接收封包操控（RFS）擴展 RPS 行為以增加 CPU 快取命中率，並因此減少網路延遲。在 RPS 僅僅根據佇列長度而轉發套件，RFS 使用 RPS 後端計算最合適的 CPU，然後根據消耗套件的應用程式的位置，轉發套件。這將增加 CPU 緩存效率。

在預設情況下，RFS 是停用的。欲啓動 RFS，您必須編輯以下兩個檔案：

**/proc/sys/net/core/rps_sock_flow_entries**

設定此檔案值至使用中的並行連線的最大期望值。對於中等伺服器載量，我們建議其值為「**32768**」。 實際上，所有輸入的值皆會四捨五入至最近的次方。

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

以一個您欲設定的網路裝置名稱取代「*device*」（例如：「**eth0**」），並以一個您欲設定的接收佇列取代 「*rx-queue*」（例如：「**rx-0**」）。

設定此檔案的值至「**rps_sock_flow_entries**」，除以「**N**」，其中「**N** 」為裝置上接收佇列的數量。例如：若「**rps_flow_entries**」被設定為「**32768**」，且有 16 個接收佇列，「**rps_flow_cnt**」應該設定為「**2048**」。對於單一佇列裝置，「**rps_flow_cnt**」的值與「**rps_sock_flow_entries**」的值相同。

自一單一傳送者接收到的資料並不會傳送至一個以上的 CPU。若自一單一傳送者接收到的資料量大於單一 CPU 能夠處理的數量，請設定一個較大的框架大小，以減少中斷次數以及 CPU 處理的工作量。或者，考慮使用 NIC 承載選項或更快速的 CPU。

考慮使用「**numactl**」或「**taskset**」 連同 RFS，以釘選應用程式至特定核心、通訊端或 NUMA 節點。這能夠避免套件被不合乎規程的方式受處理。

### 6.3.8. 設定加速 RFS

加速 RFS （Accelerated RFS）透過增加硬體協助來加強 RFS 速度。如同 RFS，套件的轉寄是根據消耗該套件的應用程式的位置。然而，不同於傳統 RFS 的是。封包是直接傳送至 CPU，而此 CPU 是消耗資料的本機線程：執行應用程式的 CPU，或者一緩存層級的本機 CPU 。

增速 RFS 只在以下情況符合時方可行使：

- 增速 RFS 必須由網路介面卡支援。增速 RFS 是由輸出 **`ndo_rx_flow_steer()`** **`netdevice`** 功能的介面卡支援。
- **`ntuple`** 篩選必須是啓動的。

一旦這些條件符合，CPU 至佇列對應將會根據傳統 RFS 設定自動將低。亦即 CPU 至佇列對應的降低，是根據驅動程式為各個接收佇列設定的 IRQ 親和性。有關設定傳統 RFS 的詳細資訊，請參閱〈節 6.3.7, “配置接收封包操控 （RFS）”〉。

Red Hat 建議在任何適當的情況下，以及在網路介面卡支援硬體增速時，使用增速 RFS。

# 附錄 A. 工具參考

本附錄為 Red Hat Enterprise Linux 7 中各種能夠用於調整效能的工具提供一個快速參考。請參閱相關 man page 提供的完整、最新和詳細的參考資料。

## A.1. irqbalance

「**irqbalance**」是處理器分發硬體中斷的命令列工具，能夠改善系統效能。此工具在預設情況下以 daemon 運行，但僅能與「**--oneshot**」選項運行一次。

以下參數有益於改善效能。

**--powerthresh**

此工具在 CPU 處於節能模式前，設定可閒置 CPU 的數量。若 CPU 數量比門檻值高，在平均值 softirq 工作負載之下高過一個標準差；沒有 CPU 數量在平均值以上高過一個標準差，且有一個以上的指派 irq，則 CPU 會處於節能模式。在節能模式中，CPU 並不是 irq balancing 的一部分，所以不會在非必要情況下被啓動。

**--hintpolicy**

此工具決定如何處理 irq kernel 親和性提示（irq kernel affinity hinting）。有效值為**exact**（irq 親和性提示一直適用），**subset**（irq 是平衡的，但指派物件是親和性提示的子集），或 **ignore**（完全忽視 irq 親和性提示）。

**--policyscript**

此工具決定指令碼執行每個中斷的位置， 以裝置路徑和通過的 irq 號碼作為引數，與一個**irqbalance** 預期的零結束代碼。受定義指令碼可以指定零個或多個關鍵值，以配對管理通過 irq 中的指引 **irqbalance**。

以下為受認可的有效關鍵值配對。

**ban**

有效值為 **true**（不包含自平衡的通過 irq）或 **false**（於此 irq 執行平衡）。

**balance_level**

此工具使使用者優先於通過 irq 的平衡層級。在預設情況下，平衡層級是以 PCI 裝置類別中，擁有 irq 的裝置作為基礎。有效值為 **none**、**package**、**cache** 或 **core**。

**numa_node**

此工具使使用者優先於 NUMA 節點，而此節點被視為本機通過 irq。若有關本機節點的資訊未在 ACPI 中指明，配置將被視為與各節點等距。有效值為分辨一個特定 NUMA 節點的整數（從零開始），和指明一個 irq 應被視為與各節點等劇的 **-1**。

**--banirq**

伴隨特定中斷請求數量的中斷，新增至禁止中斷列表。

您可以也使用 ***IRQBALANCE_BANNED_CPUS*** 環境變數，以指明被 **irqbalance** 略過的遮罩 CPU。

更多詳細資訊，請參閱 man page：

```
$ man irqbalance
```

## A.2. Tuna

**Tuna** 使您能夠控制處理器和排程親和性（scheduling affinity）。此章節涵蓋命令列，但提供有著相同範圍功能的圖形化介面。欲啓動圖形化公用程式，請在命令列中執行 **tuna**。

**Tuna** 接受若干依順序處理的命令列參數。以下命令橫跨一個四通訊端系統分配負載。

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

**--gui**

--gui 啓動圖形化使用者介面。

**--cpus**

--cpus 使用逗號分隔列表，列出由 **Tuna** 控制的 CPU。至有指定的新列表為止，此列表維持有效。

**--config_file_apply**

--config_file_apply 使用一個設定檔名，以於系統中應用。

**--config_file_list**

--config_file_list 列出負載前設定檔。

**--cgroup**

--cgroup 與 ***--show_threads*** 一起使用。若控制群組已啓用，顯示處理屬於 ***--show_threads*** 控制群組的類別。

**--affect_children**

指定 --affect_children 時，**Tuna** 會影響子執行續與母執行緒。

**--filter**

--filter 篩選顯示僅受影響的實體。

**--isolate**

執行 --isolate 時，請使用逗號分隔列表，列出 CPU。**Tuna** 會把所有指定 CPU 上的所有執行緒轉移至其他地方。

**--include**

--include 使用逗號分隔列表，列出 CPU。Tuna 使所有執行緒能夠運行於特定 CPU。

**--no_kthreads**

有特定參數時，Tuna 並不影響 kernel 執行緒。

**--move**

--move 移動選定實體至特定 CPU。

**--priority**

指定排程器原則和執行序的優先順序。有效排程器原則為 **OTHER**、**FIFO**、**RR**、**BATCH**，或 **IDLE**。

當優先順序為 **FIFO** 或 **RR**，有效優先順序值則為最低指數 1 至最高指數 99 。預設值為 **1**。舉例來說，**tuna --threads 7861 --priority=RR:40** 設定 **RR** 優先順序（循環配置資源）與執行緒 **7861** 的 **40** 優先順序）。

當原則為 **OTHER**、**BATCH** 或 **IDLE**，唯一有效優先值則為亦是預設值的 **0**。

**--show_threads**

--show_threads 顯示執行緒列表。

**--show_irqs**

--show_irqs 顯示 IRQ 列表。

**--irqs**

--irqs 使用逗號分隔列表，列出影響 IRQ 的 **Tuna** 。此列表將維持有效，直至有新的指定列表。 使用**+**，IRQ 能夠新增至列表，而使用 **-** 能夠將 IRQ 從列表中移除。

**--save**

--save 保存 kernel 執行緒排程於指定檔案。

**--sockets**

--sockets 使用逗號分隔列表，列出由 **Tuna** 控制的 CPU 插槽。此選項將系統拓樸納入考量，比方：共享單一 cache 處理器的核心，與在同一實體晶片的核心。

**--threads**

--threads 使用逗號分隔列表，列出由 **Tuna** 控制的執行緒。此列表維持有效，直至有新的指定列表為止。透過使用 **+**，可新增執行緒至列表，而使用 **-** 則能將其從列表移除。

**--no_uthreads**

--no_uthreads 避免運作被使用者執行緒影響。

**--what_is**

--what_is 查看選定實體上的進一步協助。

**--spread**

--spread 在 **--cpus** 指定 CPU 之間，平均分配 **--threads** 指定執行緒。

## A.3. ethtool

**ethtool** 公用程式使系統管理員能夠查閱和編輯網路界面卡設定。此工具有助於觀察特定裝置的數據，比方：由裝置刪除的封包數量。

**ethtool** 的選項與其使用量皆完整地記錄在 man page 中。

```
$ man ethtool
```

## A.4. ss

**ss** 為命令列公用程式，能夠列出有關通訊端的數據資訊，使系統管理員能夠隨時間取得裝置效能。在預設情況下，**ss** 會列出開放的非聽取性 TCP 通訊端以建立連線，但 ss 亦提供多種有用選項，協助系統管理者篩選有關特定通訊端的數據。

一個常見的命令為 **ss -tmpie**，其顯示所有 TCP 通訊端（**t**）、內部 TCP 資訊（**i**）、通訊端記憶體使用量（**m**）、使用通訊端的程序（**p**），和詳細的通訊端資訊（**i**）。

Red Hat 建議在 Red Hat Enterprise Linux 7 中選擇 **ss** 更勝 **netstat**。

**ss** 由 *iproute* 套件提供。更多資訊，請參閱 man page：

```
$ man ss
```

## A.5. tuned

**Tuned** 為微調 daemon，透過設定微調檔案，能夠調整作業系統在特定作業負載有更好的效能。此工具亦能設定以應對 CPU 與網路使用量的更動，並調整設定，以改善活躍裝置中的效能和降低其中的能源消耗量。

欲設定動態微調顯示方式，請編輯 ***dynamic_tuning*** 參數，位於**/etc/tuned/tuned-main.conf** 檔案中。您亦可在微調確認使用量和更新微調細節 ***update_interval*** 參數時，將時間用量設為秒。

更多有關微調的詳細資訊，請參閱 man page：

```
$ man tuned
```

## A.6. tuned-adm

**tuned-adm** 為一個命令列工具，提供若干不同設定檔以改善特定使用例子中的效能。此工具亦能提供子命令（**tuned-adm recommend**），進入您的系統和輸出一個建議的微調設定檔。這亦於安裝時，為您系統設定預設設定檔，以用於回復預設設定檔。

Red Hat Enterprise Linux 7 中，**tuned-adm** 包含運作任何命令，作為啓動與停用微調設定檔的一部分。這使您能夠新增在 **tuned-adm** 中沒有的環境特定檢閱，例如：檢閱系統是否在選取應用哪一微調設定檔前，為 master 資料庫節點。

Red Hat Enterprise Linux 7 亦於設定檔定義檔案中提供 ***include*** 參數，使您能夠將自己的 **tuned-adm** 設定檔以現存設定檔為基礎。

以下微調設定檔具備 **tuned-adm** 並在 Red Hat Enterprise Linux 7 受支援。

**輸出量效能**

此工具為一針對改善輸送量的伺服器設定檔。這是一個預設設定檔，並建議用在大多數系統上。

透過設定 **intel_pstate** 與 **max_perf_pct=100**，此設定檔有助於效能，更勝節能。此設定檔能夠啓動通透的大頁面、使用 **cpupower** 以設定 **performance** cpufreq 管理員，與設定輸入 / 輸出排程器至 **deadline**。此設定檔也設定 ***kernel.sched_min_granularity_ns*** 至 **10** µs、***kernel.sched_wakeup_granularity_ns*** 至 **15** µs，和 ***vm.dirty_ratio*** 至 **40**%。

**延遲效能**

此工具為一針對降低延遲的伺服器設定檔。此設定檔建議用在有助於 c-state 微調的延遲敏感作業負載，與增加透明巨大頁面的 TLB 效率。

透過設定 **intel_pstate** 與 **max_perf_pct=100**，此設定檔有助於效能，更勝節能。此工具能夠啓用透明巨大頁面、使用 **cpupower** 以設定 **performance** cpufreq 管理員，與要求 ***cpu_dma_latency*** 的值為 **1**。

**網路效能**

此工具為一針對降低網路延遲的伺服器設定檔。

透過設定 **intel_pstate** 與 **max_perf_pct=100**，此設定檔有助於效能，更勝於節能。此工具能夠啓用透明巨大頁面與自動 NUMA 平衡。此工具亦能使用**cpupower**以設定 **performance** cpufreq 管理員，與要求 ***cpu_dma_latency*** 的值為 **1**。此工具亦設定 ***busy_read*** 與 ***busy_poll*** 時間至 **50** µs，***tcp_fastopen*** 至 **3**。

**網路輸出量**

此工具為一針對改善網路輸出量的伺服器設定檔。

透過設定 **intel_pstate** 與 **max_perf_pct=100** 和增加 kernel 網路緩衝區大小，此設定檔有助於效能，更勝節能。此工具能夠啓動透明巨大頁面，與使用 **cpupower** 以設定 **performance** cpufreq 管理員。此工具亦設定 ***kernel.sched_min_granularity_ns*** 至 **10** µs、***kernel.sched_wakeup_granularity_ns*** 至 15 µs，和 ***vm.dirty_ratio*** 至 **40**%。

**虛擬客座端**

此工具為一針對 Red Hat Enterprise Linux 7 中的虛擬機器優化效能。

透過設定 **intel_pstate** 與 **max_perf_pct=100**，此設定檔有助於效能，更勝節能。此工具亦能夠啓動透明巨大頁面，與使用 **cpupower** 以設定 **performance** cpufreq 管理員。此工具亦設定 ***kernel.sched_min_granularity_ns*** 至 **10** µs、***kernel.sched_wakeup_granularity_ns*** 至 15 µs，與 ***vm.dirty_ratio*** 至 **40**%。

**虛擬主機**

此工具為一針對 Red Hat Enterprise Linux 7 中虛擬化主機的優化效能。

透過設定 **intel_pstate** 與 **max_perf_pct=100**，此設定檔有助於效能，更勝節能。此工具亦能降低虛擬記憶體的 swappiness。此設定檔能夠啓動透明巨大頁面，與更為頻密地將已變更的頁面寫回至硬碟。此工具使用 **cpupower** 以設定**performance** cpufreq 管理員。此工具亦設定***kernel.sched_min_granularity_ns*** 至 **10** µs、***kernel.sched_wakeup_granularity_ns*** 至 15 µs、***kernel.sched_migration_cost*** 至 **5** µs，和 ***vm.dirty_ratio*** 至 **40**%。

更多有關具有 tuned-adm 的節能設定檔的詳細資訊，請參閱《*Red Hat Enterprise Linux 7 能源管理指引*》。網址為： http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。

更多有關使用「**tuned-adm**」的詳細資訊，請參閱 man page：

```
$ man tuned-adm
```

## A.7. perf

**perf** 工具提供若干實用的命令，其中有一些被列在本章節。更多有關「**perf**」的詳細資訊，請參閱《*Red Hat Enterprise Linux 7 開發者指引Developer Guide*》。網址為：http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/，或參照 man page。

**perf stat**

此命令為常見效能事件提供總體數據，包括已執行指令與消耗的時間週期。您能使用選項旗幟以聚集特定事件的數據，而非預設度量事件。Red Hat Enterprise Linux 6.4 中，能夠使用 **perf stat** 以基於一個或更多特定控制群組（cgroup），篩選監視。

更多資訊，請參閱 man page：

```
$ man perf-stat
```

**perf record**

此命令將效能資料紀錄在一個能夠使用 **perf report**，進行稍後分析的檔案。更多詳細資訊，請參閱 man page：

```
$ man perf-record
```

**perf report**

此命令從一個檔案閱讀效能資料，並分析記錄資料。更多詳細資訊，請參閱 man page：

```
$ man perf-report
```

**perf list**

此命令列出在特定機器上的可用事件。這些事件根據效能監視硬體和系統的軟體設定而有所不同。更多資訊，請參閱 man page：

```
$ man perf-list
```

**perf top**

此命令與 **top**工具執行類似功能。此命令產生並顯示實時效能相應設定檔。更多資訊，請參閱 man page：

```
$ man perf-top
```

**perf trace**

此命令與 **strace**工具執行類似功能。此命令監視由特定執行緒或程序使用的系統呼叫，和所有由其應用程式接收的訊號。更多的追蹤目標，請參閱 man page 中的完整列表：

```
$ man perf-trace
```

## A.8. Performance Co-Pilot（PCP）

Performance Co-Pilot（PCP）提供大量的命令列工具、圖形化工具，與媒體庫。更多有關其中任何工具的資訊，請參閱 man page：請於命令列輸入「**man *toolname***」，以取代「*toolname*」工具名稱。

預設情況下，「*pcp-doc*」套件安裝詳細文件至「**/usr/share/doc/pcp-doc**」目錄。

Red Hat 客戶入口網站有許多實用文章，能夠幫助您瞭解如何使用 PCP。這些文章的索引網址為：https://access.redhat.com/articles/1145953。

## A.9. vmstat

「**Vmstat**」輸出您系統程序、記憶體、分頁、區塊輸入／輸出、中斷，和 CPU 活動的報告。 此工具提供這些事件自最近一次機器重新啓動，或自之前報告的即時報告。

**-a**

顯示活躍與非活躍記憶體。

**-f**

顯示自重新啓動後 fork 的數量。這包括「**fork**」、「**vfork**」， 和「**clone** 」系統呼叫，並等同於總建立任務的數量。各個程序是根據執行緒使用量，由一個或更多任務代表。此顯示不重複。

**-m**

顯示 slab 資訊。

**-n**

指明標題只會顯示一次，而非定期顯示。

**-s**

顯示有關各種事件相應與記憶體數據的表格。此顯示不重複。

**delay**

報告之間的延遲，以秒為單位。若沒有特定延遲則只會印出一份報告，並顯示機器自最近一次重新啓動後的平均值。

**count**

清算對系統提出報告的次數。若無特定計數且延遲已定義，「**vmstat**」將不受限制提出報告。

**-d**

顯示硬碟數據。

**-p**

以分割名稱作為數值，並為其分割區提出詳細報告。

**-S**

定義報告中的單位輸出。有效數值為「**k**」（1,000 位元組）、「**K**」 （1,024 位元組）、「**m**」（1,000,000 位元組），或「**M**」（1,048,576 位元組）。

更多有關由各個輸出模式提供的輸出，請參閱 man page：

```
$ man vmstat
```

## A.10. x86_energy_perf_policy

「**x86_energy_perf_policy**」工具使系統管理員能夠定義相關效能重要性與能源效率。此工具由「*kernel-tools*」套件提供。

欲參閱目前原則，請執行以下命令：

```
# x86_energy_perf_policy -r
```

欲設定新原則，請執行以下命令：

```
# x86_energy_perf_policy profile_name
```

用以下其中之一的設定檔取代設定檔名。

**performance**

程序並不因節能而犧牲效能。這是預設值。

**normal**

處理器容許犧牲一點效能以達到可能的大幅度節能。這對於大多數伺服器和桌上型電腦是合理的節能方式。

**powersave**

處理器接受為了使能源效率提升至最大限度，而可能造成的效能大幅度降低。

更多有關如何使用「**x86_energy_perf_policy**」的詳細資訊，請參閱 man page：

```
$ man x86_energy_perf_policy
```

## A.11. turbostat

「**turbostat**」工具提供詳細資訊，有關在不同狀態系統所消耗的時間量。「**turbostat**」是由「*kernel-tools*」套件提供。

在預設情況下，使用以下標題則「**turbostat**」將為整個系統印出相應結果摘要，其後為每間隔 5 秒的相應結果 。

**pkg**

處理器套件號碼。

**core**

處理器核心號碼。

**CPU**

Linux CPU （邏輯處理器）號碼。

**%c0**

CPU 淘汰指令的間隔百分比。

**GHz**

當 CPU 為 c0 狀態的平均時間速度。

**TSC**

整個間隔程序的平均時間速度。

**%c1、 %c3、 與 %c6**

分別為處理器為 c1、c3、c6 狀態的間隔百分比。

**%pc3 或 %pc6**

分別為處理器為 c1、c3、c6 狀態的間隔百分比。

在有「**-i**」選項的相應結果之間，指定一個不同期間。例如：運作「**turbostat -i 10**」以每 10秒顯示結果。

**注意**

預定 Intel 處理器可能新增額外的 c-state。 Red Hat Enterprise Linux 7.0 中，「**turbostat** 」為 c7、c8、c9 和 c10 狀態提供支援。

## A.12. numastat

「**numastat**」工具由「*numactl*」套件提供，並為程序和以 NUMA 節點為基礎的運作系統顯示記憶體數據（例如：hit 與 miss 分佈）。「**numastat**」命令的預設追蹤分類為下：

**numa_hit**

成功配置至此節點的分頁。

**numa_miss**

因在預定節點上的低記憶體，而成功配置至此節點的分頁。各「**numa_miss**」事件在另一個節點有相應「**numa_foreign**」事件。

**numa_foreign**

最初預定至此節點但被配置至其他節點的分頁。各「**numa_foreign** 」事件在其他節點有相應「**numa_miss**」事件。

**interleave_hit**

成功配置至此節點的間隔原則分頁。

**local_node**

透過此節點上的程序，成功配置的分頁。

**other_node**

透過此節點上的程序，成功配置的分頁。

提供以下任何選項，能夠更改顯示單位至 MB 記憶體 （取整數至小數第二位），也能更改以下所述的其他特定「**numastat**」行為。

**-c**

水平壓縮顯示的資訊表格。這有益於有大量 NUMA 節點的系統，但欄寬和欄與欄之間的間距難以預測。當使用此選項，記憶體量將取整數至最近 MB。

**-m**

以各節點為基礎，顯示系統上的記憶體使用量資訊，這和 **/proc/meminfo** 中找的資訊類似。

**-n**

顯示與原本 numastat 命令相同的資訊

顯示與原本 numastat 命令相同的資訊（**numa_hit**、**numa_miss**、**numa_foreign**、**interleave_hit**、**local_node**，和 **other_node**）。更新後的格式使用 MB 作為測量單位。

**-p pattern**

為特定模式顯示各節點記憶體資訊。若模式的價值包含數字，「**numastat**」將採用數字程序辨識器。 否則「**numastat**」將為特定模式搜索程序命令列。

輸入在「**-p**」選項後的命令列引數，將被視為需要額外篩選的模式。額外的模式將會擴大篩選，而非縮小篩選。

**-s**

將顯示資料依降冪分類，以使最大的記憶體消耗者首先列出（根據總欄數）。

您亦能指定節點，然後表格將會依節點欄分類。使用此選項時，節點值必須馬上依循「**-s**」 選項，如下：

```
numastat -s2
```

不包含選項和其值之間的白色空格鍵。

**-v**

顯示更多詳細資訊，即多程序的資訊將會為各程序顯示詳細資訊。

**-V**

顯示 numastat 版本的資訊。

**-z**

省略顯示資訊中值為零的資訊表格欄列。請注意，有些因顯示因素而取整數為零的近零值，將不會自顯示輸出中省略。

## A.13. numactl

「**Numactl**」使系統管理員能夠運作一個有特定排程或記憶體位置原則的程序。 「**Numactl**」也可以為共享記憶體區段或檔案設定一個持續原則，並設定程序的親和性和程序的記憶體親和性。

「**Numactl**」提供一些實用選項。此附錄簡述其中一些選項並給予使用建議，但並非詳盡。

**--hardware**

顯示系統上可用節點的列表，包含節點間相對距離。

**--membind**

確保記憶體自特定節點分配。若特定位置沒有足夠的記憶體，分配將會失效。

**--cpunodebind**

確保特定命令和其子程序在特定節點執行。

**--phycpubind**

確保特定命令和其子程序在特定處理器執行。

**--localalloc**

指明記憶體應一直分配自本地節點。

**--preferred**

指明一個分配記憶體的慣用節點。若記憶體不能夠從此特定節點分配，則另一節點將會作為後援使用。

更多有關這些以及其他參數的詳細資料，請參閱：

```
$ man numactl
```

## A.14. numad

「**numad**」為一自動 NUMA 親和性管理 daemon。此工具監控系統內 NUMA 拓樸和資源使用量，以動態改善 NUMA 資源分配和管理。

請注意，「**numad**」啓用時，其行為超越預設自動 NUMA 平衡行為。

### A.14.1. 使用自命令列的 numad。

欲使用「**numad**」為可執行檔，請運用：

```
# numad
```

「**numad**」運作時，其活動會記錄在「**/var/log/numad.log**」。 此工具會運作至被以下命令停止為止：

```
# numad -i 0
```

停止「**numad**」 並不會將其在 NUMA affinity 中的改善移除。若系統使用量有巨大改變，再次運作「**numad**」將會更動親和性（affinity），以改善新操作模式下的效能。

欲限制「**numad**」管理至特定程序，請與以下選項一起啓動。

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

**-p *pid***

此選項新增特定「*pid*」至一明確涵蓋列表。指明的程序將不會被管理，直至其符合「**numad**」程序精確度執行緒為止。

**-S 0**

此工具設定掃描至「**0**」的程序類型，其限制「**numad**」管理至一明確涵蓋列表。

更多有關可使用的「**numad**」選項的詳細資訊，請參閱 「**numad**」 man page：

```
$ man numad
```

### A.14.2. 將 numad 作為服務使用

「**numad**」作為服務運作時，試圖根據目前的系統工作負載量，動態微調系統。其活動記錄在「**/var/log/numad.log**」。

欲啓動此服務，請運作：

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

欲使服務在重新啓動後仍保存，請運作：

```
# chkconfig numad on
```

更多有關可使用的「**numad**」選項的詳細資訊，請參閱 「**numad**」 man page：

```
$ man numad
```

### A.14.3. 放置前建議

「**numad**」提供放置前建議服務，能夠透過各種工作管理系統取得，以協助其程序的 CPU 的最初繫結與記憶體資源。無論「**numad**」是否作為可執行檔案或一項服務，此放置前建議皆可使用。

### A.14.4. 以 KSM 使用 numad

若 KSM 在一 NUMA 系統上使用，請更改「**/sys/kernel/mm/ksm/merge_nodes**」參數值至「**0**」，以避免 NUMA 節點中的頁面合併。其他情況下，因為節點中的頁面合併，所以 KSM 將增加遠端記憶體存取。此外，kernel 統計數據最終可能在大量跨節點合併而相互衝突。 若為以上情況， 在 KSM daemon 合併許多記憶體頁面後，「**numad**」可能對於正確的可使用記憶體的數量與位置有所混淆。KSM 僅在過度使用系統上記憶體的情況下有益。若您的系統有足夠的自由記憶體，您可以透過關閉和停止 KSM daemon 達到更好的效能。

## A.15. OProfile

OProfile 為一「*oprofile*」套件提供的低額外負載、系統性效能監控工具。此工具使用處理序上的效能監控硬體，以取出系統上有關 kernel 與可執行檔案的資訊，例如：當記憶體受參閱，二階快存（second-level cache）的請求數量與接收到的硬體中斷請求。OProfile 亦能簡介運作於 Java Virtual Machine （JVM）中的應用程式。

OProfile 提供以下工具。請注意，舊版「**opcontrol**」工具與新的 「**operf**」工具是互斥的。

**ophelp**

為系統處理器顯示可使用事件，且提供各簡要描述。

**opimport**

為系統將樣本資料庫檔案自外部二進制模式轉變為本地模式。僅能在分析一不同架構的樣本資料庫時使用。

**opannotate**

為可執行檔案生產標註資源，若應用程式為 debugging 標誌編譯。

**opcontrol**

編譯哪一資料為分析運作中收集。

**operf**

試圖取代「**opcontrol**」。「**operf**」工具使用 Linux 效能事件子系統，使您能夠更精確地專注於分析。此工具為單一程序或橫跨系統的工具，並能夠更好的與系統上其他使用效能監控硬體的工具共存。不同於「**opcontrol**」，最初設定並不需要，且此工具能夠在沒有 root 權限的情況下使用，除非「**--system-wide** 」選項正在使用。

**opreport**

取出設定檔資料。

**oprofiled**

作為 daemon 運作，以定期顯示樣本資料至硬碟。

舊版模式（**opcontrol**、**oprofiled**，與程序後工具）仍可以取得，但不再為分析方式所建議。

更多有關其中任何的命令的詳細資訊，請參閱 OProfile man page：

```
$ man oprofile
```

## A.16. taskset

「**taskset**」工具由 「*util-linux*」套件提供。 此工具使系統管理員能夠取出並設定運作中的處理器親合性（processor affinity），或使用特定處理器親和性（processor affinity）啓動程序。

**重要**

「**taskset**」並不能保證本地記憶體分配。 若您請求本地記憶體分配額外效能益處，Red Hat 建議您使用「**numactl** 」，取代使用 taskset。

欲設定一運作中程序的 CPU 親合性 （CPU affinity）， 請運作以下命令：

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

以處理序的逗點分隔列表或處理序範圍，取代「*processors*」（例如：「**1,3,5-7**」）。以您所希望重新設定的程序辨識器，取代「*pid*」。

欲使用特定親和性（specified affinity）啓動程序，請運作以下命令：

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

以處理器的逗點分隔列表或處理器的範圍，取代「*processors*」。 以命令、選項和您所希望運作的應用程式引數，取代「*application* 」。

更多有關「**taskset**」的資訊，請參閱 man page：

```
$ man taskset
```

## A.17. SystemTap

「**SystemTap**」的詳細資訊，記錄在 system tap 指引中。。Red Hat Enterprise Linux 7 版本的《*SystemTap 初學者指引*》與《*SystemTap TapSet 參考*》兩者的網址為：
「http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/」。

# 附錄 B. 修訂記錄

**修訂 0.3-24.1** **Fri Jun 10 2016** **Chester Cheng**

說明：7.1 版翻譯、校對完成。
**翻譯、校對：**陳郁棠。
**翻譯、校對：**羅詩婷。
**校對、責任編輯：**鄭中。
**附註：**本繁體中文版來自「**Red Hat 全球服務部**」與「**澳大利亞昆士蘭大學・筆譯暨口譯研究所**」之產學合作計畫。若有疏漏之處，盼各方先進透過以下網址，給予支持指正：https://bugzilla.redhat.com/。

**修訂 0.3-24** **Mon Feb 23 2015** **Laura Bailey**

修正確認 busy poll 支援的細節，BZ1080703。

**修訂 0.3-23** **Tue Feb 17 2015** **Laura Bailey**

建立 RHEL 7.1 GA。

**修訂 0.3-22** **Tue Feb 17 2015** **Laura Bailey**

新增 busy poll 支援檢查，BZ1080703。

**修訂 0.3-21** **Thu Feb 05 2015** **Laura Bailey**

標記新的 tuned 設定檔參數 cmdline，BZ1130818。
新增註解至「7.1 的新增」 章節；更新在 swapon 命令的捨棄參數，BZ1130826.

**修訂 0.3-20** **Fri Jan 09 2015** **Charles Boyle**

更正 pgrep 命令中的錯誤。BZ1155253。
改善 vm.swappiness 的敘述。BZ1148419。

**修訂 0.3-19** **Fri Dec 05 2014** **Laura Bailey**

為形象頁面的呈現更新 sort_order。

**修訂 0.3-18** **Wed Nov 26 2014** **Laura Bailey**

為 7.1 Beta 新增連結至 PCP 文章索引，BZ1083387。

**修訂 0.3-15** **Mon Nov 24 2014** **Laura Bailey**

新增註解於 idle 平衡變更，BZ1131851。

**修訂 0.3-14** **Wed Nov 19 2014** **Laura Bailey**

更新 huge page 分配的細節，BZ1131367。
新增如何在執行階段分配 huge pages 的例子，BZ1131367。

**修訂 0.3-12** **Thu Nov 13 2014** **Laura Bailey**

為 RHEL 7.1 中的 SHMALL 與 SHMMAX 增加新的預設值， BZ1131848。

**修訂 0.3-11** **Mon Nov 10 2014** **Laura Bailey**

為 ext4 新增有關叢集分配／bigalloc 的註解，BZ794607。

**修訂 0.3-10** **Fri Oct 31 2014** **Laura Bailey**

紀錄各節點靜態，BZ1131832。

**修訂 0.3-9** **Tue Jul 22 2014** **Laura Bailey**

新增 latencytap.stp script 的敘述，BZ988155。

**修訂 0.3-7** **Thu Jun 26 2014** **Laura Bailey**

更正 CPU 章節裡的打字錯誤 。感謝 Jiri Hladky。
移除關於 tuned 更改 I/O 排程器的參考。感謝 Jiri Hladky。

**修訂 0.3-5** **Wed Jun 11 2014** **Laura Bailey**

新增尾斜杠進入 redhat.com 連結，不會重新導向。

**修訂 0.3-4** **Tue Jun 10 2014** **Laura Bailey**

新增中斷和 CPU 的禁止細節至 irqbalance 附錄 BZ852981。

**修訂 0.3-3** **Mon Apr 07 2014** **Laura Bailey**

RHEL 7.0 GA 的重建

**修訂 0.3-2** **Mon Apr 07 2014** **Laura Bailey**

為 RT#294949 更新圖書結構。

**修訂 0.2-38** **Mon Apr 07 2014** **Laura Bailey**

新增更新的 OProfile 資料， BZ955882。
移除過時註解。

**修訂 0.2-34** **Fri Apr 04 2014** **Laura Bailey**

更正 lstopo 輸出影像樣式，BZ1042800。
新增有關 irqbalance daemon 的細節， BZ955890。
新增並更正有關控制群組的細節，BZ794624。
新增有關 PCP 的細節， BZ955883。
更新 XFS 微調的細節，BZ794616。
新增更新的 OProfile 資料， BZ955882。

**修訂 0.2-27** **Fri Mar 28 2014** **Laura Bailey**

根據 Jeremy Eder 的反饋，更正 busy_poll 章節， RT276607。
根據 Jeremy Eder 的反饋，更正 nohz_full 章節並新增細節， RT284423。
新增進一步的細節至 SystemTap 章節， BZ955884。
新增進一步的細節至 SSD 章節，BZ955900。
新增有關 tuned-adm 建議命令更進一步的細節，BZ794623。
更正功能章節裡有關自動 NUMA 平衡的註解，BZ794612。
更正數個有關 NUMA 的術語問題和樣板輸出問題，包含新的影像， BZ1042800。
根據 Jeremy Eder 的反饋，更正有關 irqbalance 結合 RRS 的細節。

**修訂 0.2-19** **Fri Mar 21 2014** **Laura Bailey**

新增關於透明 huge pages 的細節至記憶體章節，BZ794621。
更正有關 NUMA 節點的使用術語，BZ1042800。
更新 kernel 上限，BZ955894。
擬定 tickless kernel 章節，RT284423。
擬定 busy polling 章節，RT276607。
更新有關檔案系統壁壘的訊息。
移除有關個節點 huge page 任務不明確的訊息。創建 BZ1079079 ， 為將來新增更多實用的訊息。
新增有關固態硬碟的細節， BZ955900。
移除檢閱標記。

**修訂 0.2-14** **Thu Mar 13 2014** **Laura Bailey**

套用 Jeremy Eder 和 Joe Mario 的反饋。
從 BZ955872 紀錄 Tuna GUI 的更新。
新增有關 SystemTap 的細節至網路章節和工具參考附錄， BZ955884。

**修訂 0.2-12** **Fri Mar 07 2014** **Laura Bailey**

紀錄自動 NUMA 移轉的支援，每一 BZ794612。
額外應用 Jeremy Eder 的反饋。

**修訂 0.2-11** **Fri Mar 07 2014** **Laura Bailey**

應用 Jeremy Eder 的反饋。

**修訂 0.2-10** **Mon Feb 24 2014** **Laura Bailey**

根據「Lukáš Czerner (BZ#794607)」的反饋，更正 Ext4 的訊息。

**修訂 0.2-9** **Mon Feb 17 2014** **Laura Bailey**

根據 Bill Gray 的反饋更正 CPU 章節。
根據 Bill Gray 的反饋，更正並新增至記憶體章節與工具參考。

**修訂 0.2-8** **Mon Feb 10 2014** **Laura Bailey**

新增 isolcpus boot 參數至 CPU 章節（RT276607）。
SME 反饋：更正參數敘述和新增新的參數 （BZ#970844）。
新增建議 tuned-adm 設定檔至網路章節。
為了檢閱，新增標記至旗標章節。

**修訂 0.2-4** **Mon Feb 03 2014** **Laura Bailey**

確認「**`numactl --membind`**」參數有被紀錄（BZ#922070）。
新增有關 Tuna 的細節至工具介紹、CPU 章節和工具參考附錄（BZ#970844）。
更正儲存和檔案系統章節中的建構性錯誤。
新增缺少的相互參照。

**修訂 0.2-2** **Fri Jan 31 2014** **Laura Bailey**

完成重寫和重新架構。
確認所有在指南中提及的工具，都有列在提供這些工具的套件。

**修訂 0.1-11** **Thu Dec 05 2013** **Laura Bailey**

建造重新架構的 RHEL 7.0 Beta 指南。

**修訂 0.1-10** **Wed Nov 27 2013** **Laura Bailey**

Pre-Beta 客戶建立。

**修訂 0.1-9** **Tue Oct 15 2013** **Laura Bailey**

根據顧客（BZ#1011676）的反饋，做出微小的更正。

**修訂 0.1-7** **Mon Sep 09 2013** **Laura Bailey**

合併自 RHEL 6.5 的新內容。
應用編輯者的回饋。

**修訂 0.1-6** **Wed May 29 2013** **Laura Bailey**

更新 ext4 檔案系統上限（BZ#794607）。
更正理論上最大的 64 位元檔案系統。
新增〈新功能〉章節至追蹤效能相關的變更。
從 「**cfq**」至 「**deadline** 」變更預設 I/O 排程器（BZ#794602）。
為 BTRFS 微調新增草擬內容（BZ#794604）。
更新 XFS 章節以提供有關目錄塊大小更明確的建議，也更新 XFS 支援上限（BZ#794616）。

**修訂 0.1-2** **Thurs Jan 31 2013** **Tahlia Richardson**

更新並出版 RHEL 7 草案。

**修訂 0.1-1** **Wed Jan 16 2013** **Laura Bailey**

本文件源出於 RHEL 6.4 版。