Android 對 Linux 核心修改的分析

梁雲豪 林盈達 張尚揚
國立交通大學資訊工程系
300 新竹市大學路 1001 號
E-MAIL：chobitscu.cs01g@nctu.edu.tw, ydlin@cs.nctu.edu.tw,
csyang.cs99g@nctu.edu.tw
September 28, 2012

摘要

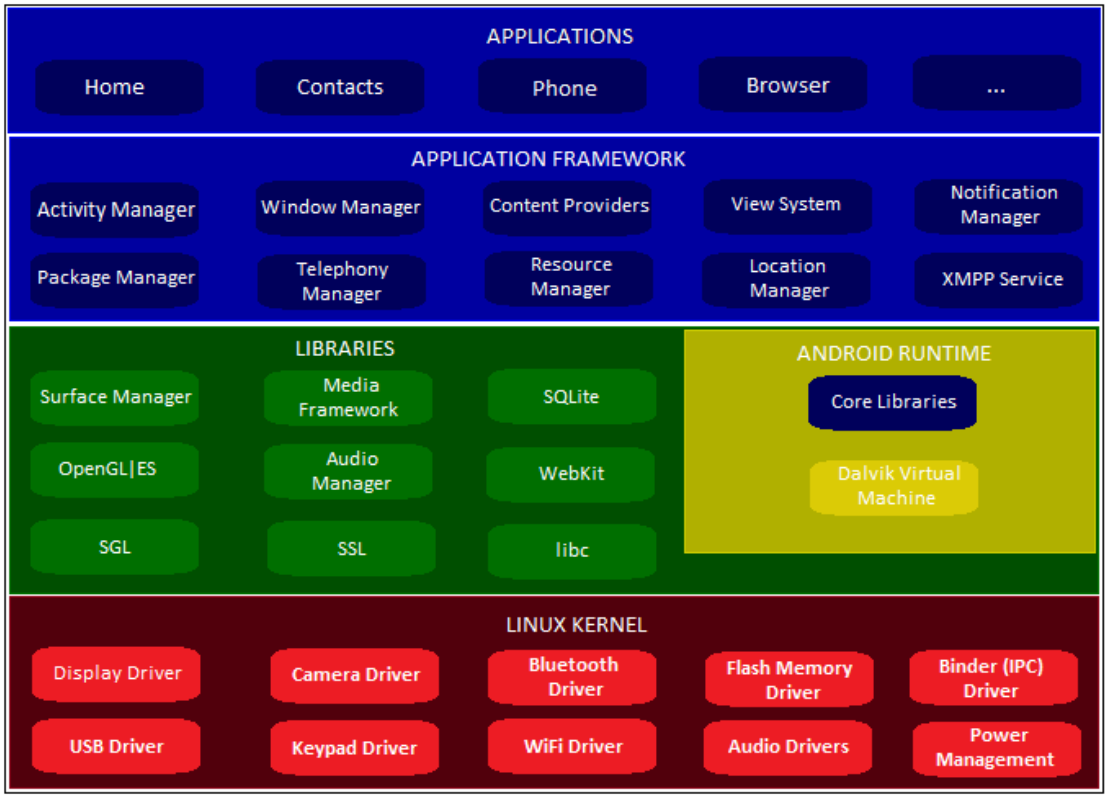
隨著智慧型手機的市占率愈來愈高，手持式作業系統也不斷推陳出新，研究 Android 相關的議題也變得非常重要，其中，Android 採用 Linux 核心架構，並且針對手機上的需要，做許多相關的修改，在手機上，系統擁有的資源不如在一般電腦上的多，為了使 Linux 核心所提供的功能更符合 Android 的需要，尤其在提升系統資源利用度方面，例如：提升 CPU 利用度、記憶體不足時所採取的記憶體回收策略、更省電的電源管理方式等， Linux 在排程方面是採用 CFS (Completely Fair Scheduler) 策略，希望能將每項工作皆公平對待，但是此方法會產生工作之間切換頻率較高的問題；記憶體管理方面則是提供了 Out Of Memory Killer 機制，避免因記憶體不足所引發的嚴重問題，但是引發 Out Of Memory 機制往往只是因為 Low Memory 用光，並非整個系統的記憶體都不足；而電源管理就分為兩種方法：APM (Advanced Power Management) 和 ACPI (Advanced Configuration and Power Interface)，兩種方法分別提供省電狀態和裝置單獨斷電之控制，這些方法對 Android 而言仍然不夠，行動裝置所需要的電源管理策略必須更加積極，並且更細緻的對每項裝置做省電狀態和斷電之控制，所以 Android 對 Linux 核心在這些方面都有所修改：在排程方面，Android 新增了 NORMALIZED_SLEEPER 這項功能，用來降低 Process 切換的頻率；而記憶體管理方面增加了 Low Memory Killer 機制，強化原本 Linux 提供的 Out of Memory 機制；電源管理新增了省電狀態、WAKE LOCK、Timer 控制等方式，藉此更積極的達到省電之目的。

關鍵字：Android、Linux 核心、Completely Fair Scheduler、Out Of Memory、APM、ACPI

1. 簡介

Android 是一個行動裝置的作業系統，與其他作業系統一樣，Android 採用了分層的架構，一共分為五層，分別為應用程式層、應用程式框架層、系統執行

階段程式庫層和 Linux 核心層，其中的 Linux 核心層就是本次介紹的主題，Android 採用了 Linux 核心的架構，但是並非完全和原有的 Linux 核心相同，而是在原有的 Linux 上，針對 Android 的需要做相關的修改。



圖一：Android 系統架構圖[1]

Android 對 Linux 修改的部分很多，主要有以下幾項：

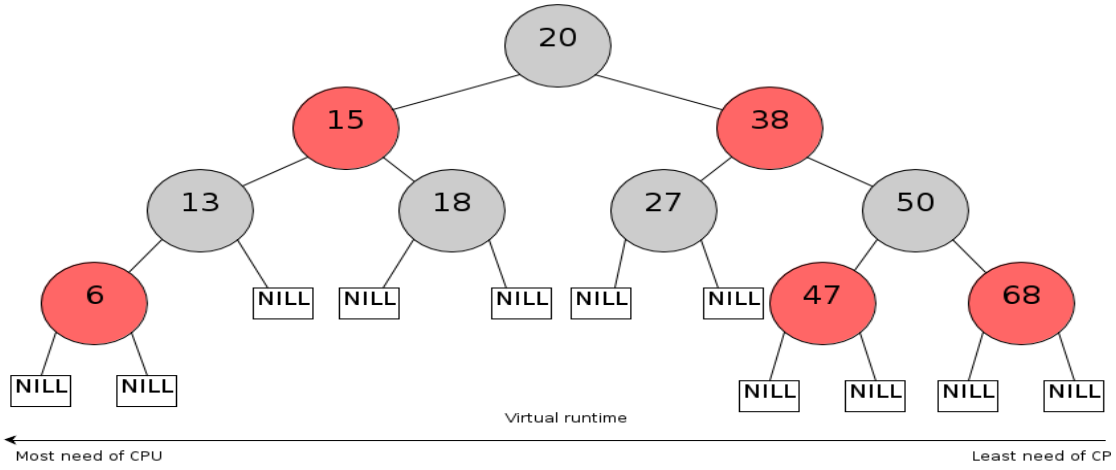
- IPC Binder (Inter-Process Communication 處理程式間通訊)[1] — 用來提供 Android 平台的處理程序間通訊功能。
  原始碼位於 drivers/staging/android/binder.c
- Android Power Management[1][2] — 基於標準 Linux 電源管理系統的輕量化 Android 電源管理驅動程式，針對嵌入式設備做了更多細部的控制與最佳化。原始碼位於：kernel/power/earlysuspend.c
  kernel/power/consoleearlysuspend.c
  kernel/power/fbearlysuspend.c
  kernel/power/wakelock.c
  kernel/power/userwakelock.c
- YAFFS2 (Yet Another Flash File System，2nd edition)[2] — 不同於個人電腦，手機使用 Flash 作為儲存媒體，但是針對 Flash 所設計的檔案系統格式 YAFFS2 的支援並不包含在標準的 Linux 2.6.25 核心中，所以 Android 在其中新增了對 YAFFS2 的支援，總共修改和新增 35 個檔案。
  註：Flash 仍能格式化成一般的 FAT32 或 ext3 檔案系統，但這些檔案系統並未對 Flash 作最佳化。新增的檔案主要在：fs/yaffs2/

- 藍牙[1] – 在藍牙通訊協定裡，Android 修正了一些與藍牙耳機相關的 Bug 以及一些與藍牙偵錯和存取控制相關的方法，總共修改 10 個檔案。
  具體內容在：drivers/bluetooth/
  net/bluetooth/
- Android Alarm (硬體時鐘)[2] – 提供計時器，用來把設備從睡眠狀態喚醒，同時還提供了一個即使在設備睡眠時也會執行的參考時鐘。
  原始碼位於:drivers/rtc/alarm.c
- Low Memory Killer (低記憶體管理)[1][2] – 比 Linux 標準的 Out Of Memory 機制更有彈性，可以依據需求刪除處理程序以釋放所需記憶體。
  原始碼位於：drivers/staging/android/lowmemorykiller.c
- 排程工具[1] – 為了降低工作之間切換的頻率，Android 在原本 Linux 上新增 NORMALIZED_SLEEPER。
  原始碼位於：kernel/sched.c
  kernel/sched_fair.c
  kernel/softirq.c
  kernel/time/tick-sched.c
  include/linux/tick.h

2. Linux 提供的功能

排程管理

Linux 核心提供的排程方法為 Completely Fair Scheduler[7]，簡稱 CFS，其精神是強調對每個項工作皆公平對待，例如有兩個工作同時進行，則每個工作取得 50%的 physical power，即達到完全平行處理的概念，並且採用以時間順序的紅黑樹，將獲得 CPU 執行時間愈少的工作排列在樹的愈左邊。



圖二：Time-ordered rbtree[7]

圖二為一個 CFS 紅黑樹的例子，Scheduler 取紅黑樹最左邊 key 值為 6(獲得

CPU 時間數最少)的節點作排程，並將前一個被替換下來的工作插回紅黑樹中，下一次 Scheduler 要再取工作時，會再取調整後(因有節點被取走、也有被替換的工作加回)紅黑樹最左邊點加入排程。

記憶體管理

Linux 在記憶體管理方面提供一項叫 Out Of Memory (OOM) Killer[8]的機制，此項機制是為了保護系統在記憶體不足時，不至於發生嚴重的問題，其作法是在記憶體不足發生時，OOM 挑選出與目前執行中無關的工作進行刪除，在 2.4 版中，OOM 是把新進的工作作為刪除的對象，2.6 版則是刪除佔用最多記憶體的工作。

電源管理

Linux 在電源管理方面提供了兩種方法，分別是 APM 和 ACPI。

APM (Advanced Power Management)[3] 是一種基於 BIOS 的電源管理方法，提供了多個省電狀態，讓系統在閒置的時候，可以把目前的狀態存進記憶體或是磁碟中，如：Standby、Suspend、Hibernate。由於是基於 BIOS 所提供的服務，此管理方式並非 Linux 所特有，只是 Linux 所採用。

ACPI (Advanced Configuration and Power Interface)[3] 提供比原本 APM 更多的系統狀態，另外增加了裝置電源狀態、處理器電源狀態、裝置和處理器效能狀態，讓作業系統能對某一個裝置做單獨斷電之控制，藉此達到省電之目的。同 APM，在其它作業系統也有類似的電源管理。

3. Android 修改與新增的功能

排程管理

Android 在排程方面增加了一項 NORMALIZED_SLEEPER 功能，增加此項功能的原因在於，手持式系統的資源較少，雖然 CFS 的排程方法可以降低每項工作的平均等待時間，但有時會造成工作之間的切換太過頻繁，然而在手機上，工作之間的切換成本較高，為了降低切換的頻率，Android 在原有的 Linux 核心上加上了此項功能。此功能為一個可開關的功能，關閉此功能時，大型的任務將會獲得較多的 CPU time，小型的任務會有較少的 CPU time。

表一：NORMALIZED_SLEEPER 存在之版本

	Android 是否提供此功能	Linux 是否提供此功能
Android 1.0 → Linux 2.6.25	N/A	否
Android 1.5 → Linux 2.6.27	是	是
Android 2.2 → Linux 2.6.32	是	是
Android 2.3 → Linux 2.6.35	否	否

表一為此項功能出現的版本，值得一提的是，在 Android 2.3 與 Linux 2.6.35

的時候，又將其刪除，目的是恢復原本 CFS 的精神，在這之後，Android 在排程方面，也就幾乎完全採用 Linux 核心的支援。

記憶體管理

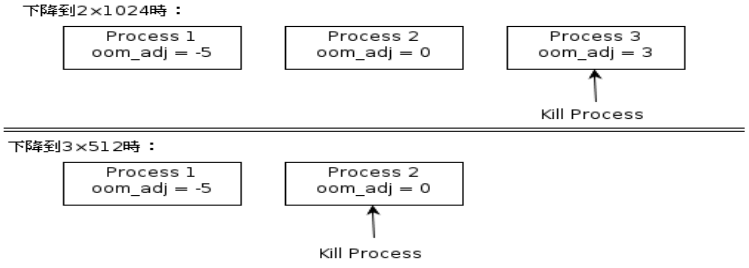
對許多系統而言，記憶體是非常重要的一項設備，Android 新增了 Low Memory Killer 來增強原本 Linux 核心所提供的 OOM 功能。

Low Memory Killer 在使用者空間中指定了一組記憶體臨界值，當其中某個數值與處理程序描述中的 oom_adj 值在同一個範圍時，該處理程序將被刪除，通常在 /sys/module/lowmemorykiller/parameters/adj 中指定 oom_adj 的最小值，在 /sys/module/lowmemorykiller/parameter/minfree 中儲存閒置分頁的數量。

```
01  static int lowmem_adj[6] = {
02          0,
03          1,
04          6,
05          12,
06  };
07  static int lowmem_adj_size = 4;
08  static size_t lowmem_minfree[6] = {
09          3 * 512,        /* 6MB */
10          2 * 1024,       /* 8MB */
11          4 * 1024,       /* 16MB */
12          16 * 1024,      /* 64MB */
13  };
14  static int lowmem_minfree_size = 4;
```

圖三：Low Memory Killer 機制[5]

圖三為 Low Memory Killer 實作程式碼片段，當處理程序的閒置空間下降到 3x512 個分頁時，oom_adj 值為 0 或者更大的處理程序會被刪除；當閒置空間下降到 2x1024 個分頁時，oom_adj 值為 1 或者更大的處理程序會被刪除，以此類推，簡單的說，佔用實體記憶體最多的那個處理程序會優先被刪除。



圖四：Low Memory Killer 演示

電源管理

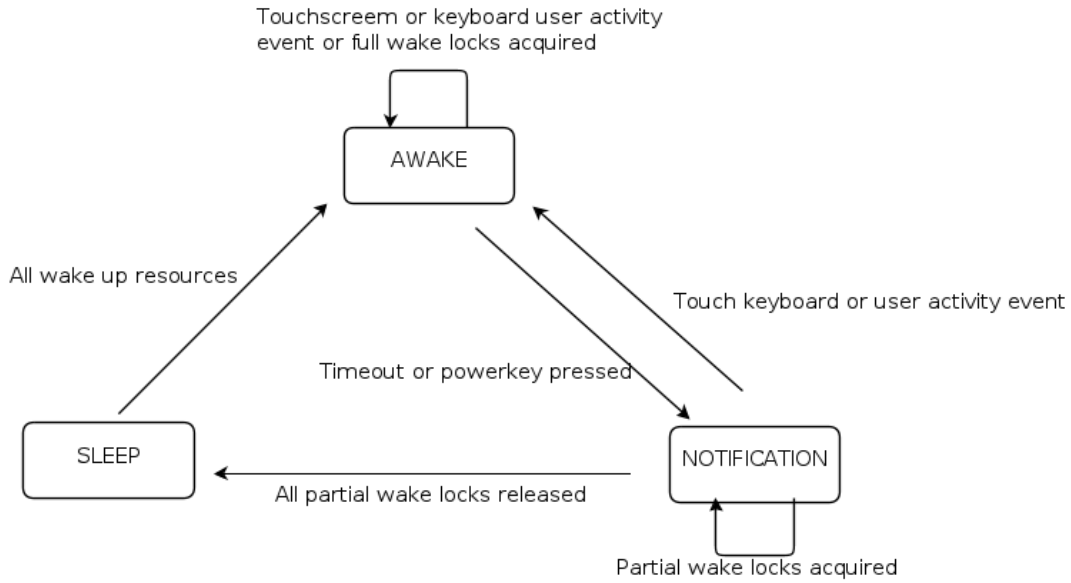
Android 在原有的 Linux 標準電源管理上，修改並新增了一些 Android 專屬的機制，在省電狀態中，Android 定義了幾種低功耗狀態：early suspend（早期暫停）、suspend（暫停）、hibernation（休眠）。其中，早期暫停可以讓某些設備選擇進入某種較為省電的狀態，例如 LCD 可以降低亮度或是熄滅；暫停是指除了電源管理以外的其他外圍模組和 CPU 均不工作，只有記憶體保持 self-refresh 的狀態；休眠是指所有記憶體映像都被寫入磁碟中，然後系統關機，重新啟動後系統將能恢復到關機之前的狀態。

除了幾個省電狀態之外，Android 還採用了 WAKE LOCK 和 TIMER 來更細緻的控制每項裝置。

表二：WAKE LOCK 的 Flag value[3]

FLAG VALUE	CPU	SCREEN	KEYBOARD
PARTIAL WAKE LOCK	On	Off	Off
SCREEN DIM WAKE LOCK	On	Dim	Off
SCREEN BRIGHT WAKE LOCK	On	Bright	Off
FULL WAKE LOCK	On	Bright	Bright

表二為 WAKE LOCK 機制所提供的 flag value，每個 flag value 用來切換處理器、螢幕和鍵盤背光,Android 藉此功能來更加細緻的控制手機裡的每項裝置，以達到省電之目的，其中只有 FULL WAKE LOCK 會把所有裝置的電源打開，當 PARTIAL WAKE LOCK 釋放的時候，則會關閉大部分的裝置，使系統進入 SLEEP 狀態。



圖五：Android Power Management 運作流程圖[3]

圖五為 Android Power Management 運作之流程，一般而言，系統正常開機後會進入 AWAKE 狀態，系統的 screen off timer 開始計時，在指定時間到達之

前，如果有任何 activity 事件發生，則 reset screen off timer，使系統保持在 AWAKE 狀態。若時間到了沒有任何 activity 事件發生，或是使用者按下電源鍵，那麼系統將被切換到 notification 狀態，並且把相關的裝置轉換成 early suspend 狀態，此時系統會判斷是否有 partial wake lock acquired，如果有的話則等待其釋放，在等待過程中若有 user activity event 發生，系統會馬上回到 AWAKE 狀態，如果沒有的話，則系統會關閉相關的裝置，並進入 SLEEP 狀態。系統在 SLEEP 狀態時，如果檢查到任何一個 wake up resource，系統就會被喚醒，回到 AWAKE 狀態。

4. 總結

在本次的介紹中，分析了 Linux 和 Android 在排程、記憶體管理、電源管理方面的不同之處，了解在排程方面，因為 Android 的系統資源較少，為了減輕 Process 切換的負擔，提供 NORMALIZED_SLEEPER 來減少切換的頻率；在記憶體方面，則是利用 Low Memory Killer 增進原有在記憶體不足時所採取的策略，使其更有彈性；而在電源管理方面，Android 做了較多的修改，除了增加裝置的省電狀態以外，還新增了 WAKE LOCK 機制，讓系統可以把目前沒有用到的裝置暫時關閉，藉由更積極的控制各項裝置的方式，來達到更為省電的目的。

參考資料

[1] 楊豐盛, “ Android 技術內幕—探索 Android 核心原理與系統開發,” 碁峯資訊股份有限公司, 2011.

[2] F. Maker, “A Survey on Android vs. Linux,” University of California, 2009. [Online]. Available: http://handycodeworks.com/wp-content/uploads/2011/02/linux_versus_android.pdf

[3] M. Motlhabi, “Android Power Panagement,” cs.uwc.ac.za. [Online]. Available: http://www.cs.uwc.ac.za/~mmotlhabi/apm.pdf

[4] Android: Dalvik VM Internals, https://sites.google.com/site/developerdaytaiwan/google-developer-day-2008-taiwan/android-dalvik-vm-internals

[5] The Linux Kernel Archives, http://www.kernel.org/

[6] Android open source project, http://source.android.com/

[7] Linux 2.6 Completely Fair Scheduler, http://www.ibm.com/developerworks/cn/linux/l-completely-fair-scheduler/

[8] Linux Out-of-Memory(OOM) Killer
http://blog.csdn.net/guomsh/article/details/6536915
[9] Wong C.S., Tan I.K.T., Kumari R.D., W. Fun, "Towards Achieving Fairness in the Linux Scheduler", in ACM SIGOPS Operating Systems Review: Research and Developments in the Linux Kernel, vol. 42, issue 5, July 2008.
[10] Wong, C.S., Tan, I.K.T., Kumari, R.D., Lam, J.W., Fun, W., "Fairness and interactive performance of O(1) and CFS Linux kernel schedulers," in Information Technology, 2008.