ESWEEK 2008 的參訪報告

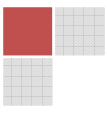
內容

前言

今年的 ESWEEK 2008 在亞特蘭大舉行，10/19 – 10/24 六天共包含四場 Tutorials (第一天)、一場 Turing Award Lecture (第二天)、三個會議 CASES, CODES+ISSS, and EMSOFT (第二、三、四天)以及七個 Workshops (第一、五、六天)。

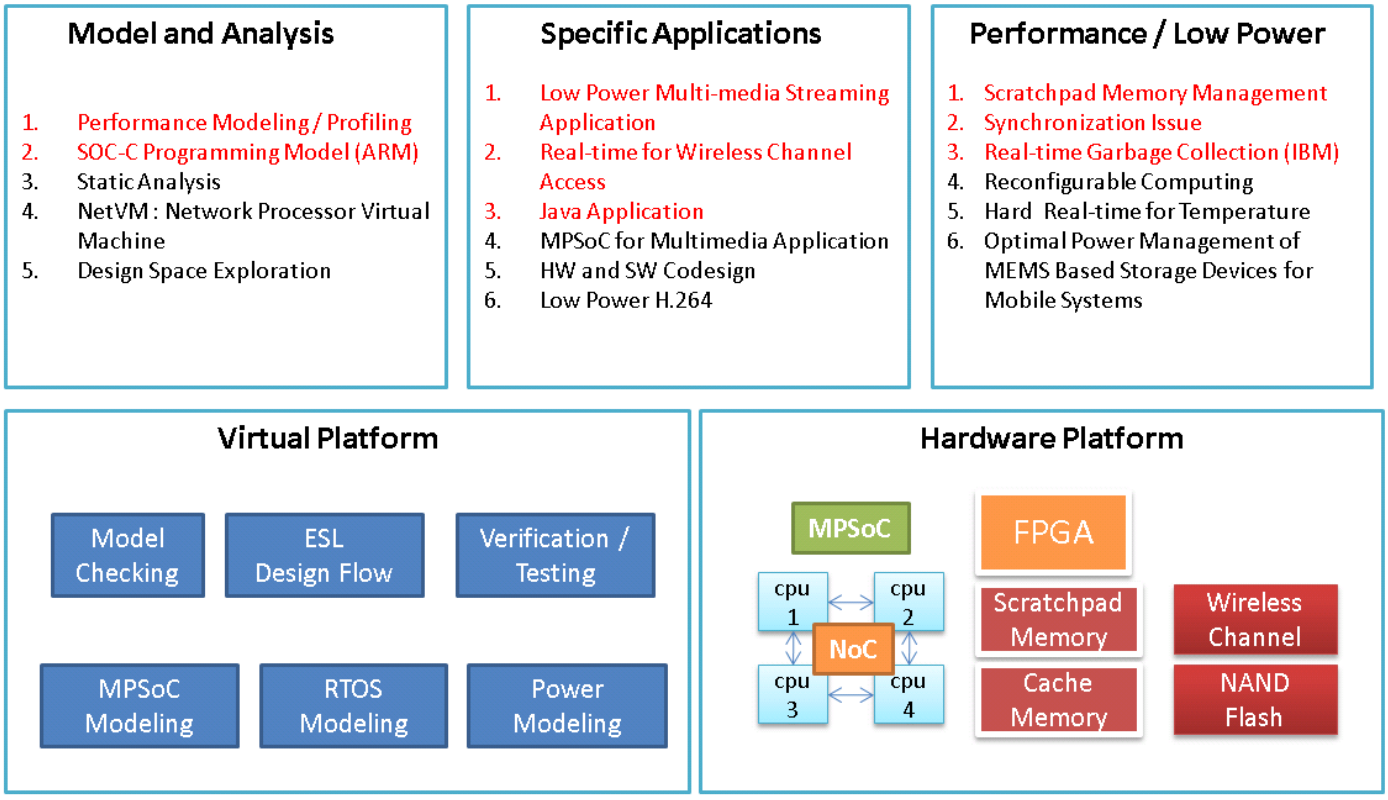
在這六天內，ESWEEK2008 所涉及的主題非常廣泛，從編譯器(Compiler)、NoC (Network on Chip)、MPSoC (Multiprocessor System-on-Chip)、系統層級建模 (System Level Modeling)、驗證(Verification)、排程理論(Scheduling Theory)、NAND Flash Storage、Scratchpad Memory、Low Power、Cache Profiling ……等等，呈現出嵌入式系統的研究領域博大精深，且在各領域有其專特性，透過不同研究團體的分工合作，讓嵌入式系統的發展越來越快速。

然而在如此複雜的研究體系架構中，如何剖析前瞻的研究主流與趨勢，的確是一件重要的工作。因此我嘗試將所見聞的內容在本文中透過歸納與分析，尋找出嵌入式系統研究的脈絡，進而提出具體的研究方向與計畫，期許我們的研究團隊能夠將論文發表至 EMSOFT 2009，並且期許所進行的研究也能夠應用在手機的發展趨勢中，以結合研究與實務的特性。



研究架構

鑒於 ESWEEK 2008 所包含的主題非常廣泛，所以經過大量的資料分析與想法的統整後，嘗試繪出下圖作爲 ESWEEK 2008 之研究架構。此架構圖區分爲上下兩層，下層屬於抽象的概念，研究人員可以跟據此研究環境建立相關的研究議題；上層是較具體的研究議題，乃是從 ESWEEK 中擷取出一些有趣且值得深入探討的想法，作爲未來制定研究議題的參考。然而此架構圖雖力求完備，但是仍然沒有包含所有會議中的主題，例如在 Testing 與 CMOS 相關的議題就沒有納入考慮，主要還是依據 MTK 子計畫 A 的適用性來做資訊的取捨。接下來會一一解說架構圖每一區塊的內涵。



下層區塊：抽象概念

Hardware Platform Block

隨著硬體製程的進步，多核心架構已經成爲主流趨勢，要建構 High Scale 的核心架構，就必須要了解如何有效地使核心與核心之間產生連結，這也就是在 NoC (Network on Chip)所討論的範圍，因此 MPSoC 就是要建構一個具備 Multi-core 與 NoC 的處理器單晶片。此外，FPGA 的成本下降與可程式化的邏輯閘之數量提升，使得 FPGA 也逐漸成爲異質運算的重要角色，在過去主要是使用 FPGA 作爲軟硬體協同設計的雛形(Prototype)，但是現今已經可以用來當作 Application 的最佳化執行的硬體平台，也就是說在動態時期能夠將 Application 負載較重的運算切割到 FPGA 上高速執行，這也稱爲 Reconfigurable Computing。

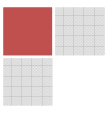
在嵌入式系統中，記憶體的存取效能是非常關鍵的因素，所以在階層式的記憶體中有效地使用快取記憶體(Cache Memory)一直是很重要的工作。在進入多核心的時代裡，快取記憶體的使用更容易出現爭競的現象而導致大量的 Cache Miss，發生 Cache Miss 就需要付出代價來存取 off-chip memory，將會大幅提高 Application 的執行時間延遲(latency)，此外對於即時應用程式也會出現執行時間不可預測的問題，因此對於 Cache 在多核心架構下有深入的分析研究，將助益整體系統的效能。

Scratchpad Memory 與 Cache 功能相近，但是又具備獨特的特色，所以在嵌入式系統中也被廣泛地採用。最重要的特色之一就是 Scratchpad Memory 可以容易地讓 programmer 來操控，有別於 cache line 的存取是交付硬體來控制。鑒於此高速記憶體能夠具有彈性地規劃，因此如何在有效地運用其有限的空間來加速 Application 對於記憶體的存取，就是一件很重要的工作。一般在規劃上，有採用編譯器做爲靜態分配的方法，也有透過作業系統來進行動態分配的方法。

至於 NAND Flash 近幾年也成爲研究的熱門主題，並且也相當具有實用性，未來幾年 NAND Flash 也將成爲手機上主要的儲存設備。因此在 NAND Flash 發展的趨勢下，這股研究與實用的熱潮應該還會持續幾年。

另外從手持裝置手機的發展來看，無線網路的服務也會有蓬勃的發展，所以在無線網路設備的使用，也會成爲熱門的話題。從研究的角度出發，可以思考無線網路存取的架構如何與 Application 一併考慮，或者更有趣的想法是提出 wireless-aware application

從上述所說的 Multi-core, NoC, FPGA, Cache, Scratchpad Memory, NAND Flash, Wireless Channel



等等，可以視爲研究討論的抽象概念，也就是說所制定的研究議題可以圍繞此平台環境來討論，在這一範圍裡面，進一步思索研究議題所要討論的具體項目，例如 Power Consumption, Real-time Scheduling，至於研究議題的考量維度就取決於議題是否具有研究價值。透過此研究架構來討論與思索，更容易幫助我們釐清觀念，也更清楚知道我們研究的定位，且研究的方向至少與目前研究的趨勢相去不遠。

Virtual Platform Block

從會議中也不難發現，虛擬平台的建構也是一個相當重要的議題，藉由虛擬平台來做 Model Checking, 系統層級的模擬與驗證，甚至是專特地建構 MPSoC 的虛擬平台也是主要的研究範圍。其實近來所討論非常熱烈的 ESL 就是一個很明顯的範例。

此外在 Virtual Platform 對於 RTOS 的建模與 Power 的模型也有熱烈討論的議題。

然而此部分比較與 MTK 的子計畫 A 內容沒有直接的關係，所以這部分就不詳述說明。

上層區塊：具體議題

上層具體的議題，其實就是從 ESWEEK 2008 中，看出大家在制定研究題目時大致的方向，基本上就是在我所提出下層抽象的架構上來延伸出特定的議題。在議題裡面我特別選出三類來說明，事實上在相同原則裡，我們也能夠制定出我們所感興趣的議題且符合目前研究的發展趨勢。

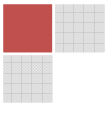
Performance / Low Power Block

效能與耗能是這些年來大量被討論的議題，在會議中有三個題目可提出來進一步地討論，包括

1. Dynamic Scratchpad Memory Management
2. Synchronization Issue for MPSoC
3. Real-time Garbage Collection (IBM) for Java Virtual Machine

從 Dynamic SPM 的議題中，發現嵌入式系統如果能夠善用 Scratchpad Memory，將會有大幅效能的提升，或許在子計畫 A 中的 Memory Management 小組也可以嘗試將 SPM 納入研究的範圍中，不僅具有理論研究的價值，也非常具有實用性。

從多核心程式的發展趨勢來看，程式之間的同步問題也將成爲效能的一大瓶頸，同步問題也常



常成爲程式開發者的一大困擾，所以如果要有效地使用多核心架構，那在同步問題的最佳化上就顯得非常重要。

另外 IBM 在會議中有提出 Real-time Garbage Collection for Java Virtual Machine 的研究，其實在手機上的應用程式大部分都是以 Java 的執行檔在運作，所以此類的研究也更有具於解決未來 Java 在使用上的效能問題。

會議中也有相關的論文，不僅要達到效能提升，也盡可能考慮降低 Power 的消耗。

Specific Applications Block

在特定的應用領域中可以發現這三篇議題也有代表性，分別是

1. Low Power Multi-media Streaming Application
2. Real-time for Wireless Channel Access
3. Java Application Performance Analysis

未來在 Wireless Streaming 的應用上會越來越廣泛，且大多數的 Application 都是用 Java 執行檔的方式在執行，所以在特定的應用上，我認爲研究未來手持裝置上 Java 的無線網路應用，也可以算是一個可見的趨勢。

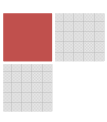
Model and Analysis Block

在建模與分析這裡可以發現，有兩件事情越來越重要，就是提供執行程式的效能剖析工具與多核心程式設計的模型，因此在會議中也有論文提到

1. Performance Modeling / Profiling
2. SOC-C Programming Model (ARM)

SOC-C 是 ARM 最近爲了多核心架構的程式設計，對 C 語言所提出的擴展。

不過在會議中最後一場的 Panel Discussion 裡，各家廠商對於多核心架構與多核心程式模型得討論裡，並沒有達到共識，也就是說這部分還算處於戰國時期，沒有任何一家的解決方案尚可被視爲最可行的方式，或許再過幾年這部分的研究才會慢慢有定案。



趨勢分析

前文所闡述的研究架構中，就已經透露出一些可見的趨勢，所以進一步分析就是將這些要點列舉如下，作爲未來研究討論的方向。

趨勢一：多核心架構已經成爲研究主流，並且在運算上也逐漸朝向異質型運算發展，例如 Multi-core + DSP / Multi-core + FPGA / Multi-core + GPU 等等較常見的運算架構。

趨勢二：最佳化記憶體效能仍然是研究的重心，Cache 與 Scratchpad Memory 在使用上的優化將爲整體效能帶明顯的改善。

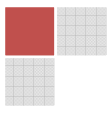
趨勢三：Flash Storage 在 Embedded System 上是當紅的趨勢無庸置疑，在這一部分的細節說明，可以參照同行前往的另外兩位成員之報告。

趨勢四：無線網路在手持裝置上的應用是一大趨勢，然而目前大部分的程式都是以 Java 方式來執行，所以同時考慮 wireless aware Java application 應該是一個很有趣且具有研究價值得議題。

趨勢五：效能剖析工具與耗能剖析工具的研究也是很重要的工作，一個合用的 profiling tool 將會幫助程式開發者找到效能瓶頸，也可以做爲研究分析的利器。

趨勢六：多核心程式開發模型，處於百家爭鳴的階段，目前尚無共識，所以也是一個研究的趨勢。

趨勢七：Green 議題是一個大家都知道的議題，但是目前還是沒有一個很有效的解決方案，所以高效能低耗能的議題，在未來仍然是熱烈討論的議題。



具體規劃

在具體規劃部分我提出兩個方案，主要是考量兩個議題

1. Dynamic Scratchpad Memory + Java Virtual Machine
   目標是使用動態地調整 SPM，使得 Java Application 運作在 JVM 上時，能夠改善記憶體存取的效能頻頸。
2. Dynamic Power Information Plug + Scheduler
   一個創新的想法，我們將制定一個應用程式可以提供的 Power 資訊，在執行時期能夠動態的 Plug 到 Scheduling Policy 裡，以提供 Power-aware Scheduling 來改善應用程式的耗能問題。

這兩個議題的概念是從 Emsoft 2008 中兩篇相關的論文中所發想。

1. **Scratchpad memory management in a multitasking environment, Emsoft 2008**
2. **Wireless Channel Access Reservation for Embedded Real-time Systems, Emsoft 2008**

從 Emsoft 中可以發現，目前 Scratchpad Memory 的討論還是很熱烈，並且此類論文接受的程度也頗高，所以我們期許透過一些創新的想法，讓 Linux 與 Android Delvik VM 能夠在 SPM 上提供支援，如此一來不僅可以達到記憶體使用上的優化，還可以具有 low power 的優勢。

此外在 Emsoft 中，很多關於 power aware scheduling 方面的論文，我們在 power 與 scheduling 都已經有些初步的成果，所以如何在 Linux 與 Android Delvik VM 中產生 power model 並且提供 schedule framework 來達到 low power，將是一個重要的議題。

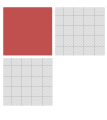
這兩個方案的詳述如下：

Dynamic Scratchpad Memory + Java Virtual Machine

主題：Scratchpad Memory allocation for virtual machine process

Motivation

Virtual machine 是一個特殊的 process，它會代替 OS 的角色去管理執行在 virtual machine 上的 thread，因此站在 kernel 的角度上，它會先把 virtual machine 視爲一個普通的 process，而不會注意到它會去影響其所管轄的 java threads，因此當 OS 判斷 java thread 的需求，並且給予它最適當 SPM 資源時，virtual machine 卻很有可能會對此資源進行重新分配，甚至是因爲 garbage collection 而影響了 SPM 資源使用的情形，而現今的 SRM allocation 機制，都是 for general process，



因此我們希望建立一個 SPM allocation 機制 for java thread，使得 SPM 資源的分配不會因爲 virtual machine 介入而受到影響。

概念

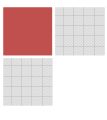
1. 了解 virtual machine 如何管理 memory 資源在 thread? virtual machine 爲了使其管轄的 thread 能夠有效利用 memory 資源，必定有其一套依循的法則，除此之外 virtual machine 還有 garbage collection 會影響 memory 的使用情形，因此我們希望深入分析 virtual machine 在這些方面的特性，並讓 OS 透過這些特性,分配更適當的 SRM 資源給 java thread 或 virtual machine，降低 virtual machine 的介入所產生的影響，並且使得其所管轄的 process or threads 能夠更有效的利用 SRM。
2. 透過 virtual machine profiling interface 提供適當資訊給 SPMM，並 dynamic 調整 virtual machine 所使用的 SRM。 由於現今的 virtual machine 都有其專屬的 profiling interface 可以用來提供內部 process or threads 各項資源的使用情形，因此我們希望 SPMM 可以去利用這些 profiling interface 來獲得 virtual machine 內部的執行資訊，並適時調整 SRM 資源的分配。
3. 在 compile time 進行 code 的 SRM 最佳化，來促使 SRM 的效率提高在 compile time 標記好哪些 data 適合放在 SRM 上，或者計算每個 code function 與 data 的存取次數，方便在 run time 能夠依照這些資訊來進行 SRM 的分配。

執行前先要了解的事

1. virtual machine 的 memory management 的運作情形。
2. 現今的 SRM dynamic allocation 機制應用在 virtual machine 上，是否可以運行得很好。
3. virtual machine 的 profiling interface 可以提供內部執行資訊到什麼程度。
4. OS 上有那些 system call 或者是 function 可以跟 virtual machine 進行溝通。
5. 確定 virtual machine 的版本

執行階段

1. 進行研究 virtual machine memory management 的運作情形，重點是確定 virtual machine 是否會在 Run time 時 更動 java thread 所分配到的 memory 資源。
2. 確定 garbage collection 對於 SPM allocation 所造成的影響，例如會否回收 SPM memory。
3. survey 現今 paper 對於 SRM dynamic allocation 做到什麼程度。
4. 測試現今 SPM allocation 機制用在 virtual machine 上，是否運作得很好，是否常常做出錯誤的分配，或者效率必沒有提高很多。
5. 當前兩階段執行完之後，如果真的很有需要改進的地方，就進行研究 virtual machine memory management 的運作情形。
6. survey virtual machine interface 的支援程度，看它能提供什麼資訊。
7. 研究 java code 的在編譯時，那些 code 可以先進行 SRM 的最佳化



8. 實作 code optimizer with SPM allocation for java code.

Dynamic Power Information Plug + Scheduler

主題：Linux Kernel Power-aware Scheduler on Mobile Platform

Motivation

從過去十年的論文發表可知，與 power 相關的 scheduling 議題大約佔全部總數的百分之五十，由此 可知兩者密不可分，再者 green issue 愈來愈被重視，所以希望能夠跳脫過去的 DVS(DVM?)，提出一個新的 model 達到 low power scheduling 這最終目標，此外或許透過這全新的機制可以降低 runtime 時計算上所需的 overhead，也能夠結合現今已有的 power profiling tool 來加速此 model 的開發

概念

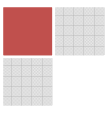
1. 如何提供資訊
   在現存的 scheduling model 下，將 power profiling tool 提供的資訊轉換成一個 plugin，能夠動態 的與原先的 scheduling model 做連接，如此透過 profiling 得到的資訊可以準確並有效的進行 power consumption 上的修正，達到 low power 的目的。
2. 得到資訊後，如何找到運行的機制
   (1) 可以藉由每次 AP 執行時所記錄下來的歷史資訊作爲機制預測的參考
   - AP 本身帶有一些程式設計者先行定義好的特性資訊，可以以此當做歷史資料，或者可以作爲 AP 歷史資訊的參考基礎，都能夠降低維護歷史資料的 overhead

   (2) AP 每次在執行時都動態的去做 profiling，並將得到的資訊馬上對之後運行的機制做修正
   (3) 將 1 與 2 結合，以基礎資料做爲基底，動態 profiling 稍作微調，如此便能快速且精準的決定所要 運行的機制

執行前先要了解的事

1. 該提供給 scheduler 哪些 power information
2. 如何去量化 power 的資訊
3. 如何建立 profiling tool 與 scheduler 之間溝通的 channel
4. 取得資訊後，要讓誰決定要不要將 profiling informantion plug 上 scheduler?
5. 需不需要提供另一個 user space 與 kernel space 之間溝通的窗口，此爲針對上面的 1-1

執行階段



1. 參考 powerTop 了解提供的 power information 可以有初步的量化概念，並且了解 powerTop 究竟提供了多少種 power 相關的資訊可供 power-aware scheduler 使用
   (1) 提供了多少資訊，提供的資訊能供 scheduler 使用嗎?
   (2) 如何量化?量化的方式符合我們的訴求嗎?需要另外提出自己的 profiling tool 嗎? 此部分時程可參考 MM 組的 LatencyTop & OProfile 的 study 規劃
2. 了解 2.6.2x 的 scheduling model(俊宏學長先前已掌握)，是否能夠不修改大架構就能提出我們的模組並其中至入我們需要的溝通窗口，或者仿造 kernel scheduling model 的大架構做修改
   (1) 了解 kernel scheduling 整體的階層架構關係
   (2) 能否像修改 CFS 一樣，集中在一個模組裡面做修改，還是要做更底層的修正
   規劃上以 trace code 爲主要運行方向
   p.s.如果需要提供 AP 與 kernel 的窗口，這邊又要補充!!
3. porting 上現實的機器上，需不需要與精密儀器測量的結果對照，了解 profiling 的誤差?
4. 在 2 完成後，溝通的窗口已建立，將運行機制分三個步驟來實現
   (1) 單純透過 history 來調整機制，包含 history 資訊的收集
   (2) 單純透過 runtime profiling 爲此次運行，不斷的作機制上的修正
   (3) hybrid

國外學者的互動紀實

在出國參加會議期間，認識了從新加坡來的兩位研究生，它們分別是研究關於 Cache 的效能分析，還有 Scheduling 的議題，也從他們的分享裡面了解新加坡的做研究方式。在會議後有研究人員將自己的研究內容使用海報展覽分享，我對於 GPU 方面的議題感興趣，因此也趨前詢問更深的內容，使我更了解 GPU 這種計算平台的技術。

此外也與清大得李政崑教授有較多的互動，主要是討論關於多核心架構的發展，以及未來是否有一些研究的方向與趨勢，這部分也使我收穫良多。

在互動上我獲得幾個想法的要點如下：

1. 目前 Multicore / MPSoC 仍然是非常熱門的議題，甚至在最後一場 panel discussion 中，各家廠商與幾位學者都提出一些想法，但是李政崑教授與我都發現在這 Multicore 的戰國時代中，雖然許多研究人員都有提出見解，但是在各派系中尚未存在一個最主流的方法，所以現階段仍然有很多研究問題可以繼續深入，例如在 multicore programming model 上就是非常顯著的例子。
2. 在 GPU 的討論中，發現 GPU 的運算會越來越被重視，由於它是屬於 many cores 的架構，因此如此有效將程式分析與切割來符合這樣的架構，就是一個研究議題，未來我也將會在這議題上有深入的研究。