# 在雲端計算環境下使用改良的 D_EDF 排程演算法之 Xen 排程器

曾嘉影
大同大學資訊工程研究所
e-mail：cytseng@ttu.edu.tw

黃柏鈞
大同大學資訊工程研究所
e-mail：g10006025@ms.ttu.edu.tw

## 摘要

隨著雲端計算、移動裝置與工業自動化發展，虛擬化技術顯得日益重要。虛擬化的核心在於虛擬機監視器，直接影響平台的效能。因此，如何有效的分配硬體資源成為重要的課題。Xen 是一個虛擬機監視器，是被廣泛應用的開放原始碼專案之一。本篇論文中透過修改 Xen 提供的 Simple EDF (Earliest Deadline First) 排程器，結合截止單調(Deadline-Monotonic)排程演算法(D_EDF)，並且實作在 Xen 平台上。實驗結果表明，在過載的環境下，本文所提出的改良的 D_EDF 排程演算法提昇了效能，且縮短了核心的延遲。

**關鍵詞**：排程演算法，虛擬化，Xen，Earliest Deadline First，D_EDF

## Abstract

With the development of cloud computing, mobile device and industry automation, virtualization is more and more important today. The core of virtualization is hypervisor which directly determines the platform performance. How to allocate resource effectively becomes an important problem. Xen hypervisor is an open source project. In this paper, we have improved the D_EDF algorithm and implemented on Xen. We also combine Deadline-Monotonic Scheduling with Simple Earliest Deadline First scheduler which Xen provides. Our experiment demonstrates that the proposed algorithm increase the performance and reduce kernel latency better than Simple EDF in overloaded condition.

**Keywords**: Scheduling Algorithm, Virtualization, Xen, Earliest Deadline First, D_EDF

## 1.緒論

隨著雲端計算、移動裝置與工業自動化發展，虛擬化技術 (virtualization) 被廣泛應用，包括：企業的資訊基礎建設、嵌入式虛擬化與嵌入式系統 [1] 的部署。虛擬化的特色包括：虛擬機 (Virtual Machine) 都是獨立的、多台虛擬機使用一台實體機器。因此，帶來許多好處，如：電源管理、節省成本、易於管理和提高 CPU 利用率…等。

虛擬化技術的核心是虛擬機監視器 (Virtual Machine Monitor, VMM)，負責分配硬體資源並管理虛擬機。因此，虛擬機監視器直接決定了整個平台的效能。虛擬機監視器主要分為原生監視器 (Native hypervisor, type-1)和託管監視器 (Hosted hypervisor, type-2)。Type-1 類型的監視器是直接在硬體上執行並分配資源給虛擬機，而且有自己的排程器。例如：Xen Hypervisor 和 VMware ESX。Type-2 則是執行於作業系統層上，而且由於沒有自己的排程器，所以是依靠作業系統的排程器。例如：VirtualBox 和 VMware Workstation。

正因為虛擬監視器是影響效能關鍵，為了適應實體機器不斷的提升，例如：實體 CPU 核心增加、實體記憶體擴增…等，因此虛擬化環境的排程演算法成為重要熱點，如何更有效分配資源成為一個重要的議題。

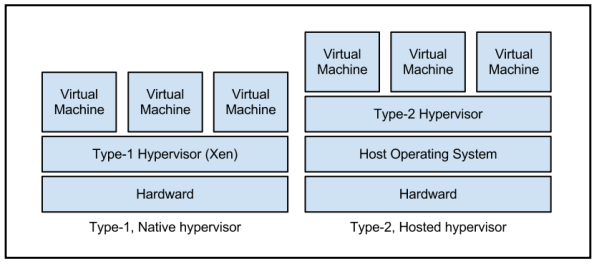


圖1. 原生監視器與託管監視器架構

## 2. Xen 的虛擬化技術

這個章節會先介紹 Xen Hypervisor 和 Xen

全虛擬化與半虛擬化，接著介紹 Xen 的排程器及其優缺點，最後是相關的即時排程演算法與其優缺點。

## 2.1 Xen 虛擬化

Xen 的虛擬化技術分為半虛擬化和全虛擬化。相對於不需要修改客端作業系統的全虛擬化，須修改客端作業系統核心的半虛擬化的客端虛擬機有比較好的效能。另外，全虛擬化是基於硬體輔助方法 (Intel VT and AMD-V)的實作，不用修改客端作業系統核心 [2]。本文所使用的實驗平台為半虛擬化虛擬機器。

主要有三種虛擬化類別，分別為：中央處理器 (CPU)、記憶體 (Memory)和 I/O [3]。本文將焦點集中於虛擬 CPU (virtual CPU, VCPU)。

## 2.2 Xen 的排程器

Xen 目前提供 Credit 和 Simple Earliest Deadline First (SEDF)兩種排程器，前者為 Xen 4.2.1 版本的預設排程器。在多核心 (SMP)架構下，Credit 能達到全域負載平衡 [4]，但延遲敏感 (latency sensitive)和在 I/O 密集的環境下效率不佳 [5]。Credit 有兩個主要參數可供調整，分別為 weight 和 cap **錯誤！找不到參照來源。**。weight 代表虛擬機器之間使用 CPU 的權重，預設值是 256。如果一個 domain 的 weight 參數設定成 256，則取得的資源為設為 256 的兩倍。cap 參數代表可使用實體 CPU (physical CPU, PCPU)的百分比，決定該 domain 可以取得多少資源。預設值為 0，代表沒有限制，亦即 WC-mode。若設為 50，則是半個 PCPU，即 NWC-mode。若是 100，則是一個 PCPU。以此類推，若設定為 400，則是 4 個 PCPU。

SEDF 是一個動態優先權的即時排程器，是基於知名的 Earliest Deadline First 排程演算法的實作 [7]。SEDF 提供 5 個參數，分別為 *period*、*slice*、*latency*、*extra* 和 *weight*，搭配不同參數設定可達到不同效果。在即時的環境下，會使用 *period* 和 *slice* 做明確的時間指定，*period* 決定了 VCPU 的任務週期，*slice* 則是最差狀況下的執行時間；*extra* 是一個布林值，決定了該 domain 是否可以使用額外的時間，因此將 *extra* 設為 1 的 domain，皆會以循環分時排程 (Round-robin scheduling)分享額外的時間。另外，*weight* 提供另一種方式間接設定 *period* 和 *slice*。實際的 *period* 和 *slice* 透過 *sedf_adjust_weights()*函式依據 *weight* 作調整。

儘管 Earliest Deadline First (EDF)被證明在單處理器、搶奪和未過載的環境下，為最佳化的演算法；在高負載的環境能保證所有任務在截止期前完成；但是在過載的環境，會產生截止期失誤 [8]。

## 2.3 相關排程演算法

截止期單調 (Deadline-Monotonic, DM)排程器是屬於靜態優先權搶奪演算法。優先權是由相對截止期決定。相對截止期愈短，優先權愈高。在高負載環境下，雖然靜態優先權無法像 EDF 演算法一樣即時反應。但在過載環境下，卻能保證高優先權任務如期完成，使得較低優先權任務截止期失誤。因此，過載的環境條件下，效能表現比 EDF 佳。

Susi Xi [9] 提出 RT-Xen，提供四種即時排程策略，其中 Deferrable Server 屬於靜態優先權排程演算法，效能優於其它三種，且有較高的承載。然後將客端作業系統內的任務，以 RM 排程決定任務的優先權，實現繼承式的即時排程。

Devendra Thakor [10] 提出 D_EDF 排程演算法，是將 EDF 結合 DM 的排程演算法，並依據紀錄的截止期失誤和截止期達成個數切換排程策略。當連續兩次截止期失誤時，由 EDF 切換至 DM 策略；當連續 10 次截止期達成，由 DM 切換至 EDF。如此結合兩種排程演算法的優點，以達到效能提升的目的。

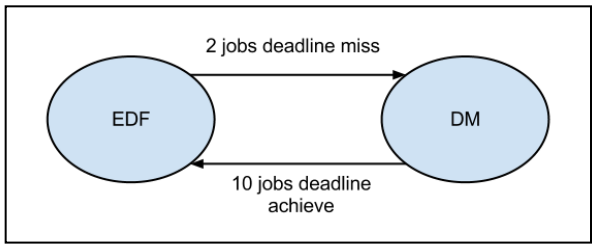


圖 2. D_EDF 的切換準則

如此好的概念，卻存在門檻定義的問題。例如在 Xen 虛擬化環境下，虛擬機個數會時常變動，因虛擬機的開機與關機，連帶影響 VCPU 的總數。在軟即時系統下，考慮兩種情況，5 個 VCPU 和 512 個 VCPU，並且定義截止期失誤比率 6%為嚴重失誤。假若 5 個 VCPU 內有 2 個 VCPU 連續截止期失誤，代表截止期失誤比率為 40%，則用 DM 來保證高優先權任務如

期完成。假若 512 個 VCPU 內有 2 個 VCPU 連續截止期失誤，代表截止期失誤比率為 0.3%，則不需要切換排程策略。因此，切換準則直接使用截止期失誤比率，比計算截止期個數來的好。

本文改良了 D_EDF，以截止期失誤比率(deadline miss ratio)作為判斷準則，並且實作在 Xen 虛擬化環境上。我們將在下一個章節介紹。

# 3. 改良的 D_EDF 排程演算法

虛擬化帶來了許多好處，包括：提高 CPU 利用率、可擴充性和節能...等。在享用這些好處時，若要滿足軟即時或硬即時的需求，必須做更多的測試。如同第二章所述，EDF 在過載環境下，會產生截止期失誤，如何保證高優先權任務如期完成，正好是 DM 可以達成的目標。在 Xen 虛擬化環境下，VCPU 數量是動態的，數量可以是 1 到 512 個，如何改善 D_EDF、準確判斷系統為過載將在 3.2 節敘述。

## 3.1 系統的模型

在虛擬化系統，至少有兩層的排程，若是巢狀虛擬化 (nested virtualization)，則會有多層的排程器。這裡，我們使用的模型是兩層，客戶端作業系統的排程器負責安排任務或應用程式使用 VCPU，虛擬機監視器的排程器負責安排 VCPU 使用 PCPU。每一個 domain 皆分配一個 VCPU，作業系統核心都是未修改的 generic 版本。所有 domain-U 的 VCPU 0 都是使用 PCPU 0，另外，將 Domain-0 的 VCPU 0，pin 至 PCPU 1。

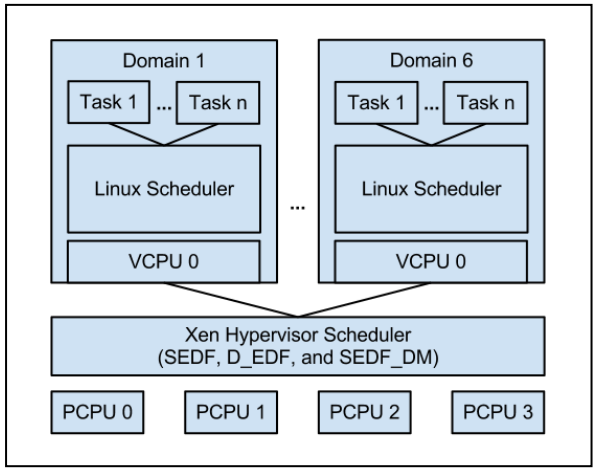


圖 3. 系統的模型與兩層的繼承排程

## 3.2 系統的運作

為了避免因微小變化，而頻繁的來回切換兩種策略，造成不必要的開銷，因此適當的定義切換門檻是非常重要的。首先，排程器必須在每一個 *period* 統計可執行佇列內，VCPU 截止期失誤的個數和全部的 VCPU 個數，以便換算成截止期失誤比率 (deadline miss ratio)。第二，排程器會紀錄至少 256 個以上的 VCPU，當統計的 VCPU 總數超過 256 個，便會結算截止期失誤比率，並判斷是否達到切換門檻，也就是觀察過去一個週期 (256+ VCPU)的行為，決定下一個週期所使用的排程策略。若失誤比率大於 6%，則判斷目前系統為過載(overloaded)，由 SEDF 切換至 DM 策略，以保證高優先權 VCPU 如期完成；若失誤比率等於 0%，則判斷目前系統已消化掉多數的任務，由 DM 切換至 SEDF 策略。最後，每結算截止期失誤比率，便會重新統計截止期失誤次數和總數。

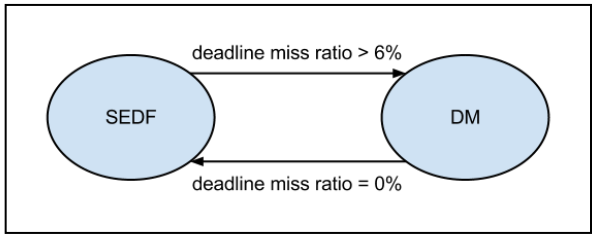


圖 4. SEDF_DM 的切換準則

## 3.3 SEDF_DM

將改良的 D_EDF 排程策略擴充於 Xen 提供的 SEDF 排程器。SEDF 排程器定義 PCPU 擁有一個可執行佇列 (runnable queue)，此佇列儲存可執行的 VCPU。可執行佇列是有序的，不論是插入或移除 VCPU，都是以絕對截止期作為比較運算子，對佇列作排序。當要執行一個 VCPU 時候，取得可執行佇列裡第一個 VCPU 即可。因此，我們以重新排序可執行佇列的方法，達到切換排程策略的目的。

基於 Xen 提供的 *sched_sedf.c* 做擴充，新增一個 *sched_sedfdm.c* 的排程器，並針對可執行佇列做合併排序 (merge sort)。當由 SEDF 切換至 DM 策略時，定義 *slice* (相對截止期)作為比較運算子，對可執行佇列排序。相對的，由 DM 切換至 SEDF 策略時，定義 *deadl_abs* (絕對截止期)作為比較算子，對執行佇列作排序。在 *update_queues()* 內以兩個整數變數

*miss_count* 和 *total*，紀錄截止期失誤及全部個數，最後計算截止期失誤是否大於 6%，比率關係式表示：

$$\text{Deadline Miss Ratio} = \frac{\text{miss count}}{\text{total}} >= 6\%$$

```
FOR each VCPU in runnable queue
    variable total increase
    IF deadline miss THEN
        miss_count increase
    END IF
END FOR
```

圖 5. *update_queue()*內部分演算程序

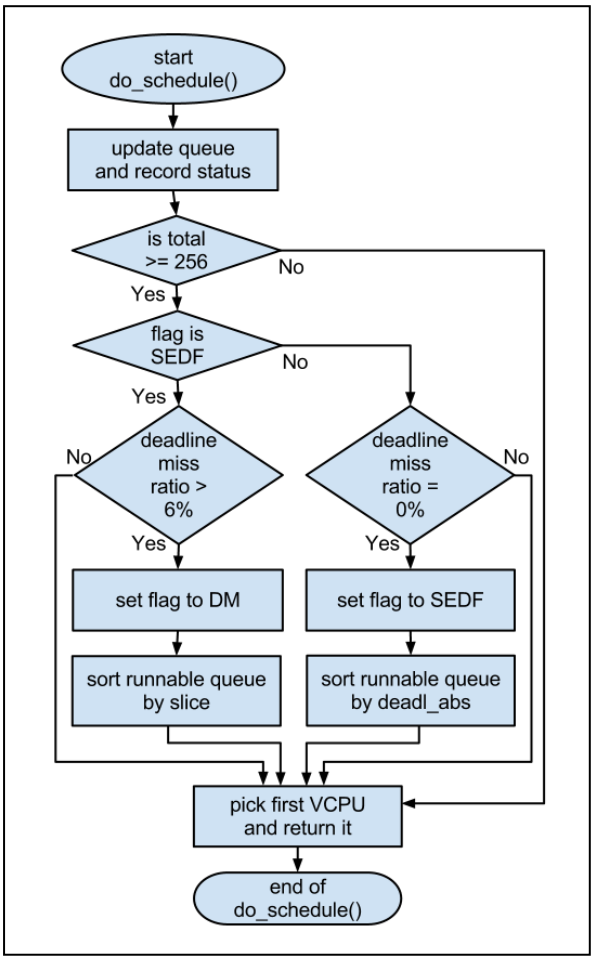


圖 6. 在 *do_schedule()*內定義切換準則

如圖 6 和圖 7，如果 deadline miss ratio 大於 6%，則切換至 DM。反之 deadline miss ratio 等於 0，則切換回 SEDF。

用來排序可執行佇列的合併排序是取自 Linux 核心。因為可執行佇列是由連結串列實作，所以使用合併排序是很合適的解決方案。

```
IF total >= 256 THEN
    IF current is EDF AND miss_count
> total right shift 4 bit THEN
        switch to DM
        sort runq by slice
    ELSE IF current is DM
      AND miss_count = 0 THEN
        switch back to EDF
        sort runq by deadl_abs
    END IF
END IF
```

圖 7. *do_schedule()*內部分演算程序

```
FOR each VCPU in runnable queue
    IF deadline miss THEN
        miss_count increase
        met_count reset
    ELSE IF deadline met THEN
        miss_count reset
        met_count increase
    END IF
END FOR
```

圖 8. *update_queue()*內部分演算程序

```
IF current is EDF THEN
    IF miss_count >= 2 THEN
        switch to DM
        reset miss_count
        sort runq by slice
    END IF
ELSE IF current is DM THEN
    IF met_count >= 10 THEN
        switch to EDF
        reset met_count
        sort runq by deadl_abs
    END IF
END IF
```

圖 9. *do_schedule()*內部分演算程序

### 3.4 實作改良的 D_EDF 排程演算法

基於 Xen 提供的 *sched_sedf.c* 實作 D_EDF 排程，新增兩個變數 *miss_count* 和 *met_count*，分別在 *update_queues()*內記錄截止期失誤和截止期達成的個數，並在 *do_schedule()*內進行門

檻判斷。如圖 8 和圖 9，分別為 *update_queues()* 和 *do_schedule()*的部份演算程序。

## 4. 實驗與結果

本章節將先介紹實驗環境以及使用的測試工具。首先我們先用 NPB 測試 6 個半虛擬化 domain-U 的效能。第二，用 Cyclictest 評估多個即時的客端虛擬機的平均核心延遲，在 CPU 負載為 100%環境下的核心延遲。

### 4.1 實驗環境

硬體規格為處理器為 Intel® Core™ i7-920 2.66GHz (Turbo 2.93GHz)並關閉超執行緒、主機板為 MSI X58 Pro-E，記憶體為 6 個 2 GB DDR3 SDRAM，共 12GB，以及硬碟 WD5000AALS 500GB。軟體規格為 Kernel 3.2.0-38、Ubuntu 12.04.2 LTS Server 64-bit 及 Xen 4.2.1。

Domain-0 的設置為一個 VCPU 且 pin 至 PCPU 1 以及 1GB 記憶體和 80GB 硬碟空間。domain-U 皆為半虛擬化虛擬機，設置一個 VCPU，不做 pin 的設定，保持在初始的 PCPU 0，分配 1GB 記憶體和 100GB 硬碟空間。新建 6 個 domain-U，分別依序命名為 Domain-1、Domain-2...、和 Domain-6，*period* 為 100ms，*slice* 分別設定為 50、54、58、62、66 和 70，*extra* 皆設定為 0。

### 4.2 比較 SEDF、D_EDF 和 SEDF_DM

我們使用 Numerical Aerodynamic Simulation Parallel Benchmarks (NAS Parallel Benchmarks, NPB)和 Cyclictest，針對 SEDF、D_EDF 和 SEDF_DM 作評估。開啟多個 domain-U，並同時執行 ep.A 程式，ep 是典型的平行程式，A 為測試資料的大小類別。分別測試每一個排程器 10 回合，將所有 CPU Process Time 進行總平均。圖 10，x 軸為開啟虛擬機的數量，y 軸為執行 ep.A 所花費的平均時間(秒)。當一個虛擬機執行的時候，在未過載狀況下，D_EDF 與 SEDF_DM 都未切換至 DM，SEDF_DM 的平均執行的時間近似於 SEDF。因此，為了紀錄截止期失誤，增加的開銷很小。當開啟 6 台虛擬機，達到過載狀態，D_EDF 共來回切換排程策略 5 次，時間縮短為 SEDF 的 95%；SEDF_DM 共來回切換 1 次，時間縮短為 SEDF 的 89%。

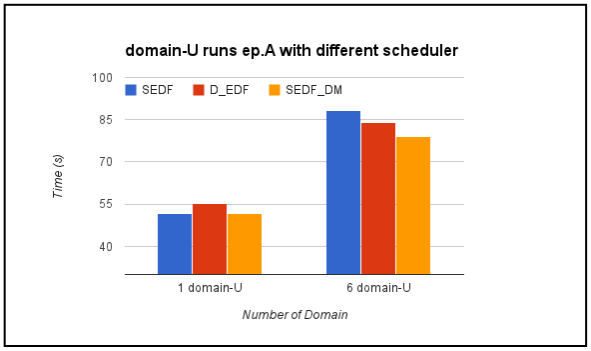


圖 10. 評估三種排程器的 CPU 時間

Cyclictest 是針對即時核心評估延遲的測試工具。為了讓 CPU 負載達到 100%，Domain-5 和 Domain-6 將執行 CPU 密集的工作。Domain-1~Domain-4 同時執行 Cyclictest，進行 150,000 次的測試。圖 11 是正規化後的結果，x 軸分別為不同的 CPU 負載，y 軸分別以 SEDF 為基準經過正規化後的平均延遲時間。

在 100%負載情況下 D_EDF 為 SEDF 的 91%，SEDF_DM 為 SEDF 的 84%。在沒有負載情況下，SEDF_DM 的核心延遲近似於 SEDF。

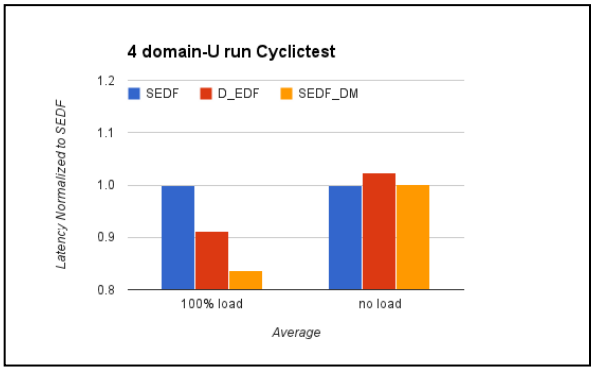


圖 11. 在不同負載情況下，評估三種排程器的核心延遲

圖 12 為一個半虛擬化虛擬機且有 100%負載的平均核心延遲。x 軸分為不同的 extra 設定，y 軸分別以 SEDF 為基準經過正規化的結果。結果顯示，當過載環境時，D_EDF 與 SEDF_DM 的核心延遲都明顯較低。若 extra 設為 0，D_EDF 為 SEDF 的 91%，SEDF_DM 為 SEDF 的 86%。若 extra 設為 1，D_EDF 為 SEDF 的 57%，SEDF_DM 為 SEDF 的 50%。

另外，測量了一個半虛擬化虛擬機且沒有負載的平均核心延遲。結果顯示 D_EDF、SEDF_DM 與 SEDF，三者的核心延遲皆非常近似。

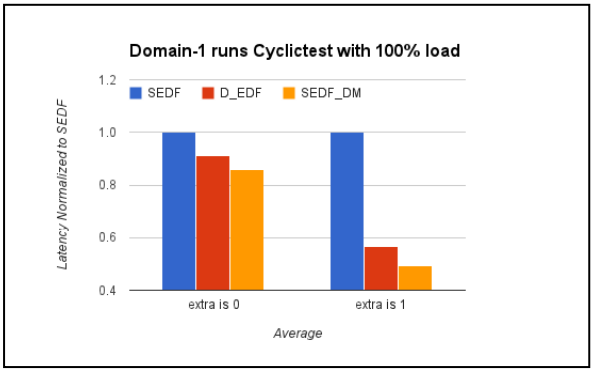


圖 12. 評估在負載 100%，並開啟一個 domain 的狀況下，三種排程器的核心延遲

## 5. 結論與未來工作

本文基於改進的 D_EDF，實作 SEDF_DM 排程策略，並在 Xen 平台上測試，並比較三種排程器。實驗結果表明，紀錄截止期失誤個數等狀態的開銷很微小，在未過載環境下進行核心延遲測試，實驗結果顯示 D_EDF 和 SEDF_DM 近似於 SEDF。在過載的環境下，實驗結果顯示 SEDF_DM 計算時間縮短為 SEDF 的 89%；平均核心延遲則縮短為 SEDF 的 86%。

隨著智慧型手機和工業自動化發展，嵌入式虛擬化變得日益重要，Xen 是一個輕量的原生虛擬機監視器，但是 SEDF 對多核心支援不佳。在預設的狀況下，所有 domain 的 VCPU 皆是使用一個核心，即 PCPU 0。若要使 SEDF 使用多個 PCPU，需使用 xm vcpu-pin 指令，將特定的 VCPU 與指定的 PCPU 作對應。即使 SEDF 可以在多核心下運作，但並沒有支援全域負載平衡及遷移 VCPU 的機制。

## 參考文獻


[1] Yoo, Seehwan, Miri Park, and Chuck Yoo. "A step to support real-time in virtual machine." ***2009 IEEE 6th Consumer Communications and Networking Conference (CCNC)***, Jan. 2009: 1-7.

[2] Ye, Kejiang, Xiaohong Jiang, Siding Chen, Dawei Huang, and Bei Wang, "Analyzing and modeling the performance in xen-based virtual cluster environment," ***2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)***, Sep. 2010: 273-280.

[3] Zhang, Xinjie, and Dongsheng Yin, "Real-time improvement of VCPU scheduling algorithm on Xen." ***2011 International Conference on Computer Science and Service System (CSSS)***, Jun. 2011: 1506-1509.

[4] Yu, Peijie, Mingyuan Xia, Qian Lin, Min Zhu, Shang Gao, Zhengwei Qi, Kai Chen, and Haibing Guan, "Real-time Enhancement for Xen hypervisor." ***2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC)***, Dec. 2010: 23-30.

[5] Kim, Hwanju, Hyeontaek Lim, Jinkyu Jeong, Heeseung Jo, and Joowon Lee, "Task-aware virtual machine scheduling for I/O performance." ***Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments***, Mar. 2009: 101-110.

[6] Tseng, Chia-Ying, Sheng-Chih Lin, Lee-Chung Chen, and Hui-Kuang Chung, "The Performance Improvement and Evaluation of an Enhanced CPU Scheduler in Virtualized Environment," ***Proceedings of the 2011 International Conference on e-CASE & e-Tech***, Jan. 2011: 3107-3123.

[7] Masrur, Alejandro, Sebastian Drössler, Thomas Pfeuffer and Samarjit Chakraborty. "VM-based real-time services for automotive control applications." ***2010 IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)***, Aug. 2010: 218-223.

[8] Zheng, Zhiyun, Ying Zhang, Zhenfei Wang, Xingjin Zhang, and Liping Lu, "The multi-processor load balance scheduler based on XEN." ***2012 International Conference on Systems and Informatics (ICSAI)***, May. 2012: 905-909.

[9] Xi, Sisu, Justin Wilson, Chenyang Lu, and Christopher Gill, "Rt-xen: Towards real-time hypervisor scheduling in xen," ***2011 Proceedings of the International Conference on Embedded Software (EMSOFT)***, Oct. 2011: 39-48.

[10] Thakor, Devendra, and Apurva Shah. "D_EDF: An efficient scheduling algorithm for real-time multiprocessor system," ***2011 World Congress on Information and Communication Technologies (WICT)***, Dec. 2011: 1044-1049.