# 建置虛擬化佇列管理架構於雲端運算

林傳筆 江茂綸 張運晟
朝陽科技大學 朝陽科技大學 朝陽科技大學
資訊與通訊系 資訊與通訊系 資訊與通訊系
助理教授 助理教授 研究生
cblin@cyut.edu.tw mlchiang@cyut.edu.tw s9830618@cyut.edu.tw

## 摘要

由於網路服務使用量快速增長，服務供應商所能夠提供的運算資源越來越不足，而為了維持服務的品質及提供更完整的網路應用服務，高擴展性及強大運算能力的雲端環境將是目前趨勢；然而，在雲端環境下達到有效率的資源分配，以解決任務分配不均而造成壅塞及遺失的問題。因此，本文結合 Combined Input Crosspoint Buffer ($CICB$)、Threshold 及 Flow control，提出一個混合式 Cross Queue Capacity Virtualization Management ($CCQVM$) 的雲端基礎架構，以降低佇列等待時間及任務遺失率。

關鍵詞：雲端運算，任務排程，虛擬化管理，佇列

## Abstract

As the uses of network service are growing rapidly, the serverice providers of computing resource have become inadequate increasingly. To keep Quality of Service and more complete network applications, the scalable and powerful computing capability of cloud environment will be a research trend. However, the cloud environment causes the problems of task loss and network congestion to achieve an efficient allocation of resources. Therefore, we in this paper propose a cloud infrastructure of hybrid Cross Queue Capacity Management ($CCQVM$) integrating Combined Input Crosspoint Buffer ($CICB$), Threshold, and Flow control mechanisms to reduce the queue wait time and task loss rate.

Keywords：Cloud Computing, Task Schedule, Virtualization Management, Queue

## 1. 介紹

個人電腦的普及和網路通訊發展的進步，網路應用服務已經成為人們生活中不可或缺的一部份。由於網路服務使用者的增加，服務供應商為了讓使用者有好的服務資源，使用垂直和水平擴展避免服務量缺乏的狀況，但擴展的速度還是遠不及使用者人數的成長，所以俱高擴展性 [1] 及強大運算能力的雲端運算 [10],[11],[12],[13] 服務將成為主要的趨勢。

雲端運算根據服務類型可以分成基礎架構式服務 (IaaS)[2]、平台式服務 (PaaS)[3] 和軟體式服務 (SaaS)[4]。基礎架構式服務是一個虛擬化的運算基礎設施，使用者只需透過設定介面就可以租賃運算資源，並在其租賃的基礎架構服務上建置系統，像最有名的例子是 Amazon EC2[5]。而平台式服務可以讓使用者不用建置系統平台或 OS，就可使用供應商所提供的 API 來開發相關的應用程式，例如 Microsoft Azure[6] 的用戶不需要在機房架設 Server，就能直接在 Azure 上設計相關的程式如.NET、SQL；另一個常見的例子是 Google App Engine[7]，能夠提供給程式設計師 JAVA、Python 語言的開發環境，開發後直接在平台上佈署開發的程式，完全不用管理後端伺服器的運作。此外，不同於前面兩種服務類型，軟體式服務主要是提供一個已開發的雲端運算軟體，專門讓使用者透過軟體來得到雲端的相關服務，例如 Google earth[8]，讓使用者本機不用安裝全世界的地圖檔，只要當有需要相關地圖時，再向 Server 發出下載地圖檔的需求即可。

雲端運算 [14] 是一個巨型的分散式系統，因此需要一個虛擬化的集中管理 [15] 技術來管理數量龐大的資源；而使用這技術的好處是可讓使用者和網路管理者透過單一的介面來使用和管理資源，而在眾多的虛擬化管理技術中以 Open Nebula[16] 和 Hadoop[17] 最為熟知。Hadoop 是 Dong Cutting 從 Google 的 MapReduce [9] 核心概念發展而成，他提供了管理分散式運算的環境，使用者可以用效能較低的運算資源，透過 MapReduce 的架構把運算分散在數個運算資源中，以提升效能。因此，Hadoop 的使用者不用購買昂貴的伺服器就能夠提供大量的運算資源。Open Nebula 是一個開放原始碼的雲端運算系統。它可簡易的在本地端利用 Xen、Kernel-based Virtual Machine(KVM) 或 VMware 來建置雲端運算基礎架構，當本地端運算資源不足時，能透過 Amazon EC2 來支援本地端擴展運算資源。

本篇將會提出一個混合式的雲端架構 Cloud Cross Queue Virtualization Management($CCQVM$)，

主要在 Combined Input Crosspoint Buffer($CICB$) 架構下以佇列和排程演算法去暫存和分配任務。在本文的第二節會探討相關學者如何利用佇列來管理任務的排程，而第三節將會介紹所提出的混合式雲端架構，第四節為模擬效能分析，最後是本篇的結論。

## 2. 文獻探討

為了使雲端服務的使用者能夠獲得更好的服務品質，已經有相關學者提出利用佇列來分配任務。

Luqun Li [19] 分析雲端運算不同服務品質的需求，提出一個 no-preemptive priority M/G/1 queuing model 去分配使用者的任務，藉此讓雲端的提供商達到最好的資源使用率，同時也保證了使用者的服務品質。此外，Dawei Sun [20] 提出在 M/M/1 的佇列演算法下，建立 M × N 的資源分配模組，進而提出一個 NECDA 演算法來提升雲端使用者的任務執行成功率。而 Murata [21] 則是使用向量超級電腦去紀錄及分析歷史執行時間，進而預測進入佇列的工作時間，讓佇列自動分配任務到適合的雲端資源環境，使工作能夠大大減少等待分配時間。

由於以上的研究在伺服器分群數量變多時，將會提高演算法實行的複雜度，所以希望能夠提出一個擁有簡單的演算法和系統的雲端虛擬化管理架構，在本研究中我們利用交換機中擁有高效能的 CICB[22], [23] 的佇列架構，讓雲端運算的任務能夠達到較低的遺失率和 queue 等待時間。

## 3.Cloud Cross Queue Virtualization Management

由於雲端運算中的虛擬管理是非常重要的技術，此技術通常介於使用者和實體資源之間，其本文所提出的雲端虛擬化管理基礎架構如圖 1. 所示；在此架構中能夠分成 Core、Command-Line Interface (CLI)、Access Drivers 和 Capacity Manager 四個元件，其詳細說明如下：

*1) Core:* 其主要功能是針對雲端運算進行管理，又分為 Network、Monitor 和 Virtualization 三個子元件，其中 Network 子元件負責建立和管理虛擬資源的網路，此元件會透過 VPN 將 Virtual Machine($VM$) 相互連結；Monitor 子元件將會透過 Monitor Agent 收集資源效能來監控雲端資源的使用率，並把有用的資訊提供給 Capacity Manager 使用；最後，Virtualization 會統一的去控制下層 Virtual Machine($VM$) 的軟體。

*2) Command-Line Interface*($CLI$)*:* 提供操作介面讓網路管理員可以建置雲端服務或相關設定。

*3) Access Drivers:* 主要安裝在實體資源中，透過 XEN 和 Kernel Virtual Machine($KVM$) 的虛擬機器軟體來虛擬化實體資源。此外，Access Drivers 也會提供插件讓上層 Core 中的 Virtualization 來統一管理。

*4) Cross Queue Capacity Management:* 負責接收使用者任務和分配任務排程。而其分配及排程的設計，將會影響整體運算環境的效能，因此本架構的 Capacity Manager 的運作流程將詳細描述如下。

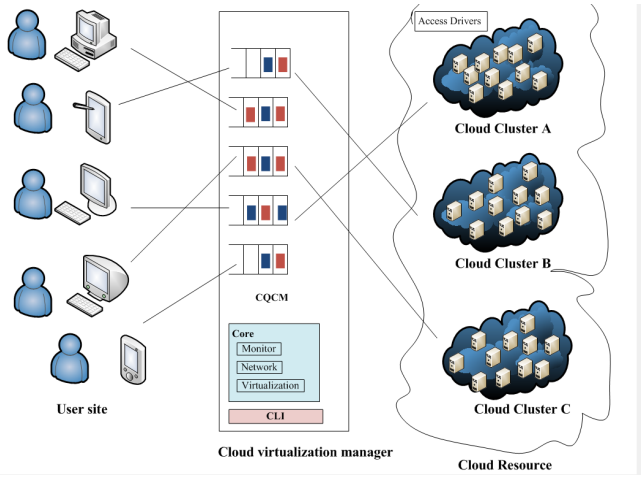


圖 1. 雲端虛擬化管理基礎架構

在本架構 CCQVM 中，我們提出 CQCM，達成使用者任務的分配及排程，其主要由 $n$ 個 Cloud Group Buffer($CGB_i$) $0 \leq i \leq n-1$ 和 Cloud Cluster ($CC_j$) $0 \leq j \leq n-1$ 所組成，其架構如圖 2. 所示。而在每一個 $CGB$ 都有 $n$ 個 Group Queue ($GQ_{ij}$) 用來暫存到從 Client 送過來的任務要求。隨後，我們在 $CQCM$ 中設計一個 Cloud Cross Buffer ($CXB_{ij}$) 來暫存到 $CC_j$ 的任務。

Cloud Cliet Group($CCG_i$) 是需要服務的使用者所聚集成的分群，當 $CCG_i$ 需要使用服務時，$CCG_i$ 會送出任務至 $CGB_i$，隨後 $CQCM$ 會依據任務的目的地把任務送到 $CGB_i$ 中的 $GQ_{ij}$ 進行暫存，因為 $GQ_{ij}$ 中的任務會產生互相爭奪傳送 [18] 的現象，因此需演算法來進行任務排程，而本篇使用了三個演算法分別是 Threshold Worst Fit Match ($TWFM$) 1、Threshold Round Robin Match ($TRRM$) 和 Threshold Best Fit Match ($TBFM$))。此外，本排程演算法利用 Threshold 和 Flow control 機制來降低 Task Loss Rate，進行提升整體系統效能。

而為了能夠讓任務可以在進入 $CC_j$ 時獲得好的服務品質，本架構使用 $threshold$ 機制來避免任務被分配至能力不足的 $CC_j$ 中。首先 CQCM 會進行初始化來計算 $CC_j$ 的 $CCt_j$；而服務進行時，Monitor 元件會透過 Monitor Agent 定期的向 $CC_j$ 收集 CPU 的剩餘使用量、Memory 的剩餘使用量和 Storage 的剩餘儲存空間，並且使用一個 value function ($VF_j$)

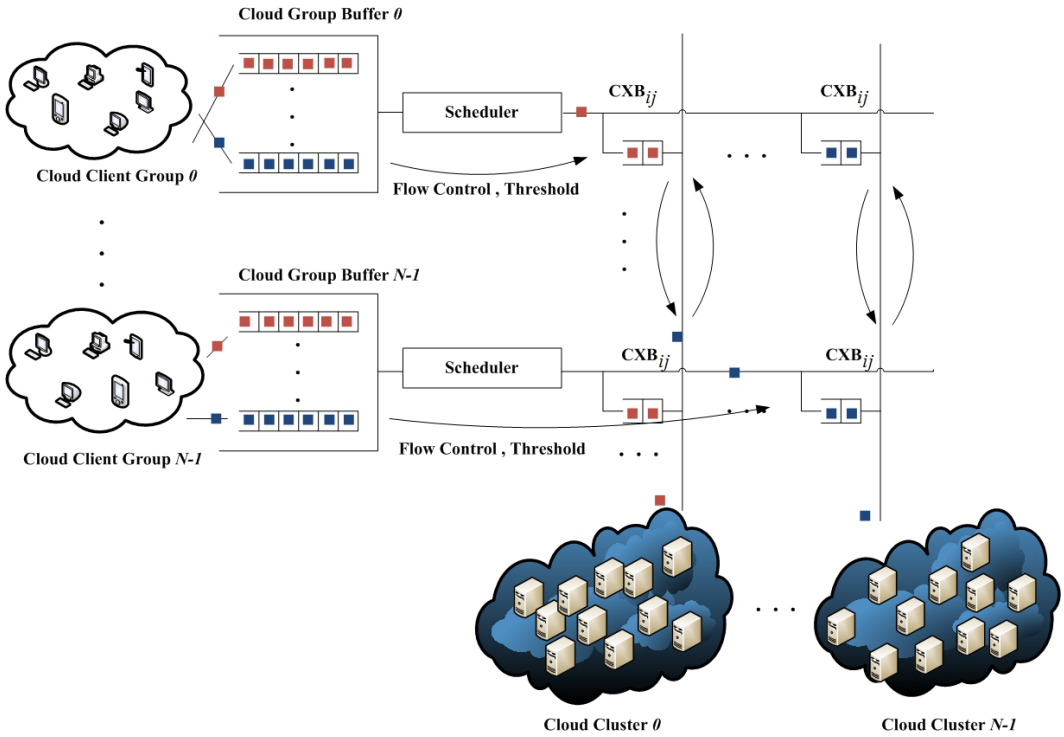


圖 2. Cross Queue Capacity Management ($CQCM$)

[24] 來計算出 $CC_j$ 的最大服務能力 $VF_{jmax}$ 。隨後 Monitor 元件會把 $VF_j$ 傳給 $CQCM$ ，而 $CQCM$ 會根據 $VF_j$ 和 $CCt_j$ 去進行判斷，如果 $CCt_j > VF_j$ ，$GQ_{ij}$ 的任務才有資格傳遞到 $CXB$ 中。其 $VF_j$ 和計算初始化 $CCt_j$ 的方法如方程式 (1) 和 (2) 所示。

$$VF_j = \sum_{i}^{n} w_x f(x_{nj}) = w_1 f(x_{1j}) + \cdots + w_n f(x_{nj});$$

$$\sum_{i}^{n} w_i = 1, 0 \le j \le N-1, 0 \le f(x_{ij}) \le N-1 \quad (1)$$

$x_{ij}$ ： $CC_j$ 決策變數 $i$ 的值
$f(x_{ij})$ ： $CC_j$ 決策變數 $i$ 的函數值
$V_i$ ： 估測 $CC_j$ 的能力值
$w_i$ ： 每個變異值的權重

$$CCt_j = VF_{jmax} \times Threshold \quad (2)$$

$Threshold$ ： 門檻值的百分比
$CCt_j$ ： $CC_j$ 的門檻值
$VF_{jmax}$ ： $CC_j$ 最大服務能力

隨後本研究在 $CGB_i$ 和 $CXB_{ij}$ 之間加入 Flow control 機制，來降低 $CGB_i$ 和 $CXB_{ij}$ 之間的 Task Loss Rate ，讓 $CXB_{ij}$ 無法暫存任務時，會發送回饋訊息給 $CGB_i$ ，在 $CGB_i$ 收到回饋訊息後，相對應的 $GQ_{ij}$ 將不會傳送任務至 $CXB_{ij}$ 中。

隨後，符合系統設計的 Threshold 和 Flow control 機制的 $GQ_{ij}$ 將由本篇所提之三種任務排程機制把任務傳送至 $CXB_{ij}$ 中，而 $CXB_{ij}$ 與 $CC_j$ 之間的排程將以 Round Robin Match ($RRM$) 進行。其演算法 $TWFM$ 、$TRRM$ 和 $TBFM$ 的詳細流程如演算法一、二及三所示。

演算法 **1** Threshold Worst Fit Match

```
while GQ_ij do
    j = Min(VF_j);
    if GQ_ij ∩ Flowcontrol then
        if CCt_j > VF_j then
            CXB_ij ← GQ_ij;
        end if
    end if
end while
```

根據上面所提之 CCQVM 架構，其分析將於下一章節詳加介紹。

## 4. 模擬效能

本章節將會針對 $TWFM$ 、$TRRM$ 和 $TBFM$ 三種演算法進行效能分析；我們使用 C 語言來進行整體模擬實驗，且為了能運作於異質性的網路環境，

演算法 **2** Threshold Round Robin Match

**while** $GQ_{ij}$ **do**
  $j = Min(VF_j)$;
  **if** $GQ_{ij} \cap Flowcontrol$ **then**
    **if** $CCt_j > VF_j$ **then**
      $CXB_{ij} \leftarrow GQ_{ij}$;
    **end if**
  **end if**
**end while**

演算法 **3** Threshold Best Fit Match

**while** $GQ_{ij}$ **do**
  $j = Max(VF_j)$;
  **if** $GQ_{ij} \cap Flowcontrol$ **then**
    **if** $CCt_j > VF_j$ **then**
      $CXB_{ij} \leftarrow GQ_{ij}$;
    **end if**
  **end if**
**end while**

本實驗將運算能力分類為 8 個等級， 1 代表效能最差， 2 則代表和 1 能力差異為兩倍，其餘依此類推。隨後並進行檢測 Task Loss Rate 和 Queue Wait Time 兩種指標，Task Loss Rate 這裡是表示當任務離開 $CXB_{ij}$ 到 $CC_j$ 之間的遺失，Queue Wait Time 是 Task 在 $CGB_i$ 和 $CXB_{ij}$ 中所暫存花費的時間。隨後 $TWFM$ 、 $TRRM$ 和 $TBFM$ 說明如下所示。

### *A. Threshold Worst Fit Match* ($TWFM$)

從圖 3. 中可以看出套用 $TWFM$ 演算法之後的 Queue Wait Time 會隨著 $Threshold$ 而跟著提高；其主要是因為隨著 $Threshold$ 的提高，任務進入 $CC_j$ 的條件也變的嚴苛。因此，當 $CC_i$ 沒有能力可以提供服務時，任務也會在 $GQ_{ij}$ 中進行暫存，所以 Queue Wait Time 也跟著增加。而在圖 4. 能夠看出 $TWFM$ 的 $Threshold$ 必須降至到 40% 才能夠降低 Task Loss Rate 到趨近於 0。

### *B. Threshold Round Robin Match* ($TRRM$)

從圖 5. 能看到 $TRRM$ 不論 $Threshold$ 設定的高低都不會影響 Queue wait Time，這是因為在 $GQ_{ij}$ 是以輪循的方式傳送任務至 $CXB_{ij}$ 中。雖然 $Threshold$ 的設定不影響 $TRRM$ 的 Queue Wait Time，但圖 6. 中顯示門檻值至 60% 時，能夠把 Task Loss Rate 降低至趨近於 0。

### *C. Threshold Best Fit Match* ($TBFM$)

而 $TBFM$ 演算法中，效能較佳的 $CC_j$ 將優先被選擇來傳送任務，所以在圖 7. 中顯示擁有最

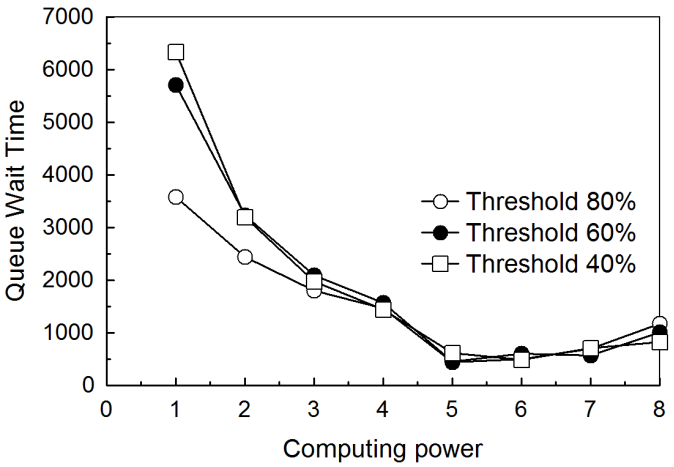


圖 3. QueueWait Time of TWFM

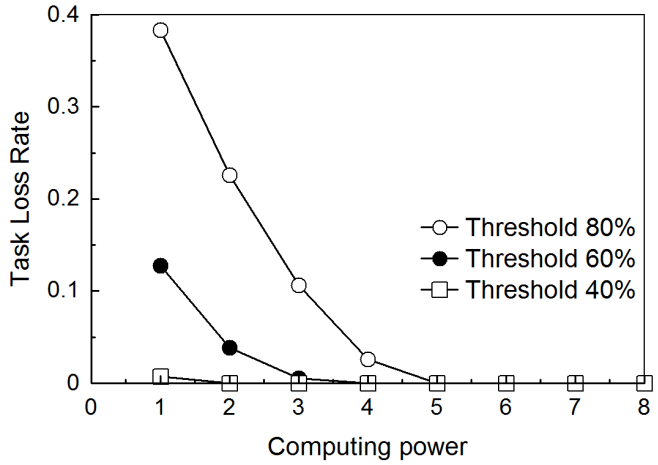


圖 4. Task Loss Rate of TWFM

佳能力 $CC_j$ 的 Queue Wait Time 是最少的。而在提高 $TBFM$ 的 $Threshold$ 設定後，較差的 $CC_i$ 的 Queue Wait Time 會稍微降低，因為擁有最佳能力的 $CC_j$ 在 $Threshold$ 值設定的更為嚴苛時，會把傳送任務的機會讓給較差的 $CC_j$ ，所以才會造成較差的 $CC_i$ Queue Wait Time 會稍微降低。此外，圖 8. 顯示 $TBFM$ 在 $Threshold$ 提高到 60% 時，就能夠大大的降低 Task Loss Rate，所以在 $CCQVM$ 中能夠使用 $TBFM$ 得到不錯的服務品質。

### *D.* 綜合分析

圖. 9 和圖. 10 中將比較 $TWFM$ 、 $TRRM$ 及 $TBFM$ 在 $Threshold$ 設定為 80% 時的 Queue Wait Time 和 Task Loss Rate 的情況。當 $Threshold$ 在 80% 時， $TWFM$ 擁有一個最好的 Queue Wait Time，但 $TWFM$ 的 Task Loss Rate 是最高的。此外， $TRRM$ 的 Queue Wait Time 和 Task Loss Rate 與 $TWFM$ 及 $TBFM$ 相比之下是最為平均的。然

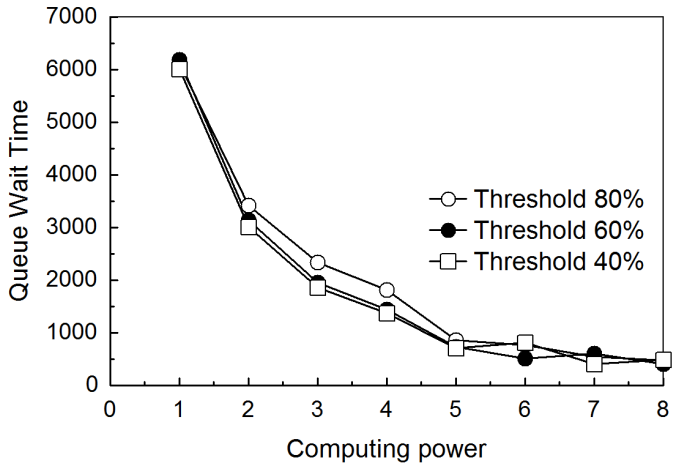


圖 5. Queue Wait Time of TRRM

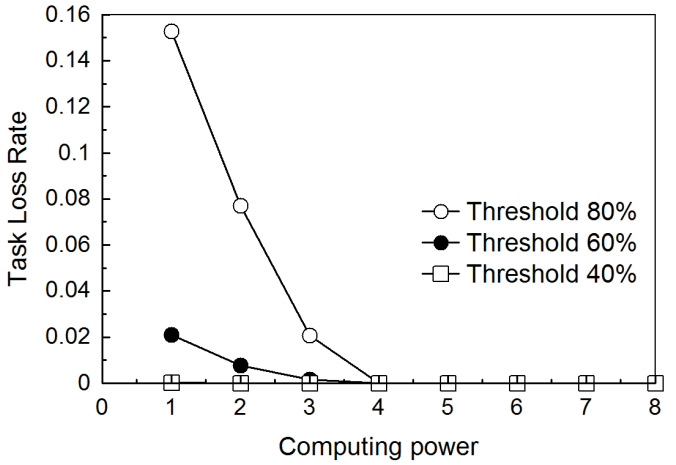


圖 6. Task Loss Rate of TRRM

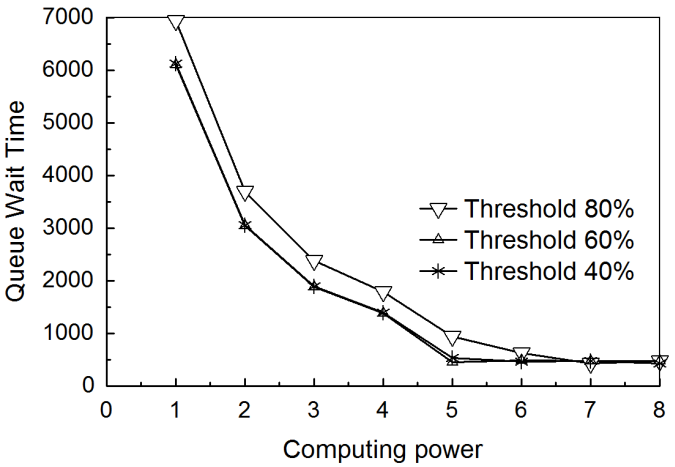


圖 7. QueueWait Time of TBFM

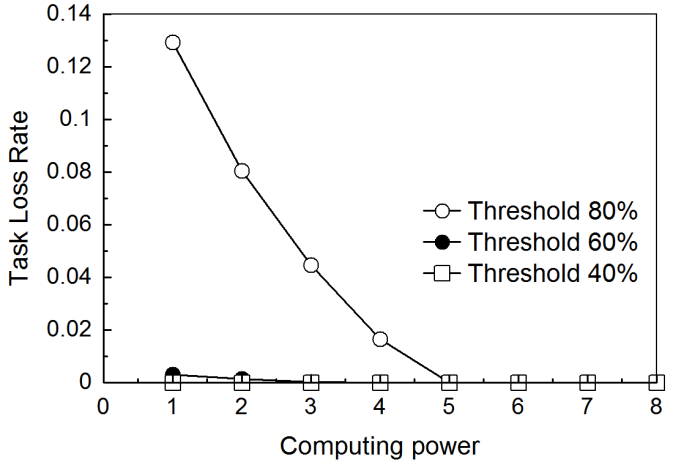


圖 8. Task Loss Rate of TBFM

而，$TBFM$ 雖然擁有最佳的 Task Loss Rate 但是卻會使最差的 $CC_i$ 有高的 Queue Wait Time。

因此，我們提出的三種演算法，根據模擬校能分析的結果，可以應用於不同的雲端服務環境中。雖然 $TWFM$ 擁有較高的 Task Loss Rate，但也擁有較低的 Queue Wait Time，所以適合在提供檔案傳輸服務中使用，透過檔案存取服務的流量控制機制，能夠允許 Task Loss Rate 的情況發生，而擁有較低 Queue Wait Time 的 $TWFM$ 演算法，將能提升檔案的存取時間。而 $TBFM$ 演算法適合在需要大量運算的服務環境中，主要是因為 $TBFM$ 演算法和前兩者演算法相比，會選擇最佳的 $CC_j$ 來分配任務，所以能夠保證用戶得到較好的運算能力。$TRRM$ 擁有最好的 Task Loss Rate 和 Queue Wait Time，而且使用公平的輪循機制，$TRRM$ 適合在多種而且複雜的網路服務環境中使用。

## 5. 結論及未來工作

本文提出一個 $CCQVM$ 的混合式的雲端架構來管理雲端運算，達到降低 Task Loss Rate 和 Queue Wait Time 的效果。然而工作的分配及排程將是整體工作效能的瓶頸，因此，本架構針對此問題提出 $CQCM$ 來管理其工作排程，而其中所包含的演算法，如 $TWFM$ 、$TRRM$ 及 $TBFM$ 各有其特色及優勢，如擁有較低 Queue Wait Time 的 $TWFM$ 適合在提供檔案傳輸服務中使用；而優先選擇最佳的 $CC_j$ 的 $TBFM$，適合在需要大量運算的服務中使用；擁有最好的 Task Loss Rate 和 Queue Wait Time 的 $TRRM$，適合在多種而且複雜的網路服務環境中使用。

由於本篇提出的雲端管理架構能夠擁有較好的 Task Loss Rate 和 Queue Wait Time，所以在未來工作將進一步探討如何效的利用排程達到雲端的工作負載平衡。

## 6. 誌謝

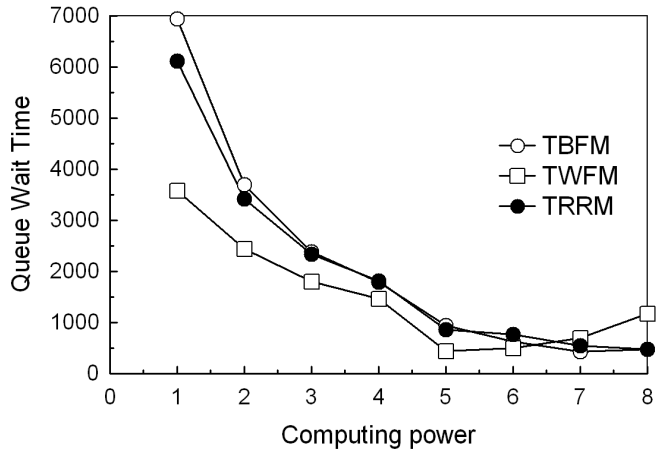


圖 9. QueueWait Time of TWFM 、TRRM 、TBFM

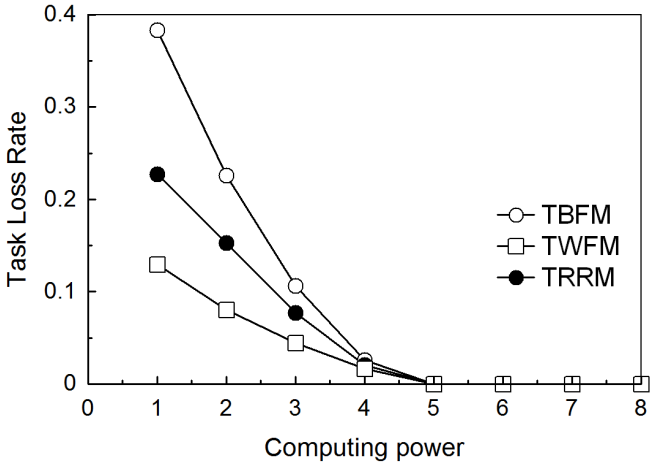


圖 10. Task Loss Rate of TWFM 、TRRM 、TBFM

這篇論文是國科會計畫 (NSC99-2221-324-041-MY3) 研究成果的一部份，我們在此感謝國科會經費支持這個計畫的研究。

## 參考文獻

[1] N. Yigitbasi, A. Iosup, D. Epema and, S. Ostermann, "C-Meter: A Framework for Performance Analysis of Computing Clouds," *CCGRID '09. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp.472-477, 18-21 May 2009.

[2] R. Prodan and, S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers,"*10th IEEE/ACM International Conference on Grid Computing*, pp.17-25, 13-15 Oct. 2009.

[3] W. Tian, S. Su and, G. Lu, "A Framework for Implementing and Managing Platform as a Service in a Virtual Cloud Computing Lab," *Second International Workshop on Education Technology and, Computer Science (ETCS)*, pp.273-276, 6-7 March 2010.

[4] E.R. Olsen, "Transitioning to Software as a Service: Realigning Software Engineering Practices with the New Business Model,"*SOLI '06. IEEE International Conference on Service Operations and, Logistics, and Informatics* , vol., no., pp.266-271, 21-23 June 2006.

[5] S. Toyoshima and, S. Yamaguchi; Oguchi, M.; , "Storage Access Optimization with Virtual Machine Migration and Basic Performance Analysis of Amazon EC2," *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp.905-910, 20-23 April 2010.

[6] Microsoft (2009) "Windows Azure Platform" .http:// www.microsoft.com/ azure/ default.mspx [Last accessed 23 Sep. 2009].

[7] A. Bedra, "Getting Started with Google App Engine and Clojure," *Internet Computing, IEEE*, vol.14, no.4, pp.85-88, July-Aug. 2010.

[8] M.T. Jones, "Google's Geospatial Organizing Principle," *Computer Graphics and Applications, IEEE*, vol.27, no.4, pp.8-13, July-Aug. 2007.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*,vol. 51 Issue 1, January 2008.

[10] A. Weiss, "Computing in the clouds," *netWorker*, vol. 11, no. 4, pp. 16 – 25, 2007.

[11] H. Brian, "Cloud computing", *Communications of the ACM*, v.51 n.7, July 2008.

[12] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," *HPCC '08. 10th IEEE International Conference on High Performance Computing and Communications*, pp. 5 – 13., Sep. 2008.

[13] D. Kondo, B. Javadi, P. Malecot, F. Cappello,and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids,"*IPDPS 2009. IEEE International Symposium on Parallel & Distributed Processing*, May 2009.

[14] B.P. Rimal, C. Eunmi and, I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems," *NCM '09. Fifth International Joint Conference*, pp.44-51, 25-27 Aug. 2009.

[15] M. Bolte, M. Sievers, G. Birkenheuer , O. Niehorster.and, A. Brinkmann, "Non-intrusive virtualization management using libvirt," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pp.574-579, 8-12 March 2010.

[16] H. Zhong, K. Tao and, X. Zhang; , "An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems," *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*, pp.124-129, 16-18 July 2010.

[17] J. Shafer, S. Rixner and, A.L. Cox; , "The Hadoop distributed filesystem: Balancing portability and performance," *IEEE International Symposium on Performance Analysis of Systems Software (ISPASS)*, pp.122-133, 28-30 March 2010.

[18] J. S.-C. Chen and T. E. Stern,"Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE J. Select. Areas Commun*, vol. 9, pp. 439 – 449, Apr. 1991.

[19] L. Li; , "An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers," *MUE '09. Third International Conference on Multimedia and Ubiquitous Engineering*, pp.295-299, 4-6 June 2009.

[20] L D. Sun, G. Chang, C. Wang, Y. Xiong ,and X. Wang, "Efficient Nash equilibrium based cloud resource allocation by using a continuous double auction," *International Conference on Computer Design and Applications (ICCDA)*, pp.V1-94, 25-27 June 2010.

[21] Y. Murata, R. Egawa, M. Higashida and, H. Kobayashi, "A History-Based Job Scheduling Mechanism for the Vector Computing Cloud," *IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, pp.125-128, 19-23 July 2010.

[22] R. Rojas-Cessa, E. Oki ,and H. Jonathan Chao, "CIXOB-k: Combined Input-Crosspoint-Output Buffered Switch," *Proceedings of IEEE Globecom 2001*, Vol. 4, pp. 2654-2660, Dallas, TX, November 2001.

[23] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined Input-One-Cell-Crosspoint Buffered Switch," *Proceedings of IEEE HPSR 2001*, pp. 324-329, May 2001.

[24] K. Q. Yana, S. S. Wanga, S. C. Wang and, C. P. Changa, "Towards a hybrid load balancing policy in grid computing system," *Expert Systems with Applications: An International Journa*, Vol. 36, Issue 10,pp. 12054-12064 , Dec. 2009.