利用最少工作量與連線數演算法作動態負載平衡之 Linux 叢集伺服器

A dynamic load balancing linux cluster server with least load and connection algorithm

何裕琨　　吳美宜　　許順興

國立成功大學電機工程學系

摘要

近年來由於網路的普及，一些熱門的網站，每秒至少都有上千人同時連線進入網站。為了應付如此高負載的網路環境，採用叢集伺服器(cluster server)是一可行之辦法。

一般常見的負載平衡叢集伺服器系統有使用領域名稱服務(DNS)、使用網路位址轉換(NAT)及使用 IPIP 通道(tunnel)封裝等三種技術，因 IPIP 通道封裝，有著較好的效能，因此本論文中將利用 Linux 中的 IPIP 通道機制來實現此一負載平衡系統。在負載平衡演算法部分有輪詢演算法(round robin algorithm)、加權式輪詢演算法(weighted round robin algorithm)、最少連線演算法(least-connection algorithm)及加權式最少連線演算法(weighted least connection)等多種，在本論文中將比較其優劣，並提出一新的負載平衡演算法：最少工作量與連線數演算法(Dynamic least Load and Connection,DLLC)，此演算法是依據真實伺服器的負載量(load)及其目前連線數來作為分配的依據，藉此達到負載平衡的目的。

本論文所建構之負載平衡叢集伺服器曾利用效能測試模擬軟體(HP 公司之 httperf)，對本論文所提出之 DLLC 負載平衡演算法與其他演算法做比較，證明在伺服器透通量、伺服器回應數及伺服器回應時間方面，均有較好之效能，由此可知此叢集伺服器具有高擴充性(high scalable)、高可用性(high availability)的特性。

關鍵字：load balancing server , load balancing algorithm , cluster server , Linux , IP-tunnel , dynamic least load and connection algorithm

1.簡介

目前常見的負載平衡叢集伺服器系統方面，對於封包的處理有使用領域名稱服務(Domain Name Service,DNS) [16]、使用位址轉換技術(Network Address Translation,NAT)[4]及使用 IPIP 通道(IP in IP tunneling)封裝技術[18]等三種。由於在一般情況下使用者的請求(request)資料常遠小於伺服器回應的資料，例如在超文件傳輸協定(hypertext transmission protocol,HTTP)中：使用者可能送出 GET test.jpg 等幾個 Bytes 的請求，而伺服器卻是要回應幾百 Kbytes 的資料，所以利用 IPIP 通道封裝之技術，將請求(request)及回應(response)分開處理可以增加整系統之效能。

在封包分配方面一般常見的演算法有輪詢演算法(round robin algorithm)[18]、加權式輪詢演算法(weighted round robin algorithm)[18]、最少連線演算法(least-connection algorithm)[18]、加權式最少連線演算法(weighted least connection)[18]等四種。輪詢演算法是依序將來自使用者的請求分配到每一伺服器上，最少連線演算法則是把使用者的請求分給目前最少連線數的伺服器，而加權式輪詢演算法及加權式最少連線演算法改良上述二種演算法，增加了權重值(weighting)，使較好效能的伺服器會被分配到較多的請求。

在本論文中將針對上述四種演算法作分析比較，並且提出一新的負載平衡演算法：動態最少工作量及連線數演算法(dynamic least

load and connection algorithm)。此動態最少工作量及連線數演算法是依據伺服器動態的工作量(load)及其目前連線數，作為分配的依據並且動態調整參數，期能更適切符合伺服器實際狀況。

2.負載平衡系統的分類

負載平衡系統中，較為簡單的則是利用如超文件傳輸協定(hypertext transmission protocol：HTTP)中有語法提供所謂轉向(redirection)的功能[17]，及瀏覽器本身的轉向(redirection)功能，都能讓伺服端可以把客戶端的請求轉到另一伺服器上。而較複雜效果較好的則使用領域名稱服務(DNS)系統[16]、使用位址轉換系統(NAT)[4]、使用 IPIP 通道封裝系統[18]等。

2.1 使用領域名稱服務(DNS)之負載平衡系統

領域名稱服務是以輪詢方式來支援負載平衡，基本上是建立在 BIND 軟體上(一套架設 DNS 的軟體， Unix 系統上之自由軟體)。是利用領域名稱服務以輪詢方式將來自使用者的請求(request)，依序分配到不同伺服器上來處理，而達到負載平衡之目的。而每台伺服器有則著相同的內容。其基本原理是：透過領域名稱服務輪詢的方式，把領域名稱依序對應到不同的 IP 位址，藉此將負載分到每台伺服器上，從而提高整個系統的執行效能。

其基本架構，如圖 1.所示。

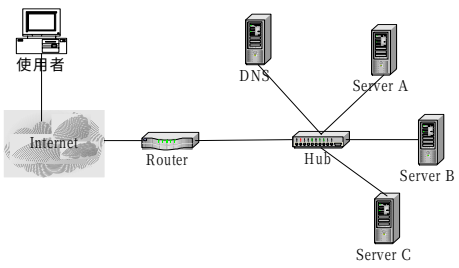

圖 1. DNS 輪詢系統

感覺上領域名稱服務輪詢系統好像是一種很好的負載平衡架構，實則不然，領域名稱服務輪詢系統有著下列幾個缺點：第一、領域名稱是種樹狀結構，各級的領域名稱伺服器快取(cache)中，需要紀錄著已有的領域名稱到 IP 的對應，如此一來，將會影響到負載平衡的效果；第二、系統的可靠性較差，一旦某一伺服器發生問題時，即使立即更改領域名稱伺服器設定，也要等一段時間（更新時間），才能發揮作用，因此在這段時間裡，使用者將有可能被對應到故障的伺服器上。所以，領域名稱服務輪詢系統，充其量只能達到查詢量的負載分配，並沒有真正做到負載平衡，而當其中有一個伺服器當機或網路斷線無法服務時，領域名稱伺服器並不知道，而仍然將此 IP 回應給使用者，造成使用者無法獲的正常的服務。更由於領域名稱伺服器只是用輪詢的方法，輪流把不同的伺服器 IP 回應給使用者，所以並未考慮到網路流量及伺服器實際負載的大小。

2.2 使用網路位址轉換(NAT)之負載平衡系統

網路位址轉換系統，主要提供內部位址與外部位址轉換的工作。網路位址轉換系統除了能讓私有位址(private)連上網際網路外，還有一個好處：可以用來作為分散網路負載平衡（load balancing）的技術。也就是說，可以將請求分別導向好幾台執行相同服務的伺服器上。例如 Cisco 的 local director[4](如圖 2.7 所示)。當使用者的請求封包來時，它會把封包的目的位址更改為網路(LAN)其中一台伺服器，而所回應的訊息，其封包的來源位址與目的位址都會被更改，藉此以單一 IP 的方式，來達到負載平衡的目的的目的。

由於 NAT 系統針對每一次的處理請求封包，都要進行修改（來源 IP、目的 IP），所以當連結請求流量高時，使用以網路位址轉換之負載平衡伺服器，將成為網路流量的新瓶頸。

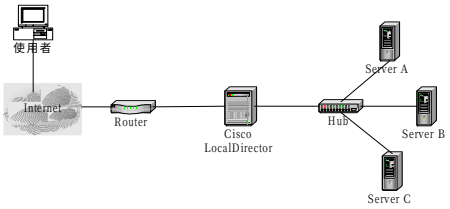

圖 2.Cisco localdirector

2.3 使用 IP 通道(tunnel)封裝之負載平衡系統

在利用以網路位址轉換(NAT)之負載平衡伺服器上，對於客戶端(client)的請求與伺服器的回應，都需要經過以網路位址轉換之負載平衡伺服器。它所造成的結果是：假如叢集伺服器中的節點(node)增多時；或是整個網路流量很大時，屆時以網路位址轉換負載平衡伺服器將會是整個叢集伺服器的新瓶頸。

為了解決以網路位址轉換作為負載平衡伺服器的問題，在進一步的研究下，可以發現目前許多網際網路的服務都有著以下相同的特點：那就是使用者的請求遠小於伺服器回應的資料，例如在 HTTP 協定中：使用者可能送出 GET test.jpg 等幾個 Bytes 的請求，而伺服器卻是要回應幾百 Kbytes 的資料，而目前熱門的 ADSL(asymmetric digital subscriber line)就是利用這種網路特性。

在此種網路環境下，如果叢集伺服器能將請求與回應分開來處理。亦即使用一台具 IPIP 通道封裝能力之虛擬伺服器(virtual server)作為負載平衡伺服器來負責分配使用者之請求(request)。而叢集中每個節點(node)的真實伺服器(real server)直接將處理完的結果回應給使用者。如此一來，將大大的提升整個叢集伺服器的網路吞吐量。圖 3 即為使用 IPIP 通道封裝之負載平衡伺服器系統。

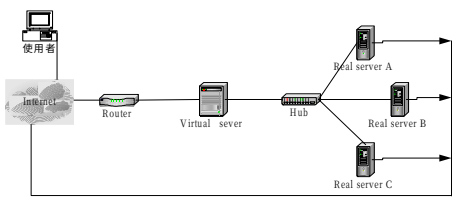

圖 3.使用 IPIP 通道封裝之負載平衡伺服器系統

3.負載平衡演算法

目前較常見的負載平衡演算法，大致有輪詢演算法 、加權式輪詢演算法 、最少連線演算法 、加權式最少連線演算法 多種，下列將針對這四種演算法之優、缺點加以探討，並提出另一較適宜之演算法。

輪詢演算法是假設所有伺服器(指 read server)，其處理能力均相同，能依序將來自外界的請求來分配到不同的伺服器上。方法較簡單，但若是每一伺服器，有著不同的機器效能，則此演算法，將不能把伺服器效能發揮到極致。

加權式輪詢演算法是改良自輪詢演算法，使用相對應的權重來表示此伺服器的處理效能，如此一來，請求數目將按照權重比例分配到每個伺服器。權重值較高者所處理的請求數將會比權重值低者多，藉此以充分利用每部機器。假如將權重值設成一樣，將造成與輪詢演算法一樣的效果：每個伺服器都處理著相同的請求數。

加權式輪詢演算法，本身也有著一些缺點，由於伺服器之效能是按不同負載而有所不同，因此要如何來決定權重值，便是一件困擾的事。例如目前之網站大都朝向多媒體方向發展(包含聲音、影像或是圖片)，其伺服器負載大小都不盡相同。假設伺服器 A 處理能力是伺服器 B 的二倍，於是我們設其權重為 A(2)、B(1)，現在若有甲、乙 2 個請求進來。伺服器 A 處理甲、伺服器 B 處理乙。但是甲的請求卻是一個影像，需較多資源；而乙的請求卻是一般文字而已。在理想狀態下，宜將下一個請求分配給乙，方能獲的較好效能。但是加權式輪詢演算法演算法卻依舊把請求分配給自顧不暇的伺服器 A，如此一來反而得到反效果。

最少連線演算法，則是考慮每位使用者所需的服務時間不一，有些較長，有些則很短，為了能真正平均分擔客戶端的連線流量，因此

最少連線演算法會將下一個來自使用者的請求分配到連線數最少的伺服器上。

其缺點則是在目前多樣化的網路環境下，或許連線數少的伺服器，他在當時的負載卻是非常重，若再依最少連線演算法再分配一些請求，將反而降低整體的效能。

加權式最少連線演算法在考慮將新連線分配到目前連線數最少的伺服器時，加上一權重值，使得在不同效能的異質環境上，能充分利用每個伺服器的能力。其缺點是權重值不易決定，同時在目前多樣化的網路環境，亦不能針對伺服器當時的工作量適時來做調整。

依據上列系統與演算法之分析，於本論文所設計之叢集伺服器系統中，我們將採用 IPIP 通道封裝之系統，提出一個動態最少工作量與連線數(dymaic least load and connection,DLLC)之演算法，此演算法能依每個真實伺服器當時之工作量與連線數動態分配使用者之請求，而達到增進整體效能之目的。

4 最少工作量與連線數演算法

為了能反映出目前伺服器本身效能，以更適切的符合負載平衡的要求，在本論文中我們將提出一個新的負載平衡演算法，名為最少工作量與連線數演算法。此最少工作量與連線數演算法是一種能更適切、有效地分配使用者之請求，使整個叢集伺服器達到更好的負載平衡之效果。而不管其叢集中的節點(指 real server)，處理能力是否相同，均能適用。同時 DLLC 演算法的另一個好處則是能提供系統之高可用性(high availability)，若是某一真實伺服器故障，則不會將請求分配到此伺服器上，進而能讓企業的網站提供良好的服務給使用者。

假設有 n 個 real server，每個 server i 的 load 為 L_i(i=1…n)，average_load=

$$\frac{\sum_{1}^{n} L_i}{n}$$

每個伺服器與平均負載的比值為 Lri=

$$\frac{L_l}{average\_load}\times 100$$

而目前連線(connections)數目為 C_i(i=1…n)，下一個網路請求將會被分配到比值與連線數乘積最低的伺服器上，計算式如下:

$$\min(Ci*Lri)\ (i=1\ldots n)$$

舉例來說，若 real server：A、B、C，其目前 Load 為：50、30、70，而連線數目(connections number)各為 10、8、12，那麼經由計算結果：1000、480、1680，故得到最小的為 server B(計算值為 480)，所以下一個連線請求，將會被分配到 server B。

5.動態負載平衡之叢集伺服器系統

本論文所建構之以 IPIP 通道封裝技術為基礎之動態負載平衡叢集伺服器系統如圖 4，在本章中分別以 IPIP 通道作為基礎的負載平衡叢集伺服器及負載平衡演算法(DLLC)，等二大部分來加以敘述。圖 5 為本叢集伺服器之軟體系統方塊圖。

在圖 5.中，當使用者發出一請求時，IP 封包來到虛擬伺服器，IP 通道封裝會先去詢問負載平衡演算法(ip_vs_dllc)要將請求分配給誰。然後 IPIP 封包來到真實伺服器會先解封裝，還原成原來的 IP 封包，接下來伺服器處理完請求後，直接回傳給使用者。

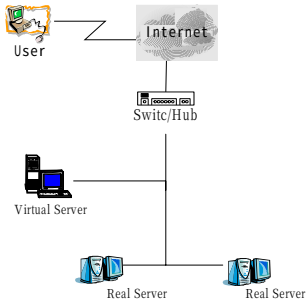

圖 4. 以 IPIP 通道封裝作負載平衡之叢集伺服器系統

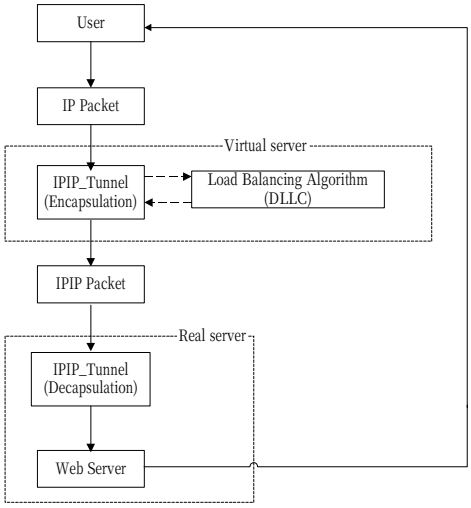

圖 5.軟體系統方塊圖

5.1 效能分析

為了驗證本論文所架構之叢集伺服器系統及 DLLC 演算法將會有較佳的效能，有需要著手進行效能之測試。在本章中有關實驗環境架設、實驗數據的取得、繪製圖表、效能分析與比較將詳述於下，藉以驗證系統之優劣。

為了驗證本叢集伺服器的優劣，在實驗之環境中暫以 2 個節點（即 2 台 real server）來作為實驗的對象。同時為了測試 DLLC 負載平衡演算法的效果，所選定之硬體設備將有著不同的執行效能（詳細規格如表 1)。實驗環境配置則如圖 6 所示。

表 1 測試機器規格

PC	CPU	Memory	Disk space	NIC
Virtual server	P-200	64 MB	2.5 GB	100 Mbps
real server 1	P-166	64 MB	1.2 GB	100 Mbps
real server 2	PII-266	128 MB	4 GB	100 Mbps
generate workload machine	P-133	40 MB	500 MB	100 Mbps

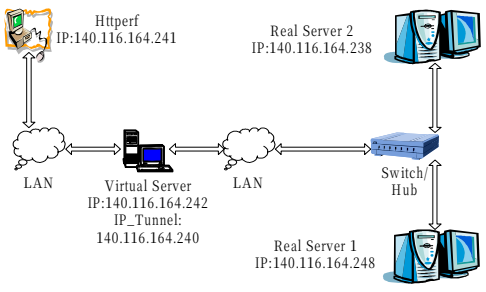

圖 6 實驗環境配置

本實驗將針對網頁伺服器來計算其效能，並藉此作為本論文所實作之系統及演算法效能分析與比較的依據。網頁伺服器採用阿帕契伺服器軟體(Apache server)，而效能評估(benchmark)軟體，則使用惠普研究實驗室(HP Research Labs)所發展的 httperf 測試軟體[9]，它是一套用來測量網頁伺服器效能的軟體，更重要的是，httperf 採用 GPL(GNU public license)的方式釋出，並附有原始碼(source code)，可讓任意修改以符合的要求。

為驗證所提出的演算法具有較佳的效能，下列項目提供參考評量：伺服器回應率(reply number per second)、伺服器回應時間(response time) 及伺服器錯誤率(error rate)等三項。

為了測試本論文叢集伺服器有著較好的效能，每個項目都將作單一伺服器與叢集伺服器之比較及負載平衡演算法：演算法有輪詢演算法(RR)、加權式輪詢演算法(WRR)、最少連線演算法(LC)、加權式最少連線演算法(WLC)四種來與動態最少工作量與連線數演算法(dynamic least load and connection,DLLC)作比較分析的工作。我們利用 HP 所提供的伺服器效能量測工具 Httperf。最簡單量測效能的方法是對預檢測的伺服器每秒鐘送給它固定速率的 request，並且量測 reply 回來的速率。重複作此實驗遞增改變 request rate，所得到的 reply rate 曲線應是逐漸增高後下降，原因是該伺服器已經到達"飽和"狀態。如圖 7 所示，針對此一虛擬伺服器所做的效能分析，其中位於上方的曲線為 reply rate 曲線，此系統所能提供有效的 request rate 最大約為 400 (飽和點)。意即超過 400 後其 error rate 會由零逐漸提升，而 reply rate 卻不再明顯增加。

此例中其反應時間在 request rate 500 以後，因為已經飽和，所以反應時間一值維持在接近 300ms 的地方。

5.2 負載平衡演算法效能比較

為了證明所提出的動態負載平衡演算法的較佳效率, 我們以圖 6 實驗室所架構的環境進行

httperf 的系統測試, 所架構的叢集環境包含一台主機伺服器及兩台 real servers (名稱為 rs1、rs2), 另外用一台主機進行對虛擬主機的 http 需求網頁測試。實驗結果如圖 8 所示，X 軸為來自使用者的請求速率(number of request per sec)，Y 軸為伺服器處理回應速率。很明顯的，rs2 單機的效能比 rs1 的效能差, 而所提出的 DLLC 效能最好, 其他 WRR、RR、LC 則表現相當。值得注意的是，利用虛擬主機所得到的效能是小於原先兩台單機效能的總合, 主要原因為執行負載平衡演算法會犧牲部份效能。

圖 9 為各種負載平衡演算法之錯誤率比較，錯誤率低，表是效能好。其中所提 DLLC 方法，甚至到 request rate=800 時，錯誤率仍小於 5%，而機器 rs2 的錯誤率是最高的。

圖 10 為各種負載平衡演算法之回應時間比較，回應時間越低者，表效能越高。圖中觀察所得結果, 各種演算法效能排名為： WLC> DLLC>WRR> RR > LC，其中 DLLC 與 WLC 在飽和情況下約為 150ms，單一主機約為 250ms，其他的演算法約為 350ms。

綜合上述實驗，證明了 DLLC 演算法優於其他演算法。

5.3 擴充性(scalability)

隨著網際網路人口不斷提升, 網路上各網站的網路流量也與日遽增, 為了有效因應網路負載的快速成長，一個企業網站的擴充性，將是決定網站成敗的關鍵所在。

在上一節實驗中, 若只是單純使用一台伺服器來服務，所能達到的 reply rate 只能達到 300 至 500 之間。若是新加入一台伺服器到叢集中，配合所提出的 DLLC 將可增加到 700 的 reply rate, 此時若再加入一台伺服器到叢集中，應可應付每秒 1000 個使用者請求。所以本論文所實作的負載平衡系統, 在網站網路流量大幅增加時, 只須加入一伺服器到叢集中即可應付高負載的網路環境。

另外在設定上，若拿 WRR、WLC 來分析，雖然它們兩種演算法改善了 RR 及 LC 的效能。但是其權重的選定，卻是一大困擾，尤其在伺服器多時。更嚴重的是，若是設錯了權重，將會導致反效果。而經由以上分析，可驗證本論文實作的負載平衡系統及演算法, 有著較好的效能且具有實用價值, 唯叢集中的伺服器，須能支援 IPIP 通道的作業系統，為其必備之條件。

7.結論及未來工作

網際網路的便利, 造就了許許多多著名的網站, 更有人因此而聲名大噪, 如雅虎(yahoo)就是一個成功的例子。而網路的普及和使用網站的便利，使得電子商務得以實現。在本論文中, 我們採用叢集伺服器之概念實作了一個以 GNU/Linux 為基礎，使用 IPIP 通道封裝技術的負載平衡叢集伺服器, 另外更提出一動態最少工作量與連線數負載平衡演算法，藉由 DLLC 演算法的幫助，來架構一個更快、更有效率之叢集伺服器系統。

本論文所提出的之動態最少工作量與連線數負載平衡演算法, 不但具備平衡負載的功用，尚有較好之可用性(availability)，而使得網站能更加的可靠。更由於利用 IPIP 通道封裝技術，使得此一叢集伺服器系統，具有高擴充性(high scalability)、高可用性(high availability)等特點。

同時目前系統所有的設定及顯示的工作，均是以命令列(command line)的方式來執行，因此設計一個以 Web 為設定及監控的介面，將是未來的課題。未來的工作，更可以加入服務品質保證(QoS)的功能，藉以達到頻寬控制，使得網站能提供更多更好的服務。

參考文獻

[1] Andrew S.Tanenbaum,”Computer

Network,”Prentice Hall,1996

[2] Aversa L., Bestavros A.,”Load balancing a cluster of web servers:using distributed packet rewriting,”Performance, Computing, and Communications Conference, 2000. IPCCC '00. Conference Proceeding of the IEEE International , Page(s): 24 –29 , 2000

[3] Bestavros A., Crovella M., Jun Liu, Martin D.,”Distributed packet rewrite and its application to scalable server architectures,”Network Protocols, 1998. Proceedings. Sixth International Conference on , Page(s): 290 –297 , 1998

[4] Cisco System,”Scaling the Internet Web Servers,”A White paper available from http:http://www.ieng.com/warp/public/cc/pd/cxsr/400/tech/scale_wp.htm..Novermer 1997

[5] C. Perkins,”IETF RFC2003: IP Encapsulation within IP,”Available from http:http://ftp.ee.ncku.edu.tw/pub/docments/rfc/rfc2003.txt

[6] Cisco System,” Speeding and Scaling Web Sites Using Cisco Content-Delivery Technology ,“A White paper available from http:http://www.ieng.com/warp/public/cc/so/neso/ienesv/cxne/spdsc_wp.htm

[7] D. Anderson T. Yang, V. Holmedahl and O. H. Ibarra, ``SWEB: Towards a Scalable World Wide Web Server on Multicomputers," available from http: http://www.cs.ucsb.edu/Research/rapid_sweb/SWEB.html, IPPS'96, April, 1996.

[8] Damani et al, “ONE-IPTechniques for Hosting a Server on a Cluster of Machines,” Sixth International WWW Conference, April 1997

[9] David Mosberger , Tai Jin,”Httperf-A Tool for Measuring Web server performance, ”HP Research Labs,Available from http: http://www.hpl.hp.com/personal/David_Mosberger/.

[10] Harish V.C. and Brad J. Owens , “Dynamic Load-Balancing DNS:dlbDNS,” available from http: http://www2.linuxjournal.com/lj-issues/issue64/3345.html , May,2000.

[11] IBM Corporation,”The IBM Network Dispatcher,” available from http: http:://www.ibm.com/software/network/dispatcher. June 2000

[12] Ibrahim Haddad and Makan Pourazand , “Linux On Carrier Grade Web Servers,” SSC , Linux Journal , page(s) 84-90 , April 2001.

[13] Ori Pomerantz , “Linux Kernel Module Programming Guide,”available from http: http://www.linuxdoc.org/LDP/lkmpg/, 1999

[14] Rusty Russell, “Linux 2.4 Packet Filtering HOWTO,” available from http: http://www.linux.org.tw/CLDP/Packet-Filtering-HOWTO.html,2000

[15] Simpson W., "IP in IP Tunneling," RFC 1853, October 1995. Available from http: http://ftp.ee.ncku.edu.tw/pub/docments/rfc/rfc1853.txt

[16] T. Brisco, ``DNS Support for Load Balancing," Network Working Group, RFC 1794.

[17] T. Berners-Lee, R. Fielding and H. Frystyk, ``Hypertext Transfer Protocol - HTTP/1.0,"

[18] Wensong Zhang, Shiyao Jin, Quanyuan Wu , “Scaling Internet services by LinuxDirector “ , High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on
Volume: 1 ,Page(s): 176 -183 vol.1, 2000 .

[19] W. R. Stevens, ``TCP/IP Illustrated," Volume 1, Addison-Wesley,1994.

[20] W. R. Stevens, ``TCP/IP Illustrated,"

Volume 3, Addison-Wesley,1996.

[21] W. R. Stevens, ``Unix Network Programming," Volume 1 second editon , Prentice-Hall ,1998.

[22] ASPAC 計畫，「Gnuplot 導讀」，中央研究院計算中心，available from ftp: ftp://phi.sinica.edu.tw/pub/aspac/doc/95/95006.ps

[23] ASPAC 計畫，「Gnuplot 使用手冊」，中央研究院計算中心，available from ftp: ftp://phi.sinica.edu.tw/pub/aspac/doc/94/94002.ps

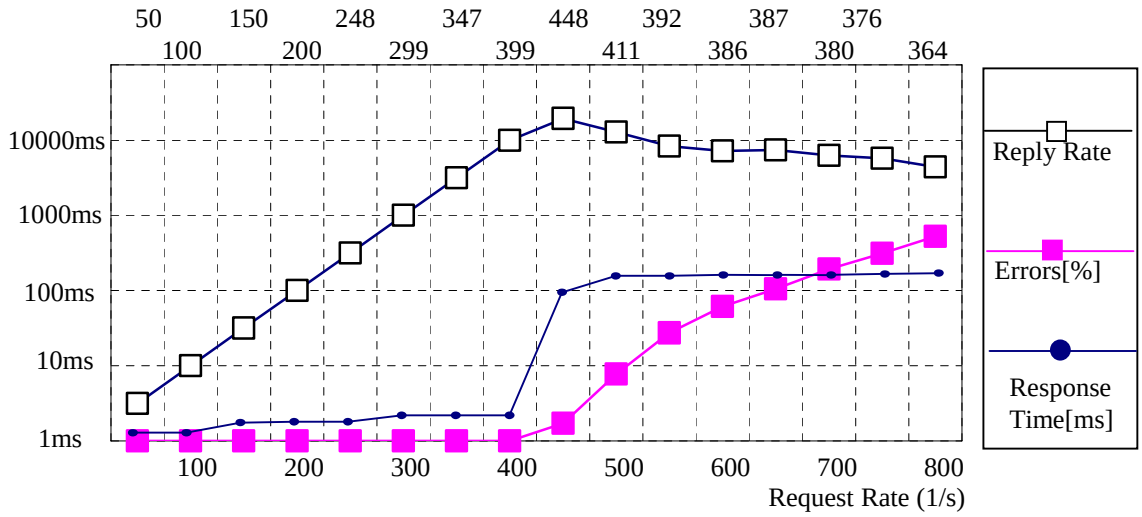

圖 7：利用 httperf 量測伺服器效能分析之案例

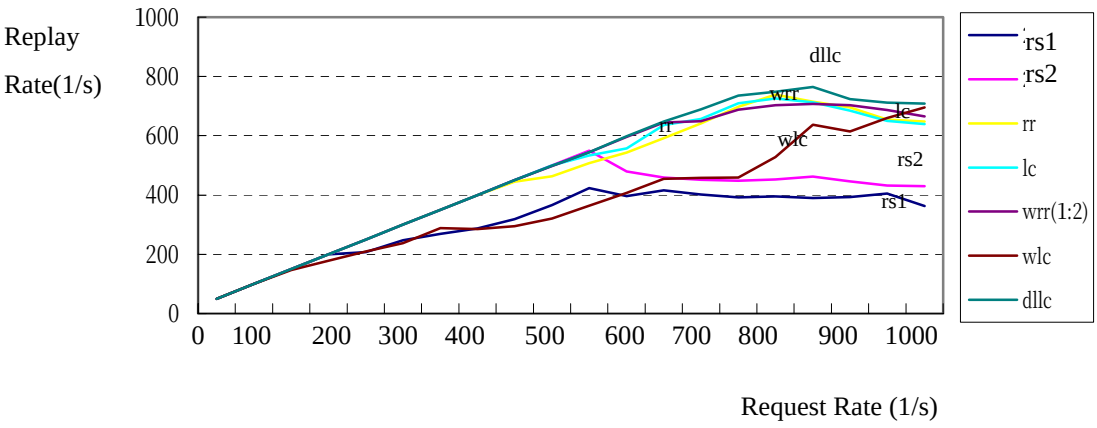

圖 8：各類負載平衡演算法之效能比較圖

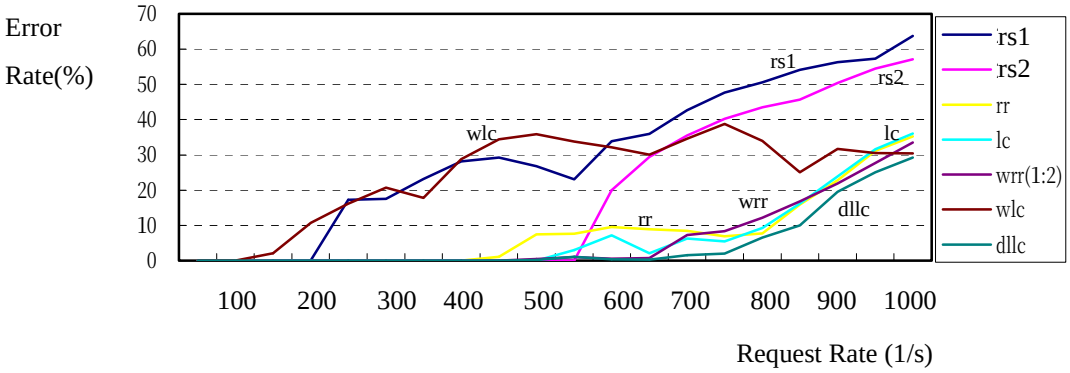

圖 9：各類負載平衡演算法之錯誤率比較圖

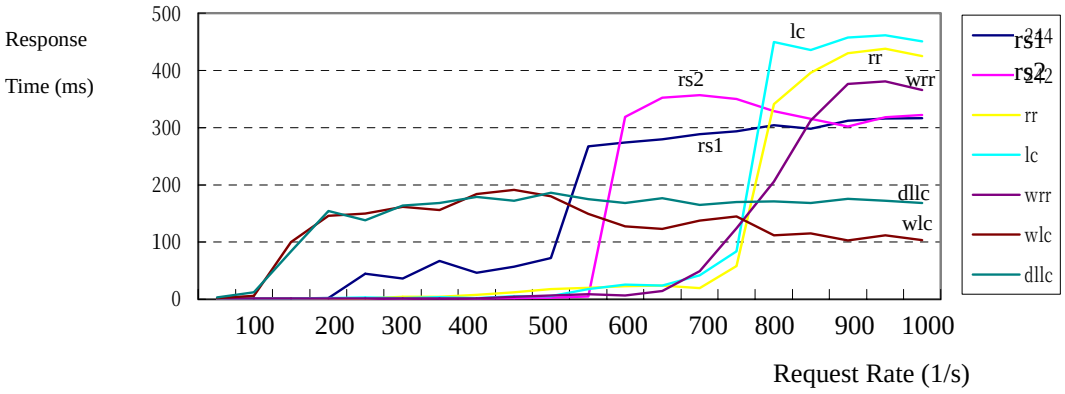

圖 10：各類負載平衡演算法之反應時間比較圖