基於需求內容分配的 Web 叢集式系統之設計與實作

Design and Implementation of Content-aware Request Dispatching for Web Clusters

劉合翰
國立暨南國際大學資訊管理系
s9213001@ncnu.edu.tw

姜美玲
國立暨南國際大學資訊管理系
joanna@ncnu.edu.tw

摘要

對於 Web 叢集式系統而言，服務需求的分配是影響整個系統效率的重要因素。為了建立有效的分配機制，本論文設計一個基於叢集式系統的網頁需求分配平台，更進一步對其應用加以實作，提出基於需求內容的分配排程演算法。此系統擷取服務需求中的內容，加以分析，依照各需求的特性來選擇負責處理的伺服器來服務。如此，對於少數需要大量資源的需求，可避免因分配不當所造成的整體叢集式系統效能下降；另一方面可針對要求認證的需求，導向某一特定的伺服器加以檢核，以維護整體的系統安全。實驗結果顯示，此系統可使 Web 叢集式系統之整體效能有效地提昇。

關鍵詞：Web 叢集式系統、Linux Virtual Server、基於需求內容之分配排程演算法

Abstract

The dispatching of requests is crucial to the performance of Web clusters. In order to build a more efficient dispatching mechanism, we design and implement a content-aware dispatching platform for Linux Web clusters. We also design a scheduling algorithm for dispatching requests in this platform. This system can extract and analysis the content in requests, then dispatch requests by their characteristics. In this way, Web clusters can avoid the overhead of some heavy requests that are not appropriately dispatched. On the other hand, Web clusters can improve the system security by directing security requests to the security-ready server. Empirical performance evaluation results indicate that the proposed mechanisms can effectively improve the performance of Web clusters.

Keywords：Web Clusters, Linux Virtual Server, Content-aware Dispatching

一、簡介

如何因應日益增加的使用者與服務需求，是企業在建構網站時的一大難題。由於單一伺服器在技術上與成本上的限制，叢集式系統已成為目前最佳的解決方案。希望藉由負載平衡(Load Balance)的概念，將客戶端(Client)的服務需求(Request)分散至不同的伺服器加以處理，以避免單一伺服器負載過重。

Linux Virtual Server (LVS)[3]是一套運作於 Linux 的叢集式系統，為 Red Hat Linux 發行版中的一部份，可提昇網頁伺服器的效率與穩定度，目前已被各界廣泛採用。然而，目前 LVS 是以 IP 為基礎，所提供的封包轉送機制均於 IP 層中實作。而且，LVS 目前的分配排程模組均無法得知服務需求的內容。本論文針對於此，以原有的 LVS 為基礎，設計並實作一個可取得需求內容的分配平台，使得 LVS 可以根據封包中的 URL 加以排程。此外，本論文進一步應用此平台，實作基於網頁類型(靜態網頁、動態網頁、安全性網頁)的分配排程法，依各類型對伺服器的負擔加以分析排程，希望提昇 LVS 的運作效率。實驗結果指出，此分配排程法於靜態網頁與動態網頁的處理效率上，皆比 LVS 原有的分配排程法優異。尤其在動、靜態網頁混合而使需求負擔有明顯差異的情況下，效能更是突出。

二、LVS 簡介

Linux Virtual Server，簡稱 LVS，是由 Wensong Zhang 發起的開發計劃，其目標是建立一個基於 Linux 的叢集式系統，並具有高性能、高可用性、高擴展性和高可靠性等優點。

2.1 LVS 之架構

LVS 的架構可分為前端分配伺服器(Dispatcher)與後端真實伺服器(Real Server)。

前端控制整個 LVS 的運作，負責封包的轉送；後端負責提供服務，回應客戶端的需求。

如圖 1 所示，客戶端提出服務需求，經 Internet 送至 LVS；前端按所設定的分配排程模組加以排程此需求，並記錄此一連線，再由封包轉送機制轉送至後端；而後端依照客戶端的需求加以回應。

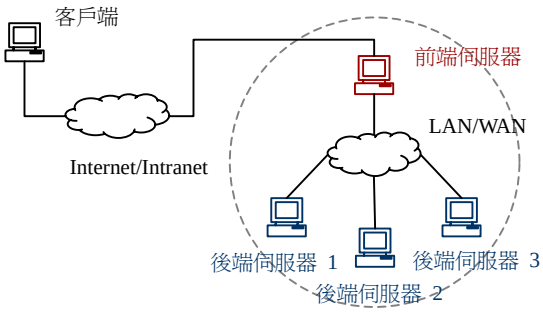


圖 1. LVS 架構圖

前端的分配排程模組目前皆以封包標頭的資訊作為分配排程之用，無法得知封包的內容。而本論文所建構的平台均支援 LVS 內建的分配排程模組與封包轉送機制，同時新增以 URL 為基礎的分配排程模組。

2.2 LVS 之核心與運作流程

LVS 分為兩部分實作，IPVS 模組與 IPVSADM 管理程式。如圖 2 所示，IPVS 模組為前端核心(Kernel)的一部份，位於 IP 層的上方，負責檢查封包的目的埠(Destination Port)是否為 LVS 所提供的服務埠，並將封包轉送至後端。

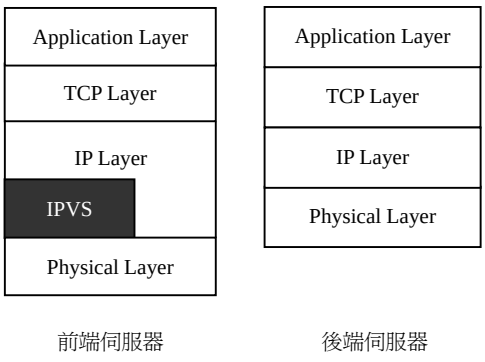


圖 2. LVS 核心網路模組示意圖

圖 3 描述 LVS 於 Direct Routing[4]封包轉送機制下的運作流程，按封包順序說明如下：

1. 按照 TCP 的連線規則，客戶端必需先與 LVS 進行交握(Handshake)，因此客戶端發出 SYN 封包，要求建立連線，開始進行 Three-way Handshaking。
2. 前端收到此封包後，交由 IPVS 模組進行判斷。若目的埠為 LVS 所服務，則呼叫分配排程模組來選擇一台後端，並記錄此一客戶端與後端的連線記錄，然後經由封包轉送機制轉送至後端。
3. 後端收到來自客戶端的 SYN 封包後，TCP 模組會回應 SYN-ACK 封包，同意連線。
4. 客戶端收到後端的 SYN-ACK 封包後，回應 ACK 封包，此時客戶端的 TCP 狀態成為連線建立(ESTABLISHED)。
5. 前端收到 ACK 封包後，進行連線記錄查詢，若記錄存在，則將封包轉送至記錄中的後端。後端接收到 ACK 封包，TCP 狀態成為連線建立，到此 Three-way Handshaking 完成。
6. 客戶端提出網頁需求，發出 PSH 封包，內含 URL 的資訊。
7. 前端進行連線記錄查詢，將 PSH 封包轉送後端。
8. 後端回應 ACK 封包，表示已收到客戶端的需求。
9. 後端完成客戶端的需求，將網頁傳送至客戶端。
10. 客戶端回應 ACK 封包，表示已收到封包。
11. 前端進行連線記錄查詢，將 ACK 封包轉送後端。
12. 後端發出 FIN-ACK 封包，表示資料傳送結束。
13. 客戶端回應 ACK 封包，表示已收到封包。
14. 前端進行連線記錄查詢，將 ACK 封包轉送後端。
15. 客戶端發出 RST 封包，結束連線。

16. 前端進行連線記錄查詢，將 RST 封包轉送後端。後端收到 RST 封包，結束連線，服務完成。

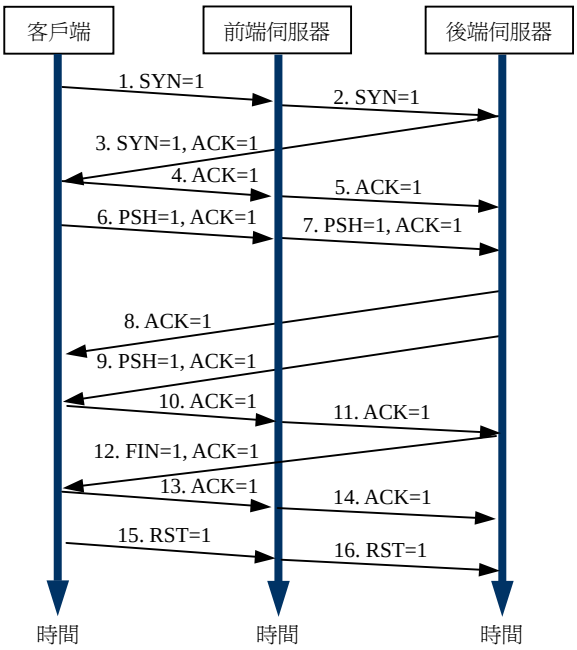


圖 3. LVS 封包轉送流程圖

2.3 LVS 依封包的標頭資訊進行分配排程

由圖 3 可知，LVS 在前端收到第一個 SYN 封包時，即呼叫 LVS 的分配排程模組來選擇負責服務的後端。由於此封包只含標頭資訊，不含內容，所以分配排程模組無法以封包內容進行排程。當前端稍後收到含 URL 的封包時，由於連線記錄已存在，所以並不會進行排程，而是依照連線記錄轉送至記錄中的後端伺服器來服務此需求。

三、需求內容分配平台之設計與實作

近年來由於電子商務與企業電子化的蓬勃發展，動態網頁已成為目前最熱門的網頁技術。但動態網頁與靜態網頁所消耗的伺服器資源並不相同，而且兩者間差異頗大，因此叢集式系統需要改進以往適用靜態網頁的分配排程法，針對動態網頁的特性加以重新設計。

然而，現有的 LVS 並無法依需求內容來做分配排程，亦無法區分動、靜態網頁，因此本論文提出可得知需求內容之分配平台，稱之為 LVS-CAD(Content-aware Dispatching)，並實作基於 URL 的分配排程法，依需求的各網頁類型對伺服器所造成的負擔加以分析排程，分散後端的負擔，以提昇 LVS 的運作效率。

3.1 LVS-CAD 之架構與核心

LVS-CAD 之架構與 LVS 相同，採取前端分配伺服器與後端真實伺服器互相搭配的架構，但前端核心以我們所設計的 IPVS-CAD 模組取代原有的 IPVS 模組，而後端核心的 TCP 模組也經過修改，以符合此架構的連線需求。

由於架構相同，故 LVS-CAD 亦支援 LVS 內建的八種分配排程模組與三種封包轉送機制。

3.2 LVS-CAD 之運作流程

此部分為本論文研究之重點，包含兩部分，3.2.1 節為客戶端與前端間 TCP 模組之交握(Handshake)；3.2.2 節為實作前端與後端間 TCP 狀態移轉(Migration)之技術。

3.2.1 TCP 模組交握

如要根據 URL 做分配排程，則呼叫分配排程模組的時機應該延後至前端接收到客戶端的需求封包時。但按照 TCP 的連線規則，客戶端的 TCP 狀態要先成為連線建立，之後才可發出含有 URL 的需求封包。因此，前端必需先與客戶端進行 Three-way Handshaking，才可獲取客戶端的需求封包。圖 4 描述 LVS-CAD 的 TCP 模組交握運作流程，按封包順序說明如下：

1. 客戶端發出 SYN 封包，要求建立連線，開始進行 Three-way Handshaking。
2. 前端收到此封包後，交由 IPVS-CAD 模組進行判斷。若目的埠為 LVS-CAD 所服務，則同意連線。IPVS-CAD 模組利用 SYN 封包中的資訊產生 SYN-ACK 封包，並傳送至客戶端。
3. 客戶端收到後端 SYN-ACK 封包後，回應 ACK 封包，此時客戶端的 TCP 狀態成為連線建立。前端收到此封包後，IPVS-CAD 把封包送往上層的 TCP 模組，則 TCP 狀態亦成為連線建立，到此 Three-way Handshaking 完成。

客戶端接著發出需求封包，內含 URL 的資訊。前端收到此封包後，呼叫分配排程模組加以排程，轉送後端。

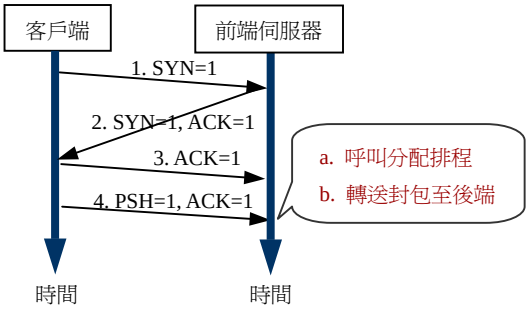


圖 4. LVS-CAD TCP 模組交握之封包流程圖

由以上可知，LVS-CAD 實作 Three-way Handshaking 的方法為：IPVS-CAD 模組直接以 SYN 封包的資訊產生 SYN-ACK 封包，並傳送至客戶端，然後直接丟棄客戶端所回應的 ACK 封包。此法的優點為：前端在整個交握的過程中，不需進入 TCP 層處理封包，可縮短回應的時間。因此，若前端搭配其他更為複雜的分配排程模組時，亦可保持良好的運作效率。

若改以前端的 TCP 模組處理 Three-way Handshaking，則會造成前端 TCP 連線的數量受限於 Apache 與前端必須自行關閉 TCP 連線等問題。

3.2.2 TCP 狀態移轉

由於 TCP 模組交握後，客戶端是與前端建立 TCP 連線，但是前端並不提供服務，應由後端來處理。因此，前端必需將 TCP 模組的連線資訊移轉至後端，使後端能完成服務需求。

目前已有學者專家提出移轉 TCP 狀態的技術，如：TCP Splicing[5]、Direct Connections[7]、TCP Handoff[6]、One Packet TCP State Migration[8]等等，而其中 One Packet TCP State Migration 技術為解決叢集式系統於提供 Session 機制時所延伸出的問題，主要專注於 Filtering Server 於 Cookie 的重新改寫。而本論文的實作類似此技術，但不同的是本論文利用前端轉送至後端的一個需求封包，即可實現 TCP 狀態的移轉，不需建構 Filtering Server。如此，即可有效降低前後端網路間的負擔，並減少後端 TCP 模組處理交握封包的時間。

本論文將 One Packet TCP State Migration 實作於前端的 IPVS-CAD 模組與後端的 TCP 模組之中。前端部分已於上一節中說明，即為 TCP 模組之交握。而後端部分的困難度較高，為 TCP 連線之重建。如圖 5 所示，在原始的 TCP 模組下，後端收到轉送的需求封包時，由於 TCP 模組尚未建立連線，所以後端會丟棄此封包，並發出 RST 的封包至客戶端以結束連線。所以後端的 TCP 模組需要加以修改，使需求封包被 TCP 模組處理前，先重建與客戶端的 TCP 連線，提供連線資訊，轉變 TCP 狀態成為連線建立。如此，後端的 TCP 模組即可以原有的流程處理與客戶端間的封包。

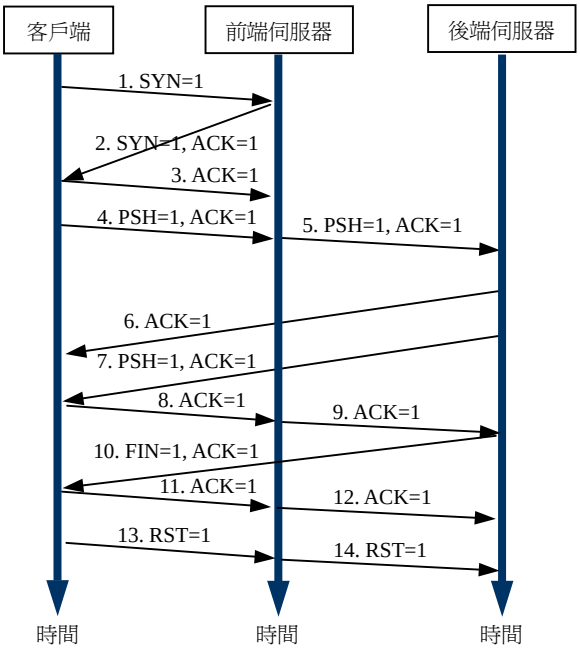


圖 5. LVS-CAD 封包轉送流程圖

後端 TCP 連線的重建主要於 tcp_v4_rcv() 這個函式中實作，此函式為 Linux 核心處理封包時，在 TCP 層所呼叫的第一個函式。

我們在實作時，先以 Kernprof [2]得知 TCP 模組於 Three-way Handshaking 時所呼叫的函式，並將這些函式加以修改。當後端接收到需求封包後，若 TCP 的狀態為傾聽(LISTENING)，則開始進行重建連線，呼叫經修改的函式。如此，TCP 模組即可獲得連線資訊。如圖 6 所示，左圖為客戶端與前端進行 Three-way Handshaking 時 TCP 模組的處理步驟；右圖為後端重建連線時 TCP 模組的處理步驟，依步驟順序說明如下：

1. 前端接收到來自客戶端的 SYN 封包，開始進行 Three-way Handshaking。
2. 前端呼叫 TCP 模組中處理 SYN 封包的原始函式。
3. 前端產生初始 Sequence Number，製造

SYN-ACK 封包，並傳送至客戶端。

4. 前端接收到來自客戶端的 ACK 封包。

5. 前端呼叫 TCP 模組中處理 ACK 封包的原始函式，到此 Three-way Handshaking 完成。

6. 後端接收到 PSH 封包，開始重建連線。

7. 為了達到圖 3 步驟 2 的效果，後端必需由 PSH 封包的 Sequence Number 反推圖 3 步驟 2 中 SYN 封包的 Sequence Number，並且設定 MSS 的值，使此 PSH 封包與 SYN 封包擁有相同的標頭資訊。

8. 後端呼叫經修改的 SYN 處理函式，使 TCP 模組認為接收到 SYN 封包。

9. 為了達到圖 3 步驟 3 的效果，後端必需由 PSH 封包的 Acknowledge Sequence Number 反推圖 3 步驟 3 中 SYN-ACK 封包的 Sequence Number，並產生 SYN-ACK 封包。但 SYN-ACK 封包不可發送至客戶端，以免擾亂客戶端的 TCP 模組，造成連線終止。

10. 為了達到圖 3 步驟 6 的效果，後端必需由 PSH 封包的 Sequence Number 和 Acknowledge Sequence Number 得知圖 3 步驟 6 中 ACK 封包的 Sequence Number 和 Acknowledge Sequence Number，使此 PSH 封包與 ACK 封包擁有相同的標頭資訊。

11. 後端呼叫經修改的 ACK 處理函式，使 TCP 模組認為接收到 ACK 封包。到此後端的 TCP 連線重建完成。

由圖 6 的右圖可知，後端重建連線時只使用一個 PSH 封包，與其他使用多個偽造封包來重建連線的技術[1]相比，較為節省 CPU 與記憶體資源。雖然此技術會遺失 SYN 封包中的 Window Size 與 MSS，但 Windows Size 的值是隨時調整的，而且可由後續的封包提供；至於 MSS 的值設定為 1460，目前主流的作業系統均可接受，所以兩者所造成的影響十分輕微。

如圖 5 所示，連線重建後，後端即以原來的流程處理封包，提供服務。而移轉後的連線可靠度由客戶端與後端共同維持，例如 PSH 封包於前端轉送至後端的過程中遺失，則客戶端會重新發送。如此，加上移轉前的 TCP 模組交握，即可確保整個連線過程中的可靠度。

前端處理交握的步驟	後端重建連線的步驟
1. 收到 SYN 封包 SEQ = C_ISN ACKSEQ = 0	6. 收到 PSH 封包 SEQ = C_ISN + 1 ACKSEQ = D_ISN + 1
2. 呼叫原始 SYN 函式	7. SEQ = C_ISN ACKSEQ = 0 MSS = 1460
3. 送出 SYN-ACK 封包 SEQ = D_ISN ACKSEQ = C_ISN + 1	8. 呼叫修改之 SYN 函式
4. 收到 ACK 封包 SEQ = C_ISN + 1 ACKSEQ = D_ISN + 1	9. 製造 SYN-ACK 封包 但不送出 SEQ = 0 ACKSEQ = C_ISN + 1
5. 呼叫原始 ACK 函式	10. SEQ = C_ISN + 1 ACKSEQ = D_ISN + 1
	11. 呼叫修改之 ACK 函式

C_ISN：客戶端初始 Sequence Number

D_ISN：前端初始 Sequence Number

圖 6. LVS-CAD TCP 狀態移轉示意圖

四、基於需求內容之分配排程演算法的設計與實作

本論文所設計的兩種基於需求內容的分配排程演算法為 CA-MAR(Content-aware Most Available Resource)與 CA-WLC(Content-aware Weighted Least-Connection)，主要依據客戶端需求的網頁類型對後端造成的負擔加以分配排程，使後端的負載盡可能的平衡。

以上兩種演算法將網頁類型區分為三類：靜態網頁、動態網頁、安全性網頁。靜態網頁包含一般的 HTML 網頁、圖形檔案、Flash 檔案等等，這類型的網頁主要消耗伺服器的磁碟資源。動態網頁包含 PHP 網頁、CGI 網頁等等，這類型的網頁依撰寫的不同，而消耗不同的伺服器資源，需要再加以區分。安全性網頁包含以 SSL 通訊協定所傳送的網頁，需要導向某一特定的後端加以檢核。以下將分別介紹 CA-MAR 演算法與 CA-WLC 演算法針對各類型網頁的分配排程。

4.1.CA-MAR 分配排程演算法

使用 CA-MAR 演算法時，前端必須判斷客戶端所需求的網頁類型，並且記錄各後端目前的負載量，將需求分配至可用資源最多的後端。為避免後端回傳負載資訊造成內部網路的

壅塞，實作上前端必須由需求封包中的 URL 判斷對後端所造成的負載量，進而累計該後端的總負載量，再由臨界值(Threshold)減去總負載即可得知該後端目前的可用資源。而實作上可使用測效軟體測試各後端的臨界值。

因此，URL 與負載量的轉換與後端的臨界值都是使用 CA-MAR 演算法時必須設定的參數。若採取事先以測效軟體測知參數，雖可求得較精準的數據，卻會降低演算法的實用性；若不採取實測，則會降低分配排程的準確性。

面對以上的問題，本論文選擇以實測求得精準之參數，此法適用於網頁內容不常變動的網站。至於經常變動網頁內容的網站，基於實用性的考量，可以採用 CA-WLC 演算法。

4.1.1 靜態網頁

由於靜態網頁以消耗磁碟資源為主，所以分配排程以檔案的大小為基準。當管理者使用 IPVSADM[5]設定前端時，IPVS-CAD 模組取得整個網站靜態網頁的檔案資料，包含各檔案的名稱與大小，並以此建立 Hash Table。而管理者必須設定各後端所能負荷的檔案大小總數，作為分配限制。實作上可使用測效軟體測試各後端的臨界值，以設定分配限制。

封包轉送的流程為：前端接收到需求封包後，由 URL 中取得檔案名稱，查詢 Hash Table 以取得需求網頁的檔案大小。前端再將此需求封包轉送至處理檔案大小總數最小的後端，其中以尚未超出分配限制的後端為優先。

4.1.2 動態網頁

由於動態網頁依撰寫的不同，會造成伺服器不同的負擔，所以必須以實測得知各網頁對於後端所造成的負擔。當網站於後端建置完畢後，管理者於客戶端執行 PHP 測試程式，此程式會由客戶端開啟網站全部的動態網頁，並記錄所花費的時間，產生 Hash Table。而管理者使用 IPVSADM 設定前端時，會載入此 Hash Table。而管理者設定各後端所能負荷的處理時間總數，作為分配限制。實作上可使用測效軟體測試各後端的極限值，以設定分配限制。

封包轉送的流程為：前端接收到需求封包後，由 URL 中取得檔案名稱，並查詢 Hash Table 以取得需求網頁的處理時間。前端再將此需求封包轉送至處理時間總數最少的後端，其中以尚未超出分配限制的後端為優先。

至於有部分動態網頁是與客戶端的輸入有關的，例如搜尋資料庫。此演算法將其視為會消耗大量的伺服器資源，必須由管理者使用 IPVSADM 設定每個後端對於此網頁的處理上限總數，而前端於分配排程時會選擇距離上限總數最遠的後端，以分散負擔。實作上可使用測效軟體測試各後端的極限值，以設定上限總數。

4.1.3 安全性網頁

由於 SSL 通訊協定所使用的通訊埠為 443 埠，與一般網頁的 80 埠不同。前端可直接由通訊埠加以辨識，將其轉送至所指定的後端加以處理。

4.2 CA-WLC 分配排程演算法

CA-WLC 演算法由 LVS 原始的 WLC 演算法改進而成。使用 CA-WLC 演算法時，前端由需求封包中取得 URL 並判斷所需求的網頁類型。而計算後端目前的連線數目時，CA-WLC 演算法會將靜態網頁、動態網頁、含客戶端輸入之動態網頁、安全性網頁等分別獨立計算，再將需求導向同類型連線數目最少的後端，以分散負擔。

與 CA-MAR 演算法相比，CA-WLC 演算法雖然於分配排程上較不精準，但卻擁有實用上的優勢。管理者不需處理參數的設定，因此適用於經常變動網頁內容的網站。

五、實驗與討論

本章說明實驗的環境與設計，並且以實驗數據驗證所設計的排程演算法的效能表現。實驗分為兩部分，一為 LVS 與 LVS-CAD 兩種平台於封包轉送的效能比較；另一為基於需求內容的分配排程模組與 LVS 內建的分配排程模組於封包轉送的效能比較。

5.1 硬體架構與環境

本實驗所使用之硬體配備詳列於表 1。實驗平台以六台 PC 作為測試端；以五台 PC 建構叢集式系統，其中一台 PC 作為前端，四台 PC 作為後端。如圖 7 所示，為確保實驗的公平及正確性，實驗平台架設於內部私有網路中，以大型交換器(DLink DES-3225G)互相連結，隔離外部網路的影響。

表 1. 實驗平台硬體環境

	CPU	Memory	H.D.	NIC
控制端/客戶端 1	P4 2.4 GHz	DDR 256 MB	7200 RPM	Realtek RTL8139
客戶端 2	P4 2.4 GHz	DDR 256 MB	7200 RPM	Realtek RTL8139
客戶端 3	P4 2.4 GHz	DDR 512 MB	7200 RPM	Realtek RTL8139
客戶端 4	P4 2.4 GHz	DDR 512 MB	7200 RPM	Intel PRO/100
客戶端 5	P4 2.4 GHz	DDR 256 MB	7200 RPM	DLink DFE 550-TX
客戶端 6	Celeron 1.7GHz	SDRAM 256 MB	7200 RPM	SiS 900
前端伺服器	P4 2.4 GHz	DDR 512 MB	7200 RPM	Realtek RTL8139
後端伺服器 1	P4 2.1GHz	DDR 256 MB	7200 RPM	3Com 3C905
後端伺服器 2	P3 800 MHz	SDRAM 128 MB	7200 RPM	SMC 1255TX
後端伺服器 3	P4 2.4 GHz	DDR 256 MB	7200 RPM	Realtek RTL8139
後端伺服器 4	P4 2.4 GHz	DDR 256 MB	7200 RPM	Realtek RTL8139

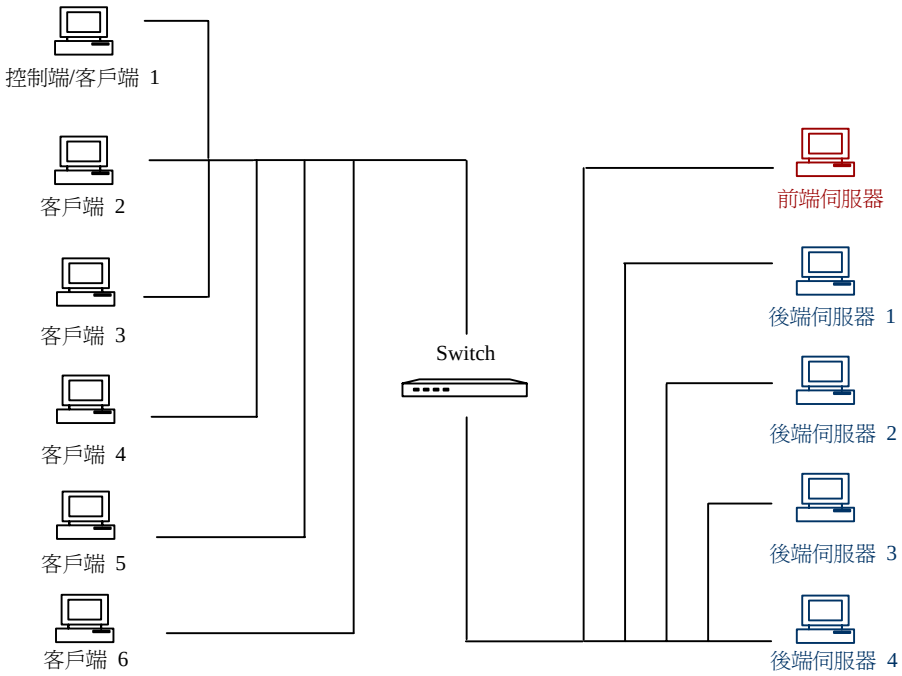


圖 7. 實驗平台架構圖

5.2 軟體環境

表 2 列出實驗平台所使用的各項軟體，其中 WebBench[12]為網站的效能測試軟體，以每秒回應的需求數(Requests/Sec)做為評量標準。WebBench 分為兩部分，控制端(Controller)與客戶端(Client)。控制端負責控制客戶端發出需求，並且記錄測試的結果。WebBench 並且使用負載模組(Workload)設定客戶端於測試時所發出的網頁需求與各需求的比例等等。

5.3 實驗設計

為比較兩平台的封包轉送效能，LVS 與 LVS-CAD 於平台效能測試時皆使用相同的分配排程模組 WRR(Weighted Round-Robin)[3] 與 WLC(Weighted Least-Connection)[3]。至於分配排程演算法則以各類型網頁依混合比例進行測試。

測試時 WebBench 使用預設的負載模組，每個客戶端開啟十個封包發送執行緒，並以 HTTP 1.1 之連線方式發送封包。

表 2. 實驗平台軟體環境

	O.S.	Kernel	Benchmark
控制端	Windows XP	SP1	WebBench 5.0
客戶端	Windows XP	SP1	WebBench 5.0

	O.S.	Kernel	IPVS	Web Server	Database Server
前端伺服器	Red Hat Linux 8.0	2.4.18	1.0.4	Apache 2.0.40	MySQL 3.23.52-3
後端伺服器 1	Red Hat Linux 8.0	2.4.18	—	Apache 2.0.40	MySQL 3.23.52-3
後端伺服器 2	Red Hat Linux 8.0	2.4.18	—	Apache 2.0.40	MySQL 3.23.52-3
後端伺服器 3	Red Hat Linux 8.0	2.4.18	—	Apache 2.0.40	MySQL 3.23.52-3
後端伺服器 4	Red Hat Linux 8.0	2.4.18	—	Apache 2.0.40	MySQL 3.23.52-3

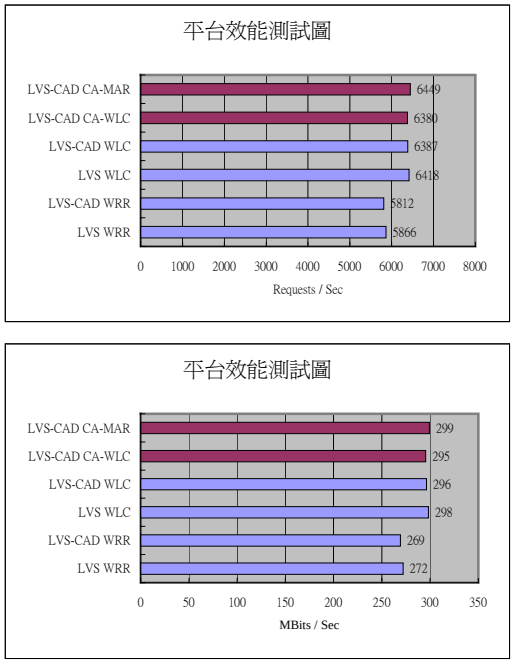


圖 8. 平台效能測試圖

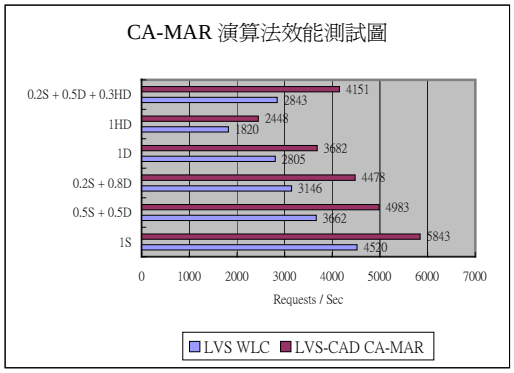
5.4 實驗結果

圖 8 為使用 WebBench 預設之 STATIC.WL 負載模組所測試的效能結果。此負載模組所測試的網頁完全為靜態網頁，而且皆為數 KB 之小檔案。LVS-CAD 使用 WRR 與 WLC 等分配排程模組時，由於後端與客戶端建立 TCP 連線的時間較晚，使需求回應受到延遲。因此，效能比 LVS 稍差，平均落後 4‰。但 LVS-CAD 使用 CA-MAR 分配排程模組時，效能即可超越 LVS 使用 WRR 與 WLC 等分配排程模組，平均領先 6.5‰。

由此可見，若忽略後端延遲所造成的效能影響，LVS-CAD 的封包轉送效率幾乎可與 LVS 相同。LCS-CAD 不需發送額外的封包、不需攔截與改寫封包、不需產生偽造封包，直接以需求封包即可重建 TCP 連線，因而減輕後端的負擔。同時後端只需處理 Three-way Handshaking 過程的封包，並且程式完全於核心中實作，因此運作效率良好。

如圖 9 所示，S、D、HD 分別代表三種網頁類型：S 代表靜態網頁，包含 HTML 網頁、大型圖片、Flash 動畫，檔案平均大小為 53 KB；D 代表會造成後端一般負荷之動態網頁，所測試的網頁以 PHP 撰寫，並搭配 MySQL 進行測試。網頁的功能為於資料庫中新增單筆資料；HD 代表與客戶端輸入有關，並會造成後端重度負荷的動態網頁。網頁的功能為於資料庫中進行全文搜尋。而網頁類型前的數字則代表混合比例，以 0.2S + 0.5D + 0.3HD 為例，

代表客戶端所發出的需求中 20%為靜態網頁，50%為一般負荷動態網頁，30%為重度負荷動態網頁。



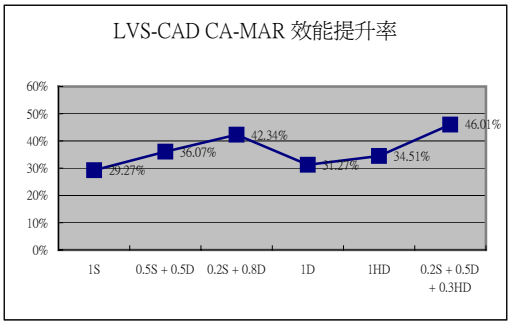
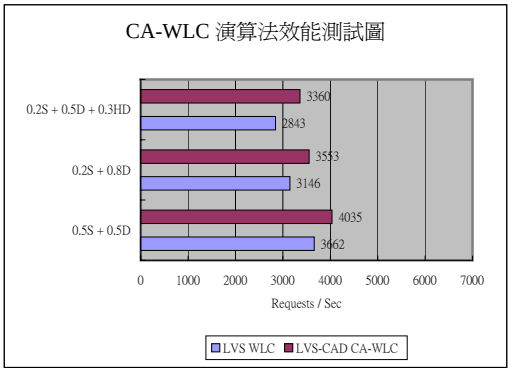


圖 9. CA-MAR 演算法效能測試圖



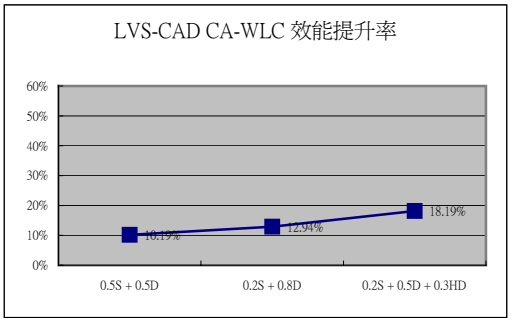


圖 10. CA-WLC 演算法效能測試圖

如圖 9 所示，相較於 LVS 使用 WLC 而言，LVS-CAD 使用 CA-MAR 於於異質性平台最多可提升 46%的效能。當網頁的檔案大小或複雜度差異越大，CA-MAR 就越能發揮其演算法的優勢，使負擔平均分擔於後端。

由於 CA-WLC 強調針對不同的網頁類型獨立計算連線數量，因此以混合各類型網頁的測試為主。如圖 10 所示，CA-WLC 對於檔案大小或複雜度具有較大差異的網頁於效能上的提升較明顯。相較於 LVS 使用 WLC 而言，LVS-CAD 使用 CA-WLC 於異質性平台最多可提升 18%的效能。

六、結論

動態網頁技術將是未來的趨勢，因此叢集式系統在分配排程上也將朝向基於需求內容的演算法發展。本論文所設計實作的平台，可搭配各種分配排程模組，提供了此方面研究的基礎。而且此平台同時兼顧封包轉送的效率與連線的可靠度。至於本論文所提出的分配排程法，亦能有效的分散後端的負擔，提昇整體效能。經實驗證明，LVS-CAD 搭配 CA-MAR 可於異質性平台提升 46%的效能，而搭配 CA-WLC 可於異質性平台則可提升 18%的效能。

未來，尚有許多課題值得進一步地研究與探討。在分配平台方面，前端與後端可預先連線以縮短 TCP 狀態移轉的時間。而在分配排程方面，可針對提高後端磁碟的快取命中率(Cache Hit Ratio)加以改進。這些都將是我們未來研究的方向。

參考文獻

[1] 黃銘毅，叢集式系統以時間為基礎之排程法與以瞭解服務內容之需求分配平台的研究與製作，國立暨南國際大學資訊管理研究所，碩士論文，六月，2003 年.

[2] Kernprof Website, http://oss.sgi.com/projects/kernprof/.

[3] Linux Virtual Server Website, http://www.linuxvirtualserver.org/.

[4] Joseph Mack, “LVS-HOWTO”, http://www.linuxvirtualserver.org/Joseph.Mack/HOWTO/index.html, July 2003.

[5] D. Maliz and P. Bhagwat, “TCP splicing for application layer proxy performance”, IBM Technical Report RC-21139, March 1998.

[6] V. S. Pai, et al, “Locality-Aware Request Distribution in Cluster-based Network Serv-

ers", Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, Oct 1998.

[7] Colby Steven, et al, "Method and system for directing a flow between a client and a server", US patent 6,006,264, December 21, 1999.

[8] Ping-Tsai Tsai and Ying-Dar Lin, "Direct Routed Web Switch with State Migration, TCP Masquerade, and Cookie Name Re-write", Open Source Software Contest, 2002.

[9] WebBench Website, http://www.etestinglabs.com/benchmarks/webbench/webbench.asp.