黎明學報 第二十四卷 第一期 中華民國一〇二年一月三十一日
JOURNAL OF LEE-MING INSTITUTE OF TECHNOLOGY
VOL. 24, NO.1, JAN. 2013

多層式網頁應用程式的允入控制
Admission Control for Multi-Tiered Web Application

朱仁貴 陳宏良
Ren-Guey Chu, Hung-Liang Chen
黎明技術學院電機工程系
Department of Electrical Engineering, Lee-Ming Institute of Technology

吳政遠 王永鐘
Chih-Yuan Wu,Yung-Chung Wang
台北科技大學電機工程系
Department of Electrical Engineering, National Taipei University of Technology,

王振興
Jenn-Shing Wang
景文科技大學資訊工程系
Department of Computer Science and Information Engineering,
Jinwen University of Science and Technology

摘 要

瀏覽人數過多所造成的網站過載，其回應時間隨請求到達率增加而明顯增長，利用允入控制機制拒絕部分請求，可避免網站過載以維持既定的回應時間。本論文實現一個具有允入控制功能的 Apache 模組，並且開發一個效能分析工具軟體，利用此軟體分析的結果設定 Apache 模組之控制參數。經由多層式網頁應用程式的實測結果證明，本論文的模組與控制策略可避免網站發生過載。

關鍵詞：允入控制、網頁應用程式、多層式、阿帕契

Abstract

Web sites become overloaded because of too many visitors. Increasing the request arrival rate prolongs the response time significantly. An admission control mechanism can prevent web sites from becoming overloaded by rejecting some requests and thereby maintains an acceptable response time. In this paper, we propose a measurement-based admission control and implement the proposed

admission control over Apache. The control parameters are set by the analytic results from our tool. Measurements from multi-tier web application show that the module and control strategies can avoid web sites overload occurred.

Key Words: Admission Control, Web Application, Multi-Tiered, Apache

1. 前言

全球資訊網(World Wide Web, WWW)發明於 1990 年代初期，超文件標示語言(HyperText Markup Language, HTML)、超文件傳送協定(HyperText Transfer Protocol, HTTP)、網頁伺服器和瀏覽器四項關鍵元件的發展造就今日的網路世界[1]。2005 年 O'Reilly 提出 Web 2.0 的應用概念，將網站視為平台使用，使得全球資訊網推向另一個世代[2]。此後提倡多樣化網際網路應用程式(Rich Internet Application, RIA)的使用概念，以網站為基礎開發各類型網頁應用程式(Web Application)，利用網際網路(Internet)部署應用程式於瀏覽器、電腦與行動裝置之上[3]。目前人們依賴全球資訊網的程度已超過 90 年代初期。

全球資訊網是建置於網際網路之上，網站無時無刻接受來自世界各地使用者的請求。在現今高度全球化與網路化的時代，當發生受人們矚目的事件時，網站將會因為瀏覽人數突然地增加而造成過載，其過載原因是瀏覽人數超出系統所設計之容量。普遍的案例有：新聞網站在奧運舉辦期間、政治選舉開票時或發生重大災難後。購物網站的特賣期間或標錯過低價格的搶購。與一般公司網站在產品發表的期間。網站發生過載時，回應使用者請求的時間將會增長，甚至造成連線失敗或網站當機。因為過長的等待時間會讓使用者選擇離開，更進一步造成商業上的損失或是對該網站的負面評價。瀏覽人數突然增加所造成的網站過載必須設法解決，由於這類型的過載問題很難預估瀏覽人數和發生時間，因此建置網站時的容量規劃較無法有效預防，若設計較大的容量將不符合經濟效益。當採用擴充的解決方式則需要時間設置，因而無法即時地處理突然增加的負載。

因此，針對這類短暫性過載的解決方式有兩種：(1)採用允入控制(Admission Control)機制，網站判斷目前的負載情形決定是否接受新的請求，當網站處於滿載時則拒絕部分請求，以避免網站進入過載[4]。(2)使用內容適應(Content Adaptation)技術，當網站處於滿載時，對於花費大量資源處理的網頁改以靜態頁面或停止處理，以便網站能有較大資源處理新的請求[5][6]。對於已經運行的網頁應用程式，若採取內容適應技術則須修改程式本身或重新開發，勢必需要停止網站服務以更新程式。因此，本論文將在現有的網頁應用程式伺服器當中，加入允入控制機制以解決過載問題，此方式可以不變動既有網頁應用程式而達到過載處理的功能。

網頁應用程式執行於網頁伺服器之中，一個網頁伺服器可視為由佇列(Queue)與服務器(Server)組成的佇列系統(Queueing System)[7][8]。佇列系統中回應時間(Response Time)與到達率(Arrival Rate)之關係隨著請求的到達率增加，其回應時間會明顯增長[9]。對網站而言，當請求到達率的增加超出額定值之後，其影響將使服務品質(Quality-of-Service, QoS)顯著下降。

Liu 等人提出多層式網頁應用程式的允入控制[10][11]，此控制器由前授控制(Feed Forward Control)與回授控制(Feedback Control)兩部分構成，其功能可控制回應時間於命令值之中。前授控制取得網站的請求到達率，使用佇列模型(Queueing Model)預估可進入系統的允入率(Admittance Rate)，而回授部分則取得回應時間，使用適應性控制調整前授控制的預估值，藉此策略有效地控制請求到達率而避免過載發生，因此可符合設計之服務品質。Liu 等人使用代理伺服器(Proxy

Server)實現允入控制，使用者的請求由代理伺服器攔截以便判斷允許或拒絕，該方式不需變動舊有系統即可達成允入控制目的。另外，Robertsson 等人所提出的允入控制則依據伺服器的利用率回授控制[12]，使用比例積分控制器調整系統的允入率，因此可避免負載超出伺服器的額定值。此研究是使用模組方式擴充 Apache 網頁伺服器的功能，使得該網站具有允入控制機制，然而此研究僅限於單部伺服器運作，而非多層式網頁應用程式之架構。

本研究將在多層式網頁應用程式架構中加入允入控制的功能，限制真正進入網站的到達率，使網站能夠維持既定的回應時間。當網站處於滿載時，允入控制機制將拒絕部分使用者的請求，避免網站繼續接收請求而造成過載。一方面讓舊的使用者不會因為過載而感受到服務品質下降，另一方面新的使用者因拒絕訊息的快速回應，節省等待時間而且不必忍受低劣的服務品質。同時間，網站管理員可以決策是否需要進一步的處理。本研究將開發一個具有允入控制功能的 Apache 模組，該模組安裝於多層式架構之中，使舊有的網頁應用程式具備過載控制的能力，因此不需額外建置具備允入控制的代理伺服器，並且提出一個簡單的控制策略，該控制參數由網站的負載特性進行設定，此模組將判斷伺服器負載與請求等待的情形允許或拒絕請求，因此在過載情形下將可提供較佳的服務品質。

本篇論文首先於第二章介紹相關研究，說明多層式架構、允入控制以及 Apache 網頁伺服器軟體。其次，為使系統達成允入控制功能，第三章說明本論文所開發的允入控制模組，敘述該模組的功能和控制策略。接著進行模組安裝和實際量測，評估本研究的控制效果，由第四章說明實驗平台，並分析實測的數據結果。最後，第五章結論說明本研究的發現和結果，提供後續研究的方向與建議。

2. 系統模型

2.1 多層式網頁應用程式

目前的網頁應用程式廣泛採用多層式架構建置，其優點是可以提高網站的擴充性(Scalability)、可用性(Availability)與安全性(Security)，並且容易與舊有系統進行整合[13]。大型的網頁應用程式採用三層式架構為主，典型的三層式架構如圖 2.1 所示[14]。

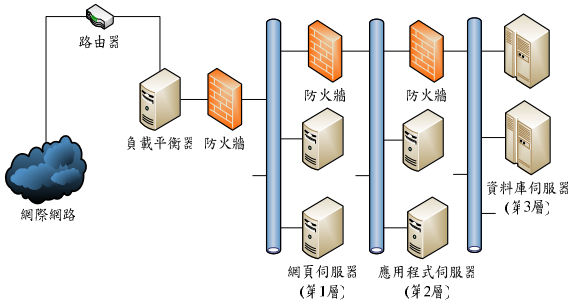

圖 2.1 典型的三層式網頁應用程式架構

第一層為展現層(Presentation Layer)，負責提供使用者介面呈現的資料，內容以不需程式執行的靜態網頁資料為主，如 JPEG 圖檔、GIF 圖檔、HTML 檔案、JavaScript 檔案以及串接樣式表(Cascading Style Sheets, CSS)等。展現層由網頁伺服器提供三項主要功能：(1)接收來自使用者瀏覽器的請求。(2)轉送複雜的動態內容請求至第二層。(3)接收來自第二層的回應，並將此回應轉送至瀏覽器。常用的伺服器軟體有 Apache HTTP Server、Microsoft Internet Information Server (IIS)、nginx 與 lighttpd 等。

第二層為商業邏輯層(Business Logic Layer)，或稱應用層(Application Layer)，

此層功能實現一系列商業規則與工作流程的運算，如進銷存貨管理、生產流程管理等。應用層接收來自展現層的請求，依據此請求向第三層搜尋資料，其回應的資料依據商業邏輯處理產生動態網頁再轉送回第一層。常用的軟體平台有 Apache Tomcat、Sun GlassFish Enterprise Server、JBoss Application Server、Zend Server、Microsoft IIS 與 IBM WebSphere 等。

第三層為資料服務層(Data Service Layer)，通常由資料庫管理系統(Database Management System, DBMS)負責搜尋與存取資料，此層負責接收來自應用層的資料請求並回應。常用的資料庫管理系統有 MySQL、Oracle、PostgreSQL、Microsoft SQL Server 與 IBM DB2 等。

三層式架構最前端由負載平衡器(Load Balancer)接收來自網際網路的請求，平均分配請求負載給後端的伺服器，避免單一伺服器超載而造成系統的瓶頸，進而提升整體多層式架構的效能。各層之間由防火牆(Firewall)隔離，僅有授權的伺服器能相互連線，確保內部設備免於遭受網際網路的攻擊，系統的安全性因此提升，而且每層之中由多台伺服器組成，故可以確保系統的可用性，避免單一伺服器故障而造成系統停擺，另外，各層之間使用網路連接而具有低耦合特性，有利於系統的擴充性。

2.2 Apache 網頁伺服器

Apache 網頁伺服器(簡稱 Apache)是目前最為廣泛使用的網頁伺服器軟體之一[15]，而且 Apache 為免費使用的開放原始碼程式，有利於各類功能開發及研究，目前最新版本為 2.2 版。圖 2.2 為 Apache 軟體內部的架構[16]，Apache 與作業系統之間使用 Apache 可移植執行階段(Apache Portable Runtime, APR)，APR 抽象化底層與系統相關的功能，並且提供 Apache 跨平台的程式庫，因此 Apache 可以於各種作業系統之上執行。

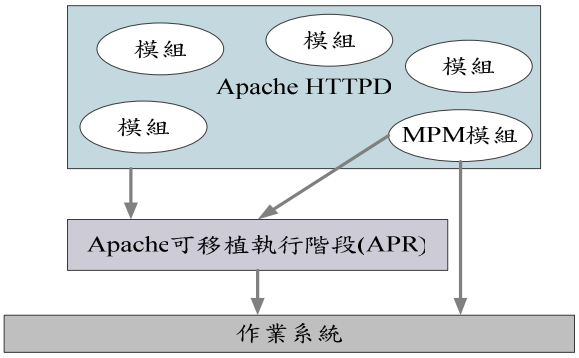

圖 2.2 Apache 網頁伺服器架構

Apache 的架構主要由一個核心程式與多個模組程式所組成，比較特殊的部分是多行程模組(Multi-Processing Module, MPM)可直接和底層作業系統溝通，此目的是針對不同作業系統最佳化設計以提高 Apache 的效能表現。針對 Windows 系統使用 WinNT MPM 模組，而 Unix 系統則使用 Prefork MPM 或 Worker MPM 模組，其他作業系統如 Netware 或 OS/2 等皆有對應的 MPM 模組。

對 Windows 系統最佳化效能的 WinNT MPM 模組是採用執行緒(Thread)的方式設計，Aapche 啟動後將產生固定數量的執行緒，由執行緒負責處理使用者的 HTTP 請求，其模組的設計架構如圖 2.3 所示。每次請求的處理到連線結束為止，該執行緒處於忙碌狀態，若未進行任何請求的處理則該執行緒為空閒狀態，即處理每一個請求將會佔用一個執行緒。當 Apache 的所有執行緒皆為忙碌狀態時，則新的 HTTP 請求會留於 Apache 佇列中，直到任一執行緒是空閒狀態為止，該請求才會被處理。其他的 MPM 模組運作方式也與 WinNT MPM 模組類似。

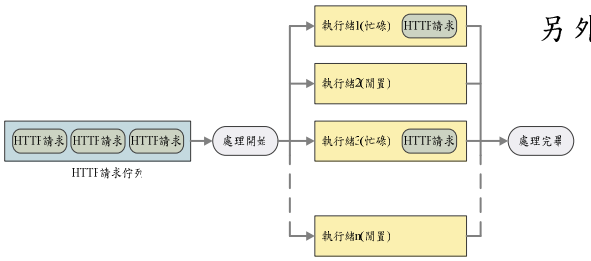

圖 2.3 WinNT MPM 模組設計架構

Apache 核心程式提供一系列掛鉤(Hook)[17][18]，掛鉤的功能是提供核心程式與模組程式溝通的介面，各個掛鉤分別位於處理請求的不同階段，同一掛鉤可以由多個模組共同使用，開始處理請求時，將依序執行掛鉤中的模組功能。因此，各模組利用掛鉤可以控制請求的處理方式，進而擴充原有 Apache 網頁伺服器的功能。

處理請求的流程如圖 2.4 所示，HTTP 請求首先進入掛鉤 1，由於掛鉤 1 已載入模組 1 與模組 2，因此該 HTTP 請求由模組 1 與模組 2 依序處理，處理完成之後轉送給掛鉤 2。掛鉤 2 接收 HTTP 請求後由模組 3 處理，模組 3 處理完成之後繼續傳遞 HTTP 請求，這種處理方式直到掛鉤 n 中的模組執行結束為止。注意到模組 1 與模組 3，Apache 中的同一個模組可使用多個掛鉤，因此 HTTP 請求會在處理請求的不同階段進入同一個模組數次，並且執行不同的功能。

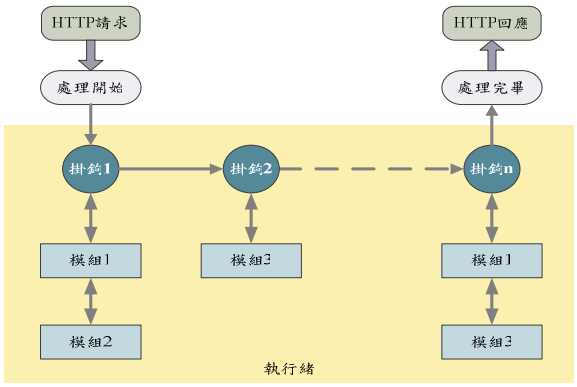

圖 2.4 Apache 處理請求的流程

另外，模組中可以傳回 HTTP 的狀態碼(Status Code)以中斷 HTTP 請求的傳遞，此 HTTP 請求隨即進入處理完畢的階段。例如圖 2.4 中的掛鉤 1，若模組 1 處理時傳回 HTTP 狀態碼，則模組 2 與後續的掛鉤皆不會收到此 HTTP 請求，此請求會進入處理完畢的階段而發出 HTTP 回應給使用者。

3. 允入控制設計

允入控制機制已普遍使用於電信網路上，當服務人數超出其系統的額定容量時，系統藉由拒絕使用者的請求以避免過載。允入控制的基本架構由三項元件組成：閘門(Gate)、控制器(Controller)及監測器(Monitor)，如圖 3.1 所示。

電腦系統上的允入控制為數位控制的觀念，將時間分割成控制區間(Control Interval)。在目前的區間當中，控制器由監測器取得回授信號，計算後決定下一個區間的允入率。於下一個區間時，閘門依據控制命令拒絕或接受請求，藉此方式控制允入率。

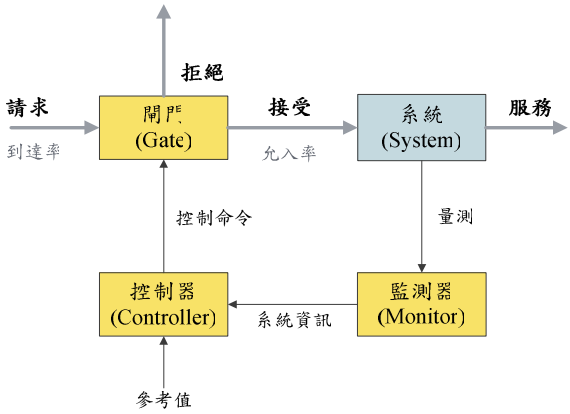

圖 3.1 允入控制架構

監測器的功能是取得系統的效能資訊，如 CPU 負載率、TCP 連線數、伺服器等待佇列長度、反應時間與請求次數等。控制器依據參考值與目前的效能資訊

產生控制命令，決定請求是否允許進入系統，藉由回授控制可使系統的效能符合參考值。

3.1 模組功能介紹

為達成允入控制機制於多層式網頁應用程式，本論文依照 Apache 所規範的介面開發，並且針對 Windows 系統進行設計，所開發的允入控制模組具有監測及控制兩項功能。

監測功能目的為取得 Apache 的系統狀態並提供允入控制之用，其監測資訊有伺服器負載情形、請求佇列長度與服務時間(Service Time)共三項數據。監測資訊不記錄於伺服器之上，而是藉由 HTTP 標頭欄位(Header Field)傳回使用者，因此可以精確取得每次請求時的伺服器服務情形，而本模組自訂的標頭欄位內容如表 3.1 所示。

表 3.1 允入控制模組之 HTTP 標頭欄位內容

欄位名稱	說明
Service-Time	該次請求的服務時間
Server-Load	接收請求時的伺服器負載情形
Server-Queue	接收請求時已等待的請求數

控制功能利用三個參數設定而進行允入控制，其參數說明如表 3.2 所示，此參數儲存於 Apache 設定檔之中，本模組於 Apache 啟動時讀取設定檔而取得三個參數，因此每次變動參數之後必須重新啟動 Apache 才能生效。本模組運作時將判斷是否啟動允入控制機制，若參數設定為關閉，則該允入控制模組僅有監測之功能。

表 3.2 允入控制模組設定參數

參數名稱	參數值	說明
Admission Control	On 或 Off	On：啟動允入控制機制 Off: 關閉允入控制機制
LoadThreshold	小數，須介於 0 至 1	Apache 伺服器負載的門檻值
QueueThreshold	整數，須大於 0	請求佇列長度的門檻值

本模組拒絕使用者請求的方式使用 HTTP 狀態碼 503，依據超文件傳送協定的定義表示為服務不可用 (Service Unavailable)，此狀態的意義是用於通知使用者伺服器過載或維護之情況。

3.2 監測方法

本節說明允入控制模組監測 Apache 三項資訊的方法與定義。

本論文將允入控制模組使用於 Windows 系統上，故 Apache 使用 WinNT MPM 模組處理 HTTP 請求，此 WinNT MPM 模組是以多執行緒方式運行，因此伺服器負載情形將利用執行緒狀態計算得知，允入控制模組由 Apache 程式庫得知目前忙碌的執行緒數量為 T_{Busy}，而閒置的執行緒數量為 T_{Idle}，則伺服器負載 S_L 的定義為(3.1)式。

$$S_L = \frac{T_{Busy}}{T_{Busy} + T_{Idle}} \tag{3.1}$$

請求佇列長度無法直接由 Apache 程式庫取得，允入控制模組必須間接計算才能得知。Apache 接受 HTTP 請求不代表會立即進行處理，其條件是必須有閒置的執行緒才能處理，否則 HTTP 請求將於佇列中等待。由於接受 HTTP 請求的 TCP 連線會持續建立直到執行緒處理完畢為止，因此利用 TCP 連線的建立數量可以推算出請求佇列長度的大小。

允入控制模組首先取得目前使用 80 埠的 TCP 連線數量，並判斷 TCP 連線狀態為建立(Established)的連線數量求得 C_{HTTP}，此數值代表 Apache 所接受 HTTP 請求的數量，而目前處理的請求數量可由 T_{Busy} 得知，因此請求佇列長度 Q_L 計算如(3.2)式所示。

$$Q_L = C_{HTTP} - T_{Busy} \tag{3.2}$$

服務時間則使用 Windows 系統中的高解析度效能計時器(High-Resolution Performance Counter)進行計算，其解析度可計算至微秒等級的時間。此計時器於 HTTP 請求進入的第一個掛鉤啟動，並於 HTTP 請求處理完畢的最後一個掛鉤將計時器停止，該段時間定義為 Apache 處理 HTTP 請求的服務時間。

3.3 控制流程

為避免網站過載所造成回應時間的增長，本論文開發的允入控制模組將考慮兩項因素進行控制策略的設計：

(1) 當 Apache 同時處理過多的 HTTP 請求時，其服務時間將會增加。這是因為伺服器上的資源有限，所有的 HTTP 請求必須共享各項軟硬體資源，於是造成服務時間的增長，若能限制同時間處理的 HTTP 請求數量，則將會縮短該次 HTTP 請求的服務時間。

(2) 當 Apache 上無閒置的執行緒時，接受的 HTTP 請求將會存放於 Apache 佇列之中等待，直到任一執行緒閒置後由佇列中取出進行處理。另外，當佇列中已經存在部分 HTTP 請求時，新的 HTTP 請求必須進行等待，直到佇列中與執行緒中的請求皆被處理完畢，此新的 HTTP 請求才能被服務處理。因此，等待的佇列長度與等待時間將有關係，若能減少等待的佇列長度，則 HTTP 請求的等待時間將能縮短。

基於上述兩項因素，允入控制模組所設計的控制策略如圖 3.2 所示，本模組使用掛鉤與 Apache 溝通，並且安裝於 HTTP 請求處理的最前端。模組依據控制策略拒絕或接受請求，控制策略首先判斷伺服器負載是否高於負載門檻值 LoadThreshold，若大於則表示伺服器將超出設計的負載容量，此 HTTP 請求將予以拒絕，藉由限制伺服器的負載可控制服務時間於期望範圍之內。

當伺服器處於輕載時，伺服器負載低於負載門檻值 LoadThreshold，控制策略接著判斷請求佇列長度是否高於佇列門檻值 QueueThreshold，此判斷目的是避免等待時間過長，因此可控制等待時間於期望範圍之內。若控制策略允許接受 HTTP 請求，則允入控制模組將轉送 HTTP 請求給其他模組服務；若拒絕 HTTP 請求則直接進入處理完畢，此 HTTP 回應將離開 Apache 並切斷 TCP 連線。

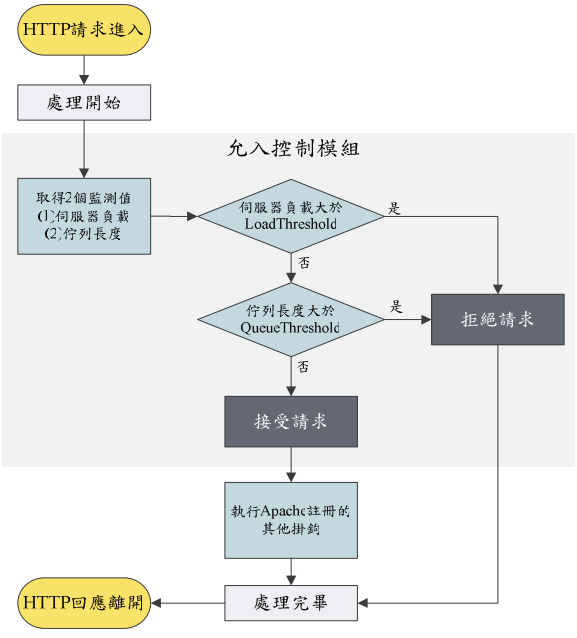

圖 3.2 允入控制模組的控制流程

3.4 允入控制運作架構

允入控制機制將安裝於多層式網頁應用程式的第一層 Apache 之中，其系統架構如圖 3.3 所示，使用者發出的 HTTP 請求首先進入允入控制模組判斷接受與否，接受請求之後交由 Apache 其餘的模組進行處理，Apache 將判斷是否轉送請求給第二層的應用程式伺服器執行，若不需要轉送則由 Apache 直接回應。

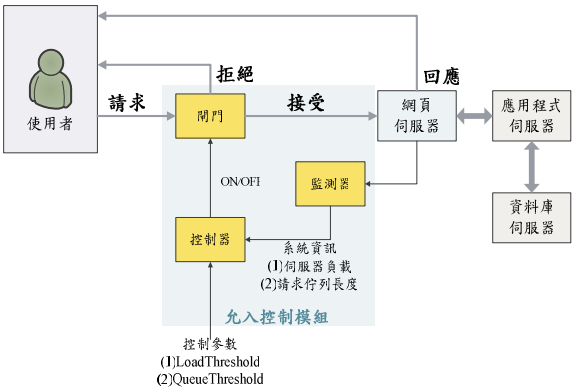

圖 3.3 允入控制模組之運作架構

4. 系統實測

4.1 實驗平台

本研究之實驗平台架構如圖 4.1 所示，三部伺服器建置成為三層式架構，提供多層式網頁應用程式執行之平台，伺服器之間以超高速乙太網路(Gigabit Ethernet, GbE)連接，可將資料傳輸時間減至最低。另外，使用一台筆記型電腦安裝效能分析工具軟體，模擬使用者使用網際網路的情形。

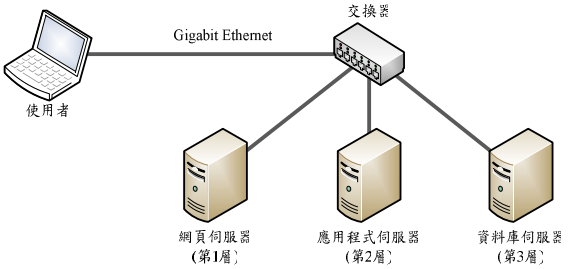

圖 4.1 實驗平台架構

系統第一層使用 Apache 網頁伺服器軟體，安裝本論文所開發的允入控制模組，此模組監測系統的負載情形以調整請求之許可與否。第二層安裝 Zend Server 應用伺服器軟體[19]，做為執行 PHP 網頁應用程式之平台。第三層則安裝 MySQL 資料庫管理系統，提供網頁應用程式的資料儲存功能。

選用這些伺服器軟體除了符合原有網頁應用程式的執行要求規格之外，並具有下列性質：(1)屬於開放原始碼並且免費使用。(2)效能表現佳。(3)個人網站至大型商業網站皆廣泛使用，具有一定可信度。

此平台上安裝之網頁應用程式為 Moodle 1.9.3 版[20]，Moodle 為一套開放原始碼的內容管理系統(Content Management System, CMS)與學習管理系統(Learning Management System, LMS)。此網頁應用程式不具有過載控制能力，因此當使用者過多時，其服務品質將受到影響，使用本研究之允入控制模組之後，不需修改此網頁應用程式即可以具有過載保護功能。

4.2 實驗條件

各層伺服器軟體的設定皆使用預設值，其中第一層因安裝允入控制模組，有關此模組運作之 Apache 的參數值特別說明如下[18]：

(1) ThreadPerChild

在 Windows 中預設值為 64，表示 Apache 將開啟 64 個執行緒服務使用者的請求，相當於佇列系統中有 64 個服務器。

(2) ListenBacklog

預設值為 511，用做 TCP 聆聽 80 埠最大可保留的連線數量，相當於佇列系統中的佇列容量。

(3) KeepAliveTimeout

預設值為 5，此功能因 Apache 支援 HTTP 1.1 之持續連線 (persistent connection)功能，使用者完成資料接收後，伺服器可以保留 TCP 連線 5 秒，避免 TCP 再次建立連線的時間。此參數將間接影響執行緒和連線的使用情形。

本實驗每次將進行 180 秒，各次實驗之間僅改變其中一個參數進行，以便獲得參數值和結果的關係。

4.3 實測數據

A. 負載測試

本實驗將各別設定 1 秒、5 秒與 10 秒的思考時間進行量測，平均回應時間與使用者數量的關係如圖 4.2 所示，當使用人數增加時，其回應時間將增加，直到系統滿載為止。以思考時間設定 1 秒為例，系統在使用者數量超過 70 個之後，平均回應時間不再繼續增加，這是因為系統已經滿載，請求將等候於佇列之中，此情形同樣發生於思考時間為 5 秒且使用者數量超過 90 個的情況。由此可知，該系統於滿載的平均回應時間約為 12.5 秒，注意此平均回應時間是以成功的 HTTP 回應而計算求得，發生逾時請求的時間並未列入計算。

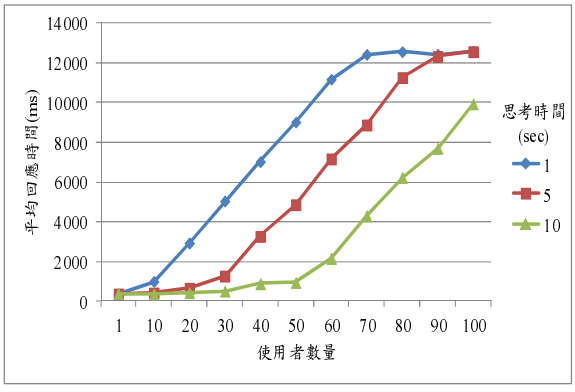

圖 4.2　未使用允入控制機制回應時間與使用者關係

圖 4.3 為回應時間與伺服器負載的關係，隨著伺服器負載的增加，其回應時間也將增長，由圖中可得知回應時間僅與伺服器負載有關，而與思考時間無關。

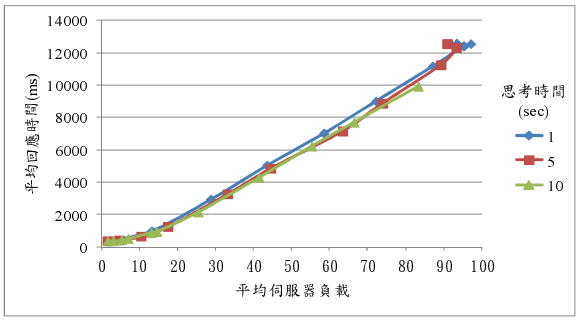

圖 4.3　未使用允入控制機制回應時間與伺服器負載關係

B. 允入控制測試

為測試本論文提出的允入控制機制之效果，今選定回應時間為 4 秒做為服務品質需求。使用允入控制機制之後，使用者數量與回應時間的關係如圖 4.4 所示。因為限制伺服器負載於固定範圍之內，回應時間隨使用者增加將會趨於固定值。以思考時間設定 1 秒為例，當使用者數量為 30 個時，伺服器負載已經超過 40%的設定值，允入控制機制開始拒絕部分 HTTP 請求，避免伺服器負載繼續增加。爾後隨使用者數目繼續增加，允入控制機制則會拒絕更多 HTTP 請求以限制伺服器負載於設定值之內，因此使用者數量高於 40 個之後，其回應時間皆為 5 秒左右。當思考時間為 5 秒與 10 秒的情況下，請求到達率隨著思考時間增加而下降，因此若要達到啟動允入控制機制的負載條件，其使用者數量必須較多時才會發生。於思考時間為 5 秒的情形，使用者數目必須到達 40 以上，允入控制機制才會開始拒絕使用者。而思考時間為 10 秒的情形，使用者數目則必須到達 60 以上才具相同的負載條件而拒絕請求。

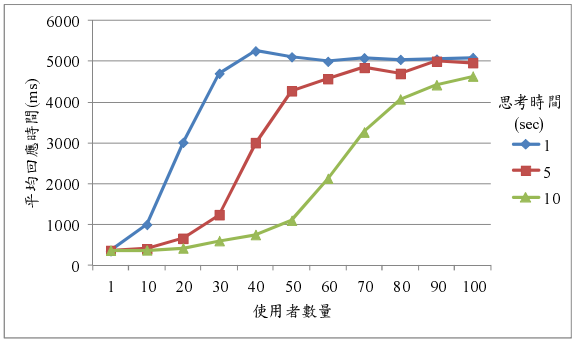

圖 4.4 使用允入控制機制回應時間與使用者關係

圖 4.5 為回應時間與伺服器負載的關係，隨著伺服器負載的增加，其回應時間也將增長，然而受到允入控制機制的作用，伺服器負載最大值會被限制在 40%以內，因此回應時間將可控制於期望範圍之中。同樣地，由圖 4.5 可得知回應時間僅與伺服器負載有關，而與思考時間無關。

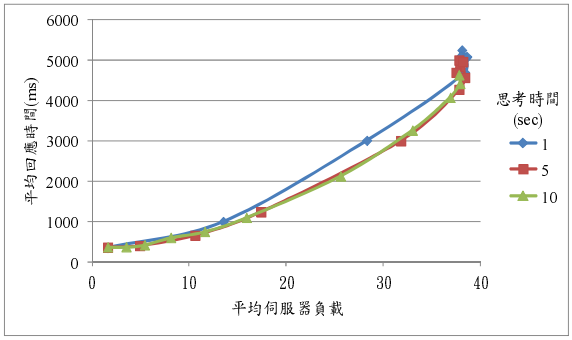

圖 4.5 使用允入控制機制回應時間與伺服器負載關係

使用允入控制前後的比較如圖 4.2~4.5 所示，結果顯示有下列差異：

(1) 使用允入控制機制前，在伺服器滿載的情形下，等待時間過長而造成請求逾時的發生，然而使用允入控制機制後，由於拒絕請求的立即回應，使用者未曾發生請求逾時的情形。

(2) 使用允入控制機制後，為限制伺服器負載的範圍，因此能夠處理請求的執行緒數目將會漸少，這使得佇列中等待的情形會提早於低負載時發生，而且佇列長度相對增加。

(3) 於 180 秒的測試期間，不論是否使用允入控制，請求成功的數目隨負載增加趨於定值，但使用允入控制後的回應時間會低於未使用時的情形。該請求成功的數目表示此系統於測試期間內能夠處理的最大吞吐量。

5. 結論與未來展望

網際網路與世界的複雜性造成網站容量規劃的困難。瀏覽人數突然增加造成網站過載，其結果將造成服務時間過長、請求逾時或連線失敗等問題，網站使用允入控制機制可以有效避免過載產生的問題。本論文使用 Apache 模組的方式實現允入控制機制，此方式使得現有的多層式網頁應用程式具有過載保護的功能，避免網站因為過載而造成服務時間增長。本論文所提出的允入控制策略可以限制伺服器負載情形與請求佇列長度，當網站處於過載情形之下，該策略使得網站的回應時間仍可維持設定值，因此服務品質可得到預期效果。

參考文獻

1. J. F. Jurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet (Third Edition)*, New York: Addison-Wesley, 2005, pp. 56-58.
2. What Is Web 2.0 - O'Reilly Media, http://oreilly.com/web2/archive/what-is-web-20.html, June 2009.
3. J. Farrell and G. S. Nezlek, “Rich Internet Applications: The Next Stage of Application Development,” *Proceedings of the 29th International Conference on Information Technology Interfaces*, Cavtat, Croatia, 2007, pp. 413-418.

4. M. Kihl and N. Widell, “Admission Control Schemes Guaranteeing Customer QoS in Commercial Web Sites,” *Proceedings of IFIP and IEEE Conference on Network Control and Engineering*, Paris, France, 2002, pp.305-316.
5. R. Mohan, J.R. Smith and C. S. Li, “Adapting Multimedia Internet Content for Universal Access,” *IEEE Transactions on Multimedia*, vol. 1, no. 1, 1999, pp. 104–114.
6. T. F. Abdelzaher and N. Bhatti, “Web Content Adaptation to Improve Server Overload Behavior,” *Computer Networks*, vol. 31, no. 11, 1999, pp. 1563-1577.
7. J. Cao, M. Andersson, C. Nyberg and M. Kihl, “Web Server Performance Modeling Using an M/G/1/K*PS Queue,” *Proceedings of the 10th International Conference on Telecommunications*, Papeete, Tahiti, 2003, pp. 1501-1506.
8. X. Liu, J. Heo and L. Sha, “Modeling 3-Tiered Web Applications,” *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Atlanta, Georgia, pp. 307-310.
9. D. Gross, J. F. Shortle, J. M. Thompson and C. M. Harris, *Fundamentals of Queueing Theory (Fourth Edition)*, New Jersey: John Wiley & Sons, 2008.
10. X. Liu, J. Heo, L. Sha and X. Zhu, “Queueing-Model-Based Adaptive Control of Multi-Tiered Web Applications,” *IEEE Transactions on Network and Service Management*, vol. 5, no. 3, 2008, pp. 157-167.
11. X. Liu, J. Heo, L. Sha and X. Zhu, “Adaptive Control of Multi-Tiered Web Applications Using Queueing Predictor,” *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium*, Vancouver, Canada, 2006, pp. 106-114.
12. A. Robertsson, B. Wittenmark, M. Kihl and M. Andersson, “Admission Control for Web Server Systems - Design and Experimental Evaluation,” *Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 531-536.
13. D. A. Menasce and V. A. F. Almeida, *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*, New Jersey: Prentice Hall PTR, 2000, p. 118.
14. D. A. Menasce and V. A. F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, New Jersey: Prentice Hall PTR, 2001, pp. 158-159.
15. Web Server Survey Archives - Netcraft, http://news.netcraft.com/archives/web_server_survey.html, June 2009.
16. N. Kew, *The Apache Modules Book: Application Development with Apache*, New Jersey: Prentice Hall PTR, 2007.
17. 張中慶、梁雪平，Apache 源代碼全景分析第 1 卷：體系結構與核心模塊，北京：電子工業出版社，2009.
18. 郭文生譯，Apache 技術手冊(第三版)，台北：歐萊禮，2003。
19. Zend Server, http://www.zend.com/en/products/server/, June 2009.
20. Moodle.org, http://moodle.org/, June 2009.