大學分發入學網路放榜系統

游象甫[ab], 曾黎明[b], 劉秋美[a], 王雅慈[a], 黃菁斌[a]
[a]國立中央大學電子計算機中心
[b]國立中央大學資訊工程研究所
{center3, center6, center11}@cc.ncu.edu.tw

摘要

中央大學於九十一及九十二年接受大學招生委員會聯合會委託辦理大學入學分發作業, 分發作業工作繁多, 可大分為行政及電腦自動化兩部分, 而電腦自動化又可分為分發入學條件輸入及簡章製作、登記生志願分發、網路放榜. 本論文特別討論其中網路放榜的過程及系統的設計, 希望我們學到的經驗可為其他類似應用的參考. 網路放榜系統需滿足下列條件－正確、安全、可靠、迅速、簡單. 為達成此目標, 我們考慮不同的情形, 使用多種技術, 同時設計本系統為兩層式架構, 利用網域名稱建立叢集運算. 依據所提之方法, 本文詳細說明系統實際運作方式, 同時測量使用者存取的狀況, 並分析效能. 根據統計, 本系統於九十二年八月九日被存取 11,412,505 次, 放榜的第一小時(即早上 8 至 9 點)被存取 2,931,369 次, 最大負載出現在當日早上 8 點 1 分至 2 分, 總共有 72,327 次存取. 前端網路查榜伺服器的負載峯值均出現在 7 點 56 分左右,且負載相當平衡, 而後端網路查榜伺服器也有類似結果. 此顯示以網域名稱建立叢集運算這種簡單策略已可達到負載平衡, 應用於瞬間大量存取的情況.

關鍵字: 大學入學, 網路放榜, 叢集運算

1 緒論

中央大學於九十一及九十二年接受大學招生委員會聯合會委託辦理大學入學分發作業, 分發作業工作繁多, 可大分為行政及電腦自動化兩部分, 行政工作主要負責聯絡各大學、高中及其他相關單位, 同時也需要與傳播媒體聯絡公佈有關分發的訊息.

電腦自動化可分為入學條件輸入及簡章製作、登記生志願分發、網路放榜. 入學條件包含各學系採計的學科考試成績、加權比例、高低標準及性別限制, 另外某些學系(如音樂或美術系)會加考術科, 因此其入學條件會包含術科成績, 但術科成績又會隨採計科目而有不同, 例如音樂系就會與考試的樂器有關, 因此入學條件非常複雜, 然而入學條件是分發正確的前題, 必須完全無誤. 以往入學條件是各校系以 MS Word 或簡單的網頁輸入, 再以人工轉成對應的分發條件, 往往需要花費大量人力, 反覆檢查. 為避免人工轉換的錯誤及節省人力, 本校電算中心開發校系條件輸入系統, 讓分發條件直接由各校系輸入, 同時用於產生簡章.

當登記生完成報名及繳交志願卡後, 由讀卡機將所填志願轉成數位資料, 利用分發程式與登記生成績, 進行分發作業. 為求正確, 本校電算中心由多位程式設計師分別撰寫不同的程式, 將分發結果以電子檔比對, 同時也與大考中心自行撰寫的分發程式輸出結果比對. 當一切正確無誤後, 會產生紙本榜單及供傳播媒體使用的電子檔.

隨著電腦網路的普及, 網路放榜變成必須提供的服務. 同時由於各校系錄取標準不同, 落點預測變得不可靠, 使用網路查榜遠比搜尋紙本榜單或平面媒體來得方便及快速. 本論文特別討論網路放榜的過程及系統的設計, 希望我們學到的經驗可為其他類似應用的參考. 網路放榜系統需滿足下列條件 – 正確、安全、 可靠、迅速、簡單. 為達成此目標, 我們考慮不同的情形, 使用多種技術, 同時設計本系統為兩層式架構, 利用網域名稱建立叢集運算. 依據所提之方法, 我們以開放授權程式碼實際建立網路放榜系統, 而硬體除前端四部採用雙 CPU Intel 伺服器外, 後端除一部採用雙 CPU Intel 伺服器外其他使用 31 部電腦教室的桌上型電腦, 以節省經費. 本文並測量使用者存取的狀況, 並分析系統效能. 實驗結果顯示以網域名稱分配請求的二層式叢集網站系統雖然原理相單簡單, 但已可在負載尖峯時平均分配使用者請求至各查榜網站, 足以應付放榜時瞬間大量存取的需求, 沒有遲緩或當機的現象, 順利放榜.

本文其餘內容如下: 第二節為相關研究; 第三節討論本系統的設計架構; 第四節描述實作及分析實驗數據; 最後是結論.

2 相關研究

本節介紹設計及發展大型(large scale)網站服務的相關研究. 1996 年奧林匹克運動會首次提供網站供使用者查詢, 其由 IBM 公司負責發展

及維運, 系統稱為 WOMplex [7], 由 60 部分佈在 4 個國家 5 個地點的網站所組成, 在奧運期間 19 天總共被點選 192,000,000 次. 本系統利用多種技術, 如網域名稱繞路及 IP aliases 將負載分佈於本地及本球的網站. 文章同時也介紹用於監測及管理網站的策略.

L. Cherkasova 提出 FLEX [4] 的方法, 以多部電腦主機(指硬體)同時提供多個網站(指軟體)服務, 希望能達到負載平衡, 同時能有效率的使用多部主機的記憶體. 主要原理是根據所收集的網路流量, 然後將主機分配給需要的網站. Clark [5] 提出以 multicast 的方式傳送網頁來降低網站的負載及網路的流量. Betz [1] 建立一個可擴充的網際網路股票資訊系統, 允許使用者可以電子郵件方式得知股票的價格, 其最大可同時接受 3 百萬個使用者請求. P. Felber, Chee-Yong Chan, M. Garofalakis, R. Rastogi 的研研 [6] 討論如何發展一個以內容繞路為主的架構用於 Web 的應用, 以處理日漸增加的 XML 文件. E. Casalicchio 和 S. Tucci [3] 的研究指出利用一個分配器來分配使用者請求的網站叢集系統是很強健可靠的, 同時指出固定的請求分配分式對 HTML 網頁或簡單的資料載搜尋已可有效處理突然變大的負載, 而只有對較大的瀏覽物件才需要使用動態的分配方式. Brewer [2] 撰文以實務的角度探討網際網路上一些超大型網站, 如 AOL, Yahoo 及 Geocities 如何提高可用度, 縮短系統停機時間及線上擴建硬體.

3 系統設計

由於網路放榜的重要性, 所於本系統至少需滿足五項條件 – 正確、安全、可靠、迅速、簡單.

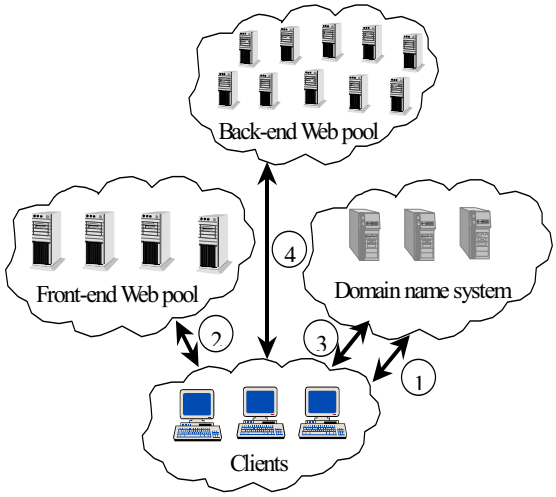
- 正確. 網路放榜系統最重要的就是必須確保程式與資料的正確性.
- 安全. 系統必須避免被入侵而竄改資料. 其次能抵擋來自於網路惡意的攻擊.
- 可靠. 系統應有容錯能力, 可以在部分伺服器、網路或電源發生異常時, 繼續提供服務.
- 迅速. 系統需能迅速的回應使用者的查詢. 最要的是面對大量查詢時, 反應時間也不會增加太多.
- 簡單. 由於榜單資料不能提前外流, 為求慎重, 大約是放榜前一日才拿到資料安裝於系統內, 再加上需要測試, 時間緊迫, 因此系統的安裝、測試及資料更新都必須簡單以節省時間.

為確保放榜的正確性, 我們撰寫網頁程式時考慮下列幾點: 第一, 儘量以程式直接測試網頁程式輸出的結果; 第二, 若榜單資訊為固定, 則儘量為 HTML 檔; 第三, 網頁輸出格式適於以人工檢查.

在網路放榜系統只需查詢且資料量不大的條件下, 本系統使用雜湊檔案進行榜單查詢, 和使用資料庫相比, 這種策略有多項優點. 首先, 因為用鍵值直接計算, 雜湊檔案的存取較資料庫快. 第二, 雜湊檔案易於安裝使用, 很多程式語言直接支援, 可節省系統建置時間; 反之資料庫系的安裝通常較費時, 且通常需要額外設定. 第三, 當資料發生損毀時, 雜湊檔案比資料庫容易復原. 第四, 現代的資料庫大都支援網路存取, 因此容易遭受從網路的惡意攻擊. 最後, 雜湊檔案可直接複製使用, 所以我們可以用記憶體作為磁碟空間(RAM Disk), 將其從硬碟拷貝至 RAM disk, 加快存取速度, 同時大幅降低硬碟損毀的可能性.

我們同時設計網路放榜系統為一個二層分散式架構如圖一, 使用者要網路查榜時, 首先必須向網域名稱稱伺服器(Domain name servers) 詢問網路查榜伺服器的 IP 位址(即 www.uac.edu.tw 的 IP), 如步驟 1. 接著使用者連線至前端網路查榜伺服器取得首頁, 如步驟 2. 當使用者開始查詢時會再向網域名稱稱伺服器查詢後端網路查榜伺服器 IP 位址, 如步驟 3. 最後使用者會被導向後端網路查榜伺服器真正進行查榜服務.

採用二層式架構的好處有兩點, 第一, 可以提高本系統的安全性, 在放榜前我們可以先開放前端網路查榜伺服器供查詢大學入學相關資訊, 但沒有榜單資料, 後端網路查榜伺服器則已安裝



圖一: 網路查榜系統架構圖

榜單資料, 但將其網路連線中斷, 如此就算前端網站被入侵資料也不會外洩. 其次, 二層式架構增加系統的可靠度, 由於前端網路查榜伺服器並不真正負責查榜服務, 只顯示首頁, 所以存取速度非常快, 可以應付瞬間高負載, 同時降低後端網路查榜伺服器的存取峯值.

本系統利用網域名稱伺服器分配各網站負載(Round-Robin Domain Name Dispatching)的理由有二. 第一, 這個方法相當簡單可靠, 容易測試. 第二, 網域名稱系統提供備援及快取的功能, 可提供系統容錯能力, 而對存取速度的影響不會太大, 同時自動分散主網域名稱伺服器負載. 表一是我們採取的方法以達成正確、安全、可靠、迅速、簡單的目標.

表一: 本系統對各項要求所採取的方法

	正確	安全	可靠	迅速	簡單
Web output tested automatically	V			V	
Web output tested manually	V				
Two-tier clustering architecture		V	V	V	
Round-robin domain name			V		V
Hash file		V	V	V	V
RAM disk			V	V	
Quick reboot				Y	
Dual power			V		
Dual network equipment		V	V		
Dual backbones		V	V		
Disable useless accounts and ports		V			
Stop useless daemons		V			

4 實作及數據分析

我們依據上述的架構實作網路放榜系統, 前端網路查榜伺服器由四部 Intel 雙 Xeon 2.4 CPU 的伺服器組成(如圖二), 記憶體均為 4GBytes, 網路卡速度為 100Mbps. 後端網路查榜伺服器由 1 部與上述相同的伺服器及 31 部個人電腦所組成(如圖三), 這些電腦原本位於電腦教室供上課實習, 並非新購置, 其硬體配備為 Intel Pentium 4 1.6G CPU 及 512MBytes 記憶體. 每部電腦的作業系統為 FreeBSD 4.8 [9], 網站軟體是 Apache 1.3.28 [8], CGI 採用的程式語言為 Perl 5.0 [10], Perl 支援許多雜湊檔案格式, 在此使用

圖二: 前端網路放榜伺服器(圖中最下面 4 部黑色電腦)

圖三: 後端網路放榜伺服器(圖中只顯示其中一列 4 部, 共有 8 列)

的是 UNIX DBM. 為確認查榜程式的正確性, 我們撰寫額外程式呼叫查榜的網頁程式, 並將輸出結果與原始放榜資料比較. 同時也設定每部電腦開機時, 會建立 96MBytes 的記憶體檔案系統(Memory File System, MFS), 並從硬碟複製含榜單資料的雜湊檔案到該檔案系統中.

由於放榜正值夏季, 為桃園地區用電高峰,

圖四: 放榜前 10 秒

圖五: 查榜時的前端伺服器(www.uac.edu.tw)

圖六: 查榜時的後端伺服器(announce.uac.edu.tw)

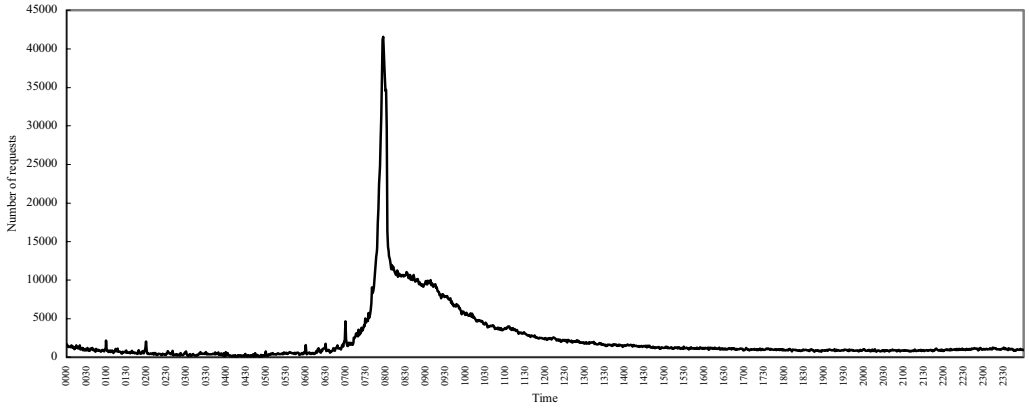
為確保電力不受影響, 前端網路查榜伺服器均為雙電源, 且分別連接至不同的不斷電系統, 電力中斷時可以發電機供電. 由於後端網路查榜伺服器除 1 部外其餘均為單電源供應器, 所以我們將其平均接於兩個不斷電系統, 以防萬一.

為避免放榜時, 受到中央大學校內網路的影響或其網路流量影響到中央大學校內網路, 本系統直接連接桃園區網中心的路由器(CISCO 6509), 同時中央大學至教育部間的線路分別由中華電信及東森固網提供 6 條 1Gbps 線路作為備援. 前端及後端網路查榜伺服器的 IP 位址均為 163.28.48.X, 而主要的網域名稱伺服器的 IP 亦為 163.28.48.X, 其有三部備援, 兩部位於中央大學校內(其 IP 分別為 163.28.48.X 及 140.115.1.X), 另一部位於教育部電算中心. 前端網路查榜伺服器的網域名稱均為 www.uac.edu.tw, 而後端網路查榜伺服器的網域名稱為 announce.uac.edu.tw, 由於聯合分發委員會公告的網站位址為 www.uac.edu.tw, 所以使用者查榜時會先至前端網路查榜伺服器, 然後被轉向至 announce.uac.edu.tw. 由於原有的 www.uac.edu.tw 其 IP 為 140.115.1.4, 為了讓網域名稱的快取資料失效, 新的網域名稱資料於放榜前數日被登錄, 所以前端網路查榜伺服器先開始運作, 但使用者瀏覽網頁時, 只能看到原有大學入學分發的資訊, 如圖四, 並無榜單資料. 等到 8 月 8 日我們才在隔離的網路環境下, 將榜單資料安裝於後端網路查榜伺服器. 8 月 9 日上午將近 8 點時我們才將後端網路查榜伺服器連上 TANet, 前端及後端網路查榜伺服器中的 CGI 程式會在倒數計時完畢後, 於 8 點準時顯示查詢網頁開始放榜, 如圖五及圖六.

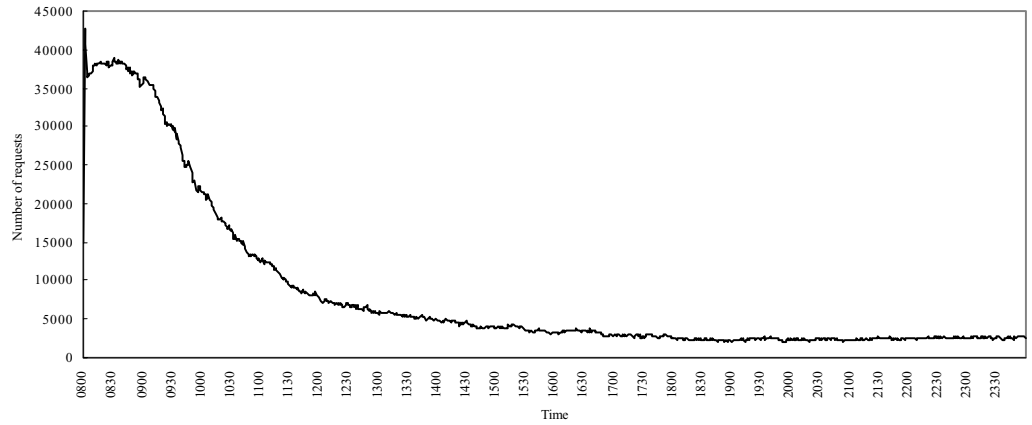
圖七顯示前端網路查榜伺服器於 8 月 9 日被存取的狀況, 總共被存取 3,332,565 次,峯值出現在早上 7 點 56 至 57 分, 共有 41,549 次存取. 在 8 點後由於使用者的存取均被轉向至後端網路查榜伺服器, 所以存取次數從 34,649 次迅速下降, 同時隨著時間過去, 網站的熱門程度也下降, 存取首頁人數也明顯減少.

圖八顯示後端網路查榜伺服器於 8 月 9 日被存取的狀況, 總共被存取 8,079,940 次, 因為從 8 點開始放榜,所以本圖從 8 點才有資料, 峯值出現在早上 8 點 1 至 2 分, 共有 42,616 次存取. 後端網路查榜伺服器一直相當熱門, 在 10 點後存取次數才減為一半左右, 隨著時間過去, 網站的熱門程度也下降, 8 小時後(即下午 4 點時), 存取次數只剩大約為峯值的十分之一.

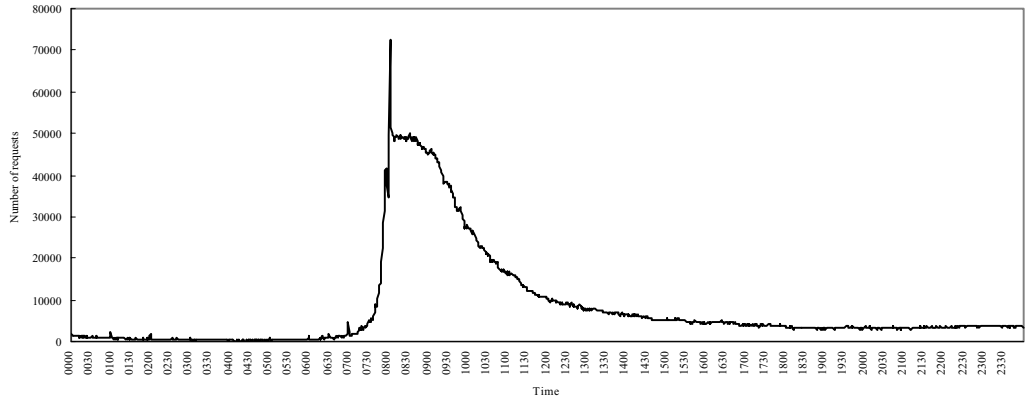
合併前端及後端網路放榜伺服器的存取次數, 我們可以得到 8 月 9 日總共被存取的次數為 11,412,505. 其分佈如圖九, 峯值出現在早上 8 點 1 至 2 分, 共有 72,327 次存取. 與前面所量測結果相比, 二層叢集式架構確實可造成使用者存取時間遞移, 而有效地將瞬間負載峯值分配於前後兩端的伺服器上. 我們以較高速的電腦作為前端, 但僅提供放榜查詢首頁(如圖五), 真正的資料查詢在後端, 其中的理由在於讓大量使用者可以迅



圖七: 前端網路查榜伺服器於 8 月 9 日被存取的狀況

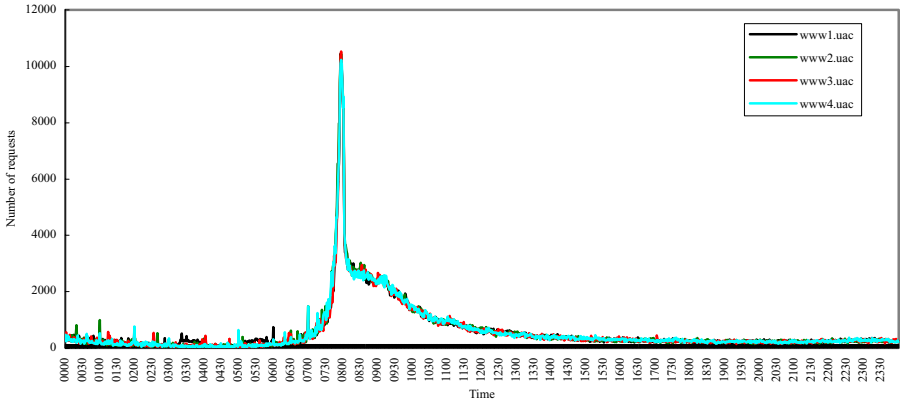


圖八: 後端網路查榜伺服器於 8 月 9 日被存取的狀況



圖九: 網路放榜系統於 8 月 9 日被存取的狀況

速地看到網頁, 覺得安心, 如果首頁出現的很慢,

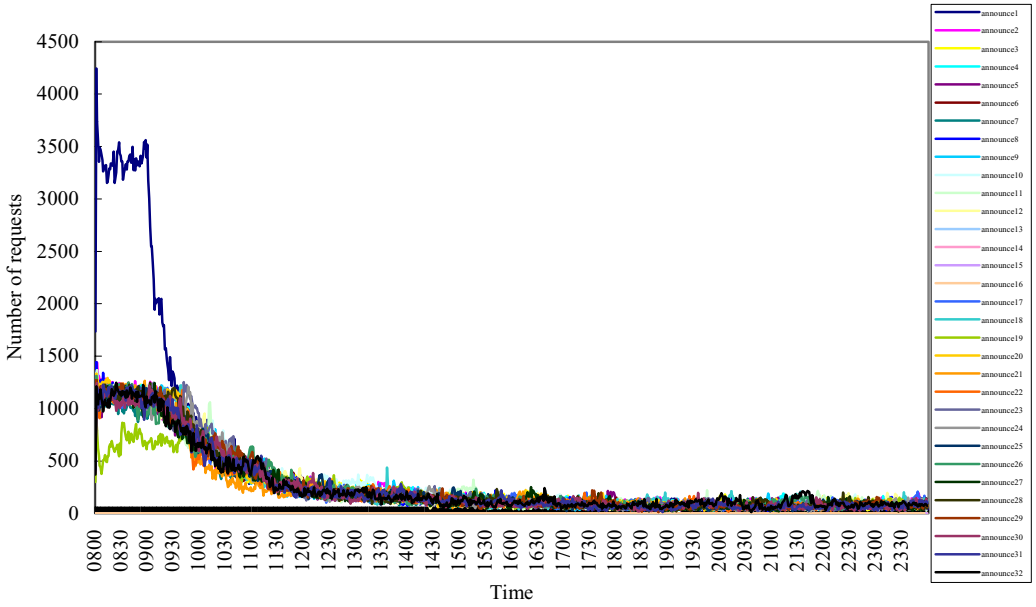


圖十: 前端網路查榜伺服器於 8 月 9 日各別被存取的狀況

使用者很可能會一直按瀏覽器的重新整理, 因而產生許多額外的存取, 讓系統負載更重. 同時由於使用者在看到首頁後, 必須輸入准考證號才能進行查詢(如圖五), 這段時間也可以讓存取以較慢的速度進入後端的伺服器.

圖十顯示 8 月 9 日前端網路查榜伺服器各別的存取分佈, 四部伺服器的負載峯值均出現在 7 點 56 分左右, 同時我們可發現其存取次數的分佈曲線大約重合, 而在峯值處更加相近, 反應出以網域名稱伺服器作存取分配在高負載時並沒有發生負載不平衡的現象.

圖十一顯示 8 月 9 日後端網路查榜伺服器各別的存取分佈, 32 部伺服器的負載峯值有一半出現在 8 點 1 至 2 分, 其餘大都也出現 8 點 5 分前. 同時可發現其存取次數的分佈曲線雖然大致相似, 以網域名稱伺服器仍可平均分配後端伺服器的負載, 但我們注意到這圖裡的曲線沒有如圖十般重合, 主要原因是為減輕網域名稱伺服器負載, 本系統設計為當使用者被從前端伺服器轉向某部後端伺服器後, 其發出的任何查詢請求均只會在該部電腦執行, 不再以網域名稱轉向, 所以每部後端伺服器的存取會隨著持續查榜的使用



圖十一: 後端網路查榜伺服器於 8 月 9 日各別被存取的狀況

者多寡而有所不同, 存取曲線不會如圖十般密合, 但因為所有存取曲線的線形類似, 所以也反應大出部分使用者查詢的行為模式大致相同.

最後我們討論圖十一中兩條明顯和其他曲線不同的曲線, 其中一條低於其他曲線, 而另一條則遠高於其他曲線. 首先討論較低曲線, 在放榜後數日我們無意中發現較低曲線所代表的那部電腦其網路連接線有很高的雜訊, 在傳送大量資料時會有超過 20%的遺失率, 參照圖中的曲線, 我們可以發現在非常熱門的時段(8 點至 9 點)其存取次數大都不大於每分鐘 800 次, 而其他後端伺服器大都被存取 1,100 次以上. 這代表當存取次數大於 800 次時, 其封包遺失率會使得其無法再接受新的存取或是使用者的請求會逾時. 其次, 本節一開始曾描述有一部後端伺服器使用較快的雙 CPU 電腦, 較高曲線代表的就是該部主機的存取分佈, 其存取次數約為其他電腦的 3 倍, 我們曾經假設一個可能的原因是因為該部電腦較快, 所以使用者會不斷查詢, 造成其存取次數較多, 但經統計該部伺服器的使用者來自於 7269 個 IP, 而其他後端伺服器使用者平均只來自於 2325 個 IP, 也多接近 3 倍, 所以其存取次數多是因為使用者多而非每個使用者存取的次數多, 這表示我們的假設不成立. 目前我們對這個現象尚無合理的解釋.

5 結論

中央大學於九十一及九十二年接受大學招生委員會聯合會委託辦理大學入學分發作業, 本論文特別討論其中網路放榜的過程, 作為其他類似應用的參考. 網路放榜系統需滿足下列條件 – 正確、安全、可靠、迅速、簡單. 為達成此目標, 我們設計本系統為兩層式架構, 利用網域名稱建立叢集運算以應付瞬間大量存取, 同時也討論多種方法(如表一)使能達成上述需求. 依據所提之方法, 本文詳細說明系統實際運作方式, 同時測量使用者存取的狀況, 並分析效能. 根據統計, 本系統於九十二年八月九日被存取 11,412,505 次, 放榜的第一小時(即早上 8 至 9 點)被存取 2,931,369 次, 最大負載出現在當日早上 8 點 1 分至 2 分, 總共有 72,327 次存取. 前端網路查榜伺服器的負載峯值均出現在 7 點 56 分左右, 且負載相當平衡, 而後端網路查榜伺服器也有類似結果. 此顯示以網域名稱建立叢集運算這種簡單的策略可達到負載平衡, 應用於瞬間大量存取的情況.

參考文獻

[1] K. Betz, “A scalable stock Web service,” in Proceedings of International Workshops on Parallel Processing, pp. 145-150, Aug. 2000.

[2] E.A. Brewer, “Lessons from giant-scale services,” IEEE Internet Computing, Volume: 5 Issue: 4, pp. 46-55, July-Aug. 2001.

[3] E. Casalicchio, S. Tucci, “Static and dynamic scheduling algorithms for scalable Web server farm,” in Proceedings of the 9th Euromicro Workshop on Parallel and Distributed Processing, pp. 369-376, Feb. 2001.

[4] L. Cherkasova, “FLEX: load balancing and management strategy for scalable Web hosting service,” in Proceedings of the 15th IEEE Symposium on Computers and Communications, (ISCC 2000), Antibes-Juan les Pins, France, pp. 8-13, July.

[5] R.J. Clark, and M.H. Ammar, “Providing scalable Web service using multicast delivery,” Second International Workshop on Services in Distributed and Networked Environments, pp. 19-26, June 1995.

[6] P. Felber, Chee-Yong Chan, M. Garofalakis, R. Rastogi, “Scalable filtering of XML data for Web services,” IEEE Internet Computing, Volume: 7, Issue: 1, pp. 49-57, Jan.-Feb. 2003.

[7] German Goldszmidt, and Andy Stanford-Clark, “Load Distribution for Scalable Web Servers: Summer Olympics 1996 - A Case Study,” in Proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Sydney, Australia, October 1997.

[8] http://www.apache.org/

[9] http://www.freebsd.org/

[10] http://www.perl.com/