Development of a Cluster System with DynamicLoading Balance for Video Streaming
具視訊串流動態負載平衡之叢集系統開發

蔡智強 教授
國立中興大學電機工程學系
jichiangt@nchu.edu.tw

陳悅真
國立中興大學電機工程學系
yueh_chen16@ares.ee.nchu.edu.tw

王駿修
國立中興大學電機工程學系
Joeiy0429@ares.ee.nchu.edu.tw

李健旭
國立中興大學電機工程學系
Stanley.simpon@ares.ee.nchu.edu.tw

憑忠信
國立中興大學電機工程學系
copam@ares.ee.nchu.edu.tw

周世洪
國立中興大學電機工程學系
seahomjkff@yahoo.com.tw

摘要（Abstract）

網際網路質與量高倍數的需求成長已經使得伺服器需求及負擔愈趨繁重，將單一伺服器作為服務的方式早已經不能夠滿足成長需求。為了能夠有效率地提昇穩定的服務品質，大型網站早已使用了大量伺服器叢集系統來解決傳統單一伺服器效能擴充問題，但是隨之而來的問題是如何根據實際伺服器負載狀況來進行負載分配，使其能不間斷的提供服務，並且考量低成本高效能以及系統容錯功能，更是最近這幾年很熱門的研究課題。

在本篇論文裡，我們利用了Linux作業系統的NAT功能結合封包改寫技術與動態負載平衡演算法，完成了軟體核心模組的實作。此軟體模組的優點可以在IP層進行封包遞送，並且可以依據叢集系統後端伺服器實際負載狀況進行負載分配。我們也將此軟體模組與視訊串流軟體作了整合，使整個叢集系統具備視訊串流的服務，期望未來能夠成為功能更為完善的視訊隨選叢集系統。

關鍵詞：Linux、ONEIP、負載平衡、叢集系統

一・緒論

1.1 研究背景與動機

目前寬頻網路服務普及，多媒體服務使用量也隨之大幅增加，例如：視訊隨選(Video On Demand)，互動式視訊隨選系統等。因此，使用多伺服器應付大量服務需求相對成為解決大量資訊處理的方法之一，具叢集功能之軟硬體研究亦受到重視。此類產品不僅提供更快的速度以適應急劇增長的網路流量，而且提高其效能是其關鍵之一。叢集系統之性能基礎在於服務器轉送與排程之能力，其中必備之技術在於處理伺服器間位址模擬共用通訊之基礎，以提供使用者及伺服器間無接縫式 (Seamless)的服務連結。

1.2 研究目的

叢集系統的負載平衡機制，以目前的派送分配技術有四種負載平衡 (Load Balance)的解決方法[1]：

- 以DNS為基礎的負載平衡（DNS based）
- 以用戶端為基礎的負載平衡（User based）
- 以派送主機為基礎的負載平衡 （Dispatcher based）
- 以伺服器為基礎的負載平衡（Server based）

本文採用的是第三種方法，希望透過修改封包位址的方式來完成封包的轉送，並且使用動態負載分配演算法，根據真實系統負載完成分散流量的工作，達到負載平衡的目標。

二・概論

2.1 叢集系統的架構與特性

叢集式系統是一群伺服主機所組合而成的，基本上叢集式系統擁有六大優點：

1・High Reliability–高可靠度
2・High Performance–高效能
3・High Scalability–高擴充性
4・Low Cost–低成本
5・Fault Tolerance–容錯性

6．Load Balancing–負載平衡機制。

其中以負載平衡（Load Balancing）與容錯（Fault Tolerance）為叢集式系統最主要的功能。利用負載平衡技術可以把使用者的服務要求由一台電腦來分散給後端伺服主機來服務，使得叢集式系統可以模擬成一台高效能的伺服主機，而如何使得叢集式系統能夠發揮最大的效能，則有賴於排程演算法是否能夠平衡後端伺服主機的負荷，使網站系統能夠接受更多的服務連線。而容錯的功能是防止分派使用者服務要求的電腦主機發生當機，因而造成整個叢集式系統癱瘓，因此，當主要的伺服主機無法分派使用者的服務要求至後端伺服主機時，備援伺服主機就會啟動並取代主要伺服主機，使得叢集式系統能夠繼續的運作。

以工作分派伺服主機平衡負而言，如圖2.1.1所示，叢集式系統中對外只公佈一個網域名稱給大眾知道，而這一個網域名稱只會對應一個網路位址作為對外連入工作分派伺服主機之用，當工作分派伺服主機接收到使用者的服務需求時，工作分派伺服主機便會決定某一台後端伺服主機來處理。其優點為實作一台工作分派機制相對於其它方法簡單，不需要修改客戶端的程式，也不需要修改路由器，且與網路現有的協定相容，又可實作成硬體設備來提高執行效率，所以大部份的叢集式系統均採用此架構為主。

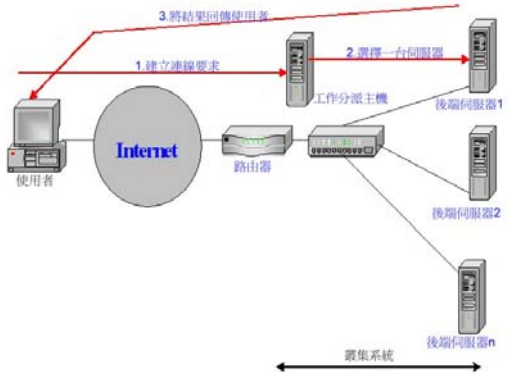

圖2.1.1工作分派主機架構

而分派主機在分派工作到後端伺服器時，所依據的即是叢集式系統之排程法，排程演算法要使得分派主機能夠正確的選擇最適當的後端伺服主機來提供服務，並且使得各後端伺服主機的負載能夠共享，以避免在負載尖峰時，工作集中在某些後端伺服主機上而造成某些伺服主機當機。但是要設計一個好的排程演算法時，首先會面臨到兩個考慮因素，其中每一使用者所發出服務請求不相同，因此同一伺服主機來完成使用者的服務請求的時間也均不相同，另一個要考慮的因素為後端伺服主機群的軟體或是硬體等級不相同時所造成的差異，這些原因將造成設計一個排程演算法的複雜度的增加。

2.2 Linux Virtual Server之架構

LVS Project[2]是實作在Linux作業系統的叢集式系統，其目地在建立一個高度可擴充性（Highly scalability）、高度可用性（Highly Availability）、可延伸性（Flexibility）、可管理性（Manageability）及成本效益（Cost-Effectiveness）的叢集式系統，使得叢集式系統可以模擬成一個高效能的伺服主機。透過使用者的連線請求，LVS會依照系統管理者最初設定LVS的排程法來決定，將請求連線封包轉送給後端伺服主機，並記錄此一連線資料及轉送至後端伺服器主機的網路位址。在LVS轉送封包給後端伺服主機的技術分別為NAT（Network Address Translation）、IP Tunneling及Direct Routing[3]，這些轉送封包的技術是利用封裝封包技術[4]（Packet Encapsulation Technique）或者利用改寫封包[4]（Rewriting Packet）的技巧來實作封包轉送的機制。

目前在Linux核心2.4版的LVS project中提供了八種平衡負載的演算法[2]供網站系統管理者來使用：

1．輪流（Round Robin, RR）排程法
2．權重式輪流（Weighted Round Robin, WRR）排程法
3．最少連線數（Least Connections，LC）排程法
4．權重式最少連線（Weighted Least Connections, WLC）排程法
5．臨近性最少連線（Locality-Based Least Connections, LBLC）排程法
6．臨近性集群最少連線（Locality-Based Least Connections with Replication, LBLCR）排程法
7．根據目的地位址雜湊（Destination Hashing）排程法
8．根據來源位址雜湊（Source Hashing）排程法

三．相關技術及產品設計

同樣使用分派機制的相關計畫有MagicRouter[9]、Cisco公司的Local Director[10]、IBM公司的Network Dispatcher[11]與Resonate公司的Central Dispatch[12]。除Local Director是硬體解決方案之外，其餘三種都是以軟體實作。當然尚有其它研究及實作產品，本文只選取其中較具代表性為研究參考。

MagicRouter[9]是實作在Linux user level上，並非針對http服務而設計，與本文的semaless mode一樣，專注於封包轉送的機制。MagicRouter在收到使用者端要求時先自後端伺服器群擇一，替換掉封包內位址header的目的地位址並重新計算checksum後，再轉遞給選定的伺服器處理。當封包要送回給使用者時，再行經MagicRouter，此時反向將選定的伺服器位址換回原公開位址。這種作法封包進出都要紿

過分派主機且封包必須被改寫兩次，減緩了分派主機的處理速度。Cisco的LocalDirector[10]作法與MagicRouter相同，但是因為與交換式集線器整合做成硬體，因此可以大幅提升轉送速率到約每秒40000個連線要求。

IBM的Network Dispatcher[11]每秒約能處理2000個連線要求，是一個廣域網路（WAN）架構的雙層式解法，其在內部網路（LAN）上的作法是所有伺服器群都共用且只用同一位址，但是都各自關閉ARP回應，只留分派主機回應ARP查詢。分派主機同樣在選定好後端伺服器後，將MAC位址改寫後把封包送給伺服器。這個作法的限制是分派主機是專屬的。

Resonate公司的Central Dispath[12]產品，每秒約可處理6000個連線，可達到n部伺服器，每台都能當成分派主機，應該是目前所有商業軟體解決方案中表現與效能最好的一個，但問題是對於RAM的需求非常高，高達200MB，本文的設計模式不需要如此高的記憶體，只需要核心約佔幾十K空間放一個hash table而已。以分派主機為基礎的解決方案最大問題是分派主機可能成為瓶頸而當機，而網站系統當機後復原時間的長短不但會影響服務的可靠性也會影響電子商務的進行。因此也研究了Linux上的High Availability解決方案（HA），如Linux High Availability Project、Linux Virtual 伺服器、TurboLinux的Turbocluster[13]伺服器。總結這些作法，都是以數部分派主機構成群集，彼此監測，一旦主要分派主機當機，可隨時接手備援。

四．系統設計與架構

4.1 系統架構

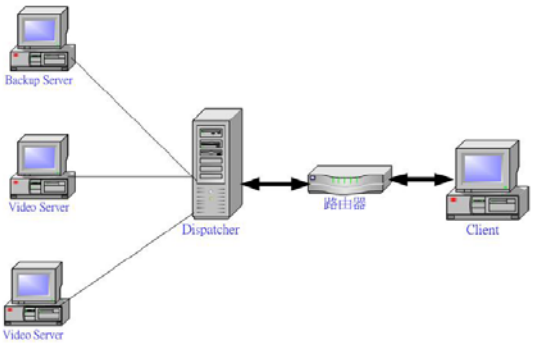

圖4.1.1系統架構

由圖4.1.1，整個叢集系統概觀分為三種伺服程式：

1．分派主機、備援主機與一般視訊伺服器群。其中分派主機內包含了以下的功能單元：

（1） 核心模組：負責修改封包欄位、動態排程、連線管理，與轉送封包。

（2） 負載搜集程式：負責在User Mode下接收由Video Server傳送過來的系統負載資訊，並將這些資訊傳送至核心模組內進行動態負載分配。

2．備援主機內包含以下的功能單元：

（1） 與分派主機相同的核心模組

（2） 隨時探測分派主機生命狀態的deamon，當這個deamon偵測到分派主機當機的時候，必須自動將系統設定值轉換成與dispatcher相同，並且改變預設的閘道，使得封包經由備用的路徑對外傳送出去。

3．位於後端的視訊伺服器群則具備下列的功能單元：

（1） 驗証使用者身份的伺服程式

（2） 播放視訊串流的Video Lan軟體[18]。

（3） 透過socket連線，每隔一定週期對分派工作主機傳遞系統資訊的程式。

4.2基本封包轉送機制設計

在解釋系統設計原理之前，先介紹網路位址轉換(Network Address Translate)的技術。所謂的網路位址轉換，就是將ipv4所提供的私有IP位址（例:192.168.0.1）進行適當的轉換，讓在私人網段上的電腦，也可以透過NAT的方式來跟網際網路上其它的電腦來作溝通，而不需使用公開IP位址(public IP address)。

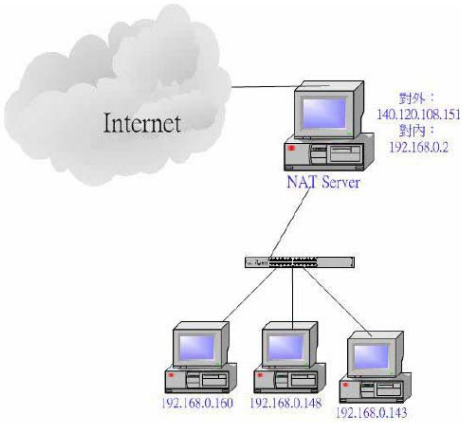

圖4.2.1 NAT示意圖

我們可以看到圖4.2.1中的NAT Server有兩個IP位址，一個是140.120.108.151 (Public IP)是用來與外部網際網路溝通的位址，而192.168.0.2 (private IP)則是與內部區域溝通的位址。假設我們今天要用來上網的電腦其IP address為圖中的192.168.0.143，這個樣子並不能直接由LAN連接到Internet，所以必須先將閘道位址設定成NAT Server的對內網路位址（192.168.0.2），如此一來192.168.0.143的對外連線即會透過NAT Server。當NAT Server接收到來自192.168.0.143的封包後必須將此IP address修改成

“140.120.108.151“才可以與外部網路溝通，這就是所謂的SNAT (Source NAT)，也就是說該封包只有在輸出時才會進行來源位址的轉換。相對的，對於由外部介面輸入的封包，NAT伺服器所要做的就是所謂的DNAT (Destination NAT)的工作，若先前建立連線的主機是192.168.0.143這部電腦，則封包的目的位址將會由140.120.108.151修改成192.168.0.143。

我們分派主機上的封包轉送機制也是利用NAT的原理來進行負載的分配，我們會先根據流進分派主機封包的ＴＣＰ header的埠號（Port Number）與IP標頭的IP位址欄位來判別這個封包是否必須傳遞到後端伺服器來進行負載分配。若是如此的話，則必須修改IP標頭中的IP位址欄位、替這個後端伺服器的IP位址來重新尋找路由，最後再進行IP標頭的檢查和(Check Sum)的計算。對於來自後端伺服器的回應封包也是如此，修改IP標頭中的來源位址欄位，再進行檢查和的計算即可完成後端伺服器與客戶端連線的建立。

4.3 環境建立與位址分配

在整個叢集系統設計的第一步就是先將系統分派器架構成NAT伺服器，在這個步驟裡所要完成的工作包含了[19]：

- 在同一塊網路卡上新增一個虛擬網路介面，作為內部私有網路的開道介面。
- 開啟Linux作業系統的封包轉遞功能
- 設定封包過慮系統iptables的網路位址轉換功能(NAT)

在Linux作業系統裡若要新增虛擬網路介面，可以使用內建指令“ifconfig“來進行設定。 由於Linux作業系統的核心預設情況並無開啟封包轉遞的功能，所以我們必須將系統核心中的IP_FORWARD功能打開，使用sysctl這個指令將/proc/sys/net/ipv4/ip_forward檔案的值設定為1。

最後則是使用封包過慮工具iptables替IP封包進行偽裝以便進行轉送。

完成了上述三個步驟以後，分派器則成為具有NAT功才的伺服器，接下來就是為叢集系統的後端伺服器指令私有IP位址並且設定預設閘道。設我們由上述的分派器設定得知來自於網路位址192.168.0.1～192.168.0.254的位址皆可以進行IP位址偽裝。所以後端伺服器的位址必須設定為192.168.0.1～192.168.0.254。而這些後端伺服器的預設閘道皆設定成分派器的虛擬網路介面。

4.4 封包遞送

當一個分派器收到一個目的地位址為”p”的TCP封包時，為了要將之傳送到叢集系統的後端伺服器，所以必須先修改IP標頭的目的地位址，將它改為後端伺服器位址”s”。當封包中的標頭欄位有所改變動時，則必須將檢查和重新計算，再填入封包中的檢查和欄位，避免目的地伺服器接收到封包後將之丟棄。最後再將封包傳送到後端伺服器處理。

4.5 動態排程機制

為了使分派器能夠根據後端伺服器實際的負載狀況來分散流量，必須採取動態排程的機制，此機制所考慮的幾個重要因素包括：

- CPU使用率
- 記憶體使用率
- 後端伺服器的回應時間

當核心模組掛載於分派器中，為了能夠方便取得後端伺服器的負載資訊，所以利用Client/Server架構的方式，使得後端伺服器每隔一定的週期，傳遞自身的系統負載資訊（CPU使用率、記憶使用率）到分派器。分派器主機也開啟了一個專門搜集負載資訊的Service，除了接收後端伺服器所傳遞過來的系統負載資訊之外，同時也為每一個後端伺服器產生個別的執行緒來進行回應時間的測量，當取得了三個最重要的負載參數時，再透過socket的方式將它們傳遞到核心模組內，以便進行伺服器優先權的排序，最後再將流量分散到負載較輕的伺服器上。

一開始在每部後端伺服器上都會執行一個搜集本機CPU使用率[23]、記憶體使用率的程式。為了獲得本機的CPU使用率，我們所使用的方法是先讀取/proc/stat[24]這個由核心所產生的檔案，其內部包含了我們所需要一些參數，例如：CPU已經閒置了多少clock、CPU工作在使用者模式下使用了多少clock、CPU工作在核心模式下使用了多少clock…等等。倘若現在每隔5秒要計算一次本機CPU使用率，首先先假設目前CPU工作於使用者模式所使用掉的clock數目為cpu_user，而5秒後CPU工作於使用者模式所使用掉的clock數目為 5_cpu_user；再假設目前CPU工作於使用者模式、核心模式、閒置模式所使用掉的clock數目加總為total_cpu，而5秒後的clock數目加總為5_total_cpu，最後的CPU使用率的計算為：

$$\text{CPU 使用率} = \frac{\text{5\_cpu\_user-cpu\_user}}{\text{5\_total\_cpu-total\_cpu}} \times 100\%$$

關於記憶體使用率的取得則透過sysinfo()[24]這個system call來取得RAM的大小以及目前還剩餘多少RAM沒有被使用，假設：總共RAM的大小為total_ram，尚未使用的RAM的大小為free_ram，則最後的記憶體使用率的計算為：

$$\text{記憶體使用率} = \frac{\text{Total\_ram-free\_ram}}{\text{Total\_ram}} \times 100\%$$

回應時間的計算則是經由分派器的伺服端程式來完成，一旦後端伺服器透過socket將系統資訊傳

遞到伺服器端，伺服器端處理該連線的process則利用POSIX函式庫所提供的createthread()[24]產生一條執行緒(Thread)來對該後端伺服器進行回應時間的探測。而回應時間的量測是在時間點A利用發送icmp echo封包到後端伺服器，藉由在時間點B接收到icmp reply封包來計算得到，則回應時間的計算為：B-A

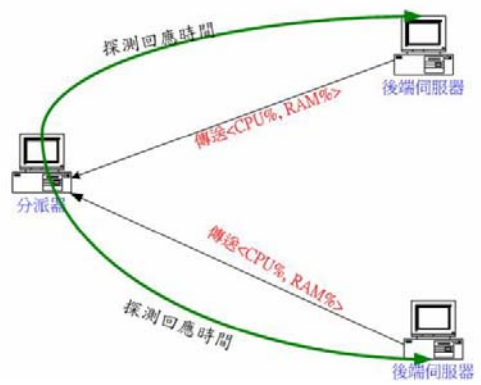

圖4.5.1 負載資訊取得的示意圖

當分派器上的伺服程式取得CPU使用率，記憶體使用率、回應時間之後則透過系統呼叫來傳遞負載資訊到核心模組內，再由核心模組將三個參數作加總的動作，加總後即可以得到目前後端伺服器的負載值，接下來再將這些負載值進行排序的動作，就可以決定目前負載最輕的伺服器的IP位址,也就是優先被分配流量的後端伺服器。隨著每台後端伺服器的負載狀況的不同，即可以完成動態負載平衡的實作。

4.6 連線記錄管理

伴隨著動態負載分配所帶來的問題就是連線管理，由於每台後端伺服器的負載值不斷變化，造成優先權值也隨之變化，所以必須將每個連線記錄起來，以避免因為伺服器優先權變化而造成連線失敗。由於連線的動作是非常頻繁的，在核心模組中我們使用快取記憶體(Cache Memory)來替同一條連線配置記憶體，接著再搭配使用雜湊表格（Hash Table）來儲存該連線的位址。這樣的好處除了增加連線處理的速度，也大幅的提升了軟體執行效率。

在核心模組初始化期間,由於雜湊表並不需要頻繁地進行配置/釋放記憶體，所以先利用vmalloc()在核心模組內配置一塊虛擬位址(virtual address)連續的空間來儲存連線雜湊表(connection hash table)，接著使用kmem_cache_create()產生新的快取記憶體。[25][26]

每當有封包流入分派主機內，立刻由封包的IP標頭與TCP標頭取出協定（Protocol）、來源端IP位址、來源端埠號這三個參數，將這三個參數傳入雜湊函數（Hash Function）[27]內以取得雜湊鍵值（key value），最後再將這個鍵值傳遞到連線雜湊表內，將已建立的連線取出，如果該連線存在的話，則可根據該連線的目地位址是哪一台後端伺服器來決定封包的流向。倘若該連線不存在的話，則呼叫kmem_cache_alloc()函式配置“連線物件“至快取記憶體內。再根據該連線物件的協定，客戶端IP位址，客戶端埠號三個參數進行雜湊，將連線物件的位址存放至連線雜湊表內。 每當有封包從後端伺服器轉送至客戶端時，則是反其道而行，將封包內的協定、目的端IP位址、目的端埠號取出，取得雜湊鍵值以後再將之傳入雜湊表格內取出連線物件，進行後續的處理。

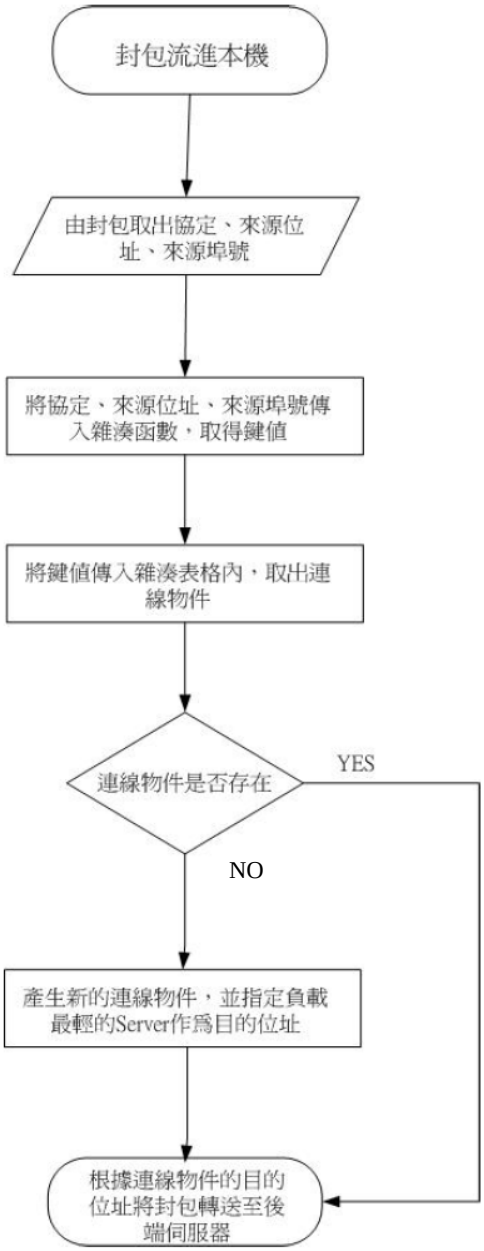

圖4.6.1 封包流進本機內的連線處理

4.7 備援機制

對於一個叢集系統來說，其最重要的功能除了擁有分散負載的功能之外，還有一個相當重要的功能就是「容錯」。許多提供網際網路服務的公司，倘若叢集系統中的某一部份節點發生故障，而系統中卻沒有即時處理錯誤的能力，如此一來有可能造成難以估計的損失。

由於我們整個叢集系統所採用的是"分派主機"的架構，所以來自客戶端的所有連線封包均會經過分派工作主機，此種架構的缺點是容易形成網路瓶頸，且一旦分派工作主機發生當機時，其整個叢集系統將無法發揮它的作用。所以關於此種架構的備援機制設計，將著重於備援主機的幾個部份：探測分派主機的生命、變更系統環境設定、重新尋找備用路徑作為路由。

為了探測分派主機是否發生當機，備援主機必須要能夠儘快地得知分派主機是否發生緊急情況。所以在備援主機與分派主機之間使用了頻外資料（Out-of-Band Data）[21]的傳遞，加速了緊急訊息的傳遞。備援主機每隔一定週期對分派主機傳送頻外資料，假設這個週期為5秒，而分派主機收到頻外資料，也對備援主機寫入頻外資料，若備援主機已經超過5秒沒有收到頻外資料，則必須自動進行系統環境變更，其中的動作包括：

- 利用ioctl()系統呼叫與SIOCSIFADDR旗標進行介面位址修改，以及產生與內部私有網路溝通的虛擬介面。
- 透過system()函式使用Linux作業系統所提供的工具指令執行核心模組的掛載動作，以及NAT功能的開啟。
- 假設叢集系統有備用路徑可以選擇的情況下，則透過system()函式使用Linux作業系統所提供的工具指令修改預設的閘道器位址，使用備用的路由進行封包傳遞。

4.8 視訊串流播放

為了達到基本的視訊串流播放功能，這一部份利用了現成軟體VideoLan來完成。關於視訊串流與負載平衡叢集系統的結合，則是先將VideoLan軟體安裝於每一台後端伺服器，使它們擁有能夠播放視訊串流能力的"Video Server"，接下來再設計一個伺服端的使用者身份驗証程式，利用帳號密碼的驗証動作，以決定這個使用者是否擁有會員的資格，若資格符合，則利用VideoLan對該使用者的IP位址傳遞視訊串流，而該使用者也可開啟VideoLan軟體來接收視訊串流。

六・結論

由於我們叢集系統所採用的是"分派器"架構，所以安裝於分派主機的核心模組是整個系統中最關鍵的部份。利用掛載核心模組的方法有下列優點：

- 可動態增加核心的功能，而不需要重新編譯核心。
- 由於掛載、卸載容易，故測試容易。

叢集系統中所採用的負載平衡演算法，能夠每隔一定周期依據節點負載資訊來進行流量分散，此方法有下列優點：

- 容易掌握實際負載狀況，進行適當的流量分散

使用備援主機監控著分派主機的狀態，若偵測到分派主機發生當機，則自動接替，完成備援的功能。如此一來叢集系統將具有容錯的功能。

利用軟體來完成負載平衡的方法，雖然在速度上是比不上其它硬體的解法，但是考量成本的問題，至少對於中小型網站而言，軟體解決方案仍是可以考慮採用的解決方案。以下為負載平衡演算法比較：

特性 / 演算法	複雜度	消耗資源	分配速度	負載分配公平性
系統所採用的動態自我適應法	高	中	普通	高
隨機法（Random）	低	低	快	低
輪流法（Round Robin）	低	低	快	低
客戶割分法（client partition）	低	低	快	低
LVS 的權重式輪流法	低	低	普通	低

負載平衡叢集系統與視訊隨選（VOD）的結合，是一種難度極高的網路技術應用，若採用封包改寫技術來轉送視訊串流的資料，將會造成分派器負擔過重，可能會使延遲時間過長。未來可以改進的部份，則是可以利用Linux Virtual Server Project中的Direct Routing技術來轉遞視訊串流資料，將可以降低分派器的負載，使視訊資料傳遞可以更有效率。[31]

七・參考文獻

[1] Valeria Cardellina, Michele Colajanni, Phil位址 S.Yu, "Dynamic Load Balancing On Web-server Systems", IEEE Internet Computing, 1999

[2] Linux Virtual Server Project http://www.linuxvirtualserver.org ,2005

[3] Linux High Availability Project http://www.linux-ha.org, 2005

[4] Packet Filtering HOWTO, Netfilter Hacking HOWTO, Netfilter Hacking HOWTO, NAT HOWTO, Netfilter Extensions HOWTO, http://www.netfilter.org/documentation/index.html, 2005

[5] David A. Ranch, Linux IP Masquerade

HOWTO,Auguest,2001
[6] Behrouz A Forouzan, TCP/IP Protocol Suite,2nd, DeAnza College,2003
[7] Daniel P. Bovet, Marco Cesati, Understanding the Linux Kernel,2nd, O'REILLY,2002
[8] Glenn Herrin, Linux IP Networking, 2000
[9] Eric Anderson, Dave Patterson, Eric Brewer, "The Magic Router, an application of fast packet interposing", 2nd,1996
[10] Cisco Inc, LocalDirector, http://www.cisco.com/warp/public/cc/pd/cxsr/400/index.shtml, 2005
[11] G.D.H Hunt et al, "Network Dispatcher: A Connection router for Scalable Internet Services", Journal of Computer Networks and ISDN System, 1998
[12] Resonate Co, Central Dispatch, http://www.resonate.com/solutions/literature/data_sheet_cd.php,2005
[13] TurboLinux, TurboCluster server, http://www.turbolinux.com/, 2005
[14] Cross-Reference Linux http://lxr.linux.no/source/, 2005
[15] The Linux Document Project, http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Linux-pdf-HOWTOs.tar.gz, 2005
[16] Beck, Kunitz, Magnus, etc. Linux Kernel Internals, Addison-Wesley, 2000
[17] Netfilter Project, http://www.netfilter.org/ , 2005
[18] VideoLan Client, http://www.videolan.org/ , 2005
[19] Using Linux and Iptables /ipchains to set up an internet gateway/firewall/router for home or office, http://www.yolinux.com, 2005
[20] Stevens, Advanced Programming in the UNIX Environment, Addison-Wesley,1992
[21] Richard Stevens, Unix Network Programming, Networking APIs:Sockets and XTI, Prentice Hall,,1998
[22] tcpdump, http://www.tcpdump.org/ , 2005
[23] David A. Patterson, John L. Hennessy, Computer Organization & Design The Hardware/ Software Interface, Morgan Kaufmann,1997
[24] CodeSourcery LLC, Advanced Linux Programming, New Riders Publishing, http://www.advancedlinuxprogramming.com/ , 2005
[25] Alessandro Rubini & Jonathan, Linux Device Drivers, 2nd Edition Corbet, 2nd Edition ,June 2001
[26] Mike Loukides, Andy Oram, Programming with GNU Software, O'Reilly, 1998
[27] Ellis Horowitz, Sartaj Sahni, Andserson-Freed ,Fundamentals of Data Structures in C, W. H. Freeman & Co, 1992
[28] Anthony Jones, Network Programming for Windows, Microsoft,2002
[29] Stephen Prata, C++ Primer Plus, 2nd , WAITE GROUP PRESS, 1995
[30] Gregory Satir and Doug Brown, C++ The Core Language, O'Reilly, 1999
[31] Jari Peltoniemi, Video-on-Demand Overview, Tampere University of Technology Telecommunications Laboratory, 1995