影音部落格平台之設計與實作

許琮琳 張堯棟 李佳瑋 李清俊 陳明惠 祝維勤 古勇勝
中華電信研究所 多媒體應用技術研究室
tsunglin@cht.com.tw dick@cht.com.tw jiawei@cht.com.tw lee228@cht.com.tw
tlcarol@cht.com.tw weichinc@cht.com.tw gary_ku@cht.com.tw

摘要

由於頻寬與數位技術的進步，影音內容逐漸成為部落格應用中資訊分享的重要元素；為了因應影音分享的需求日益浮現，一個專屬於影音分享的部落格平台有其存在的迫切性，本篇論文即在描述如何設計一個影音部落格平台。整個平台的實作涉及使用者介面開發、系統架構設計與影音格式轉換技術，其中尤以影音轉檔模組為整個系統的核心元件，用以將使用者上傳的分享影音轉換成單一格式的影音，讓影音能夠在相同的播放器被觀賞，本論文特別針對此模組的設計細節加以說明；最後，利用實驗的數據，驗證轉檔模組的成功率與擴充性。

關鍵詞：影音分享、影音轉檔、影音部落格、影音日誌

Abstract

Due to the improvement in network bandwidth and data digitizing technology, video has become a necessary element in most of the blog applications. To deal with the increasing need for video sharing of the users, a platform with the video sharing blog service has shown its importance. This thesis is aimed to describe how to design such a platform. The implementation of the platform involves the development of user interface, the design of system architecture and the conversion of video format. And the core of the system is the video transcoding module with the purpose of converting the video data of different formats into the same format so that they can be viewed with the same player. This thesis will elaborate on the design of the module and at the end, prove its success rate and expandability with the experimental statistics.

Keywords: video sharing, video transcode, video blog, vlog

1. 前言

部落格平台徹底地改變了現今網際網路資訊的產生方式。在過去，極少數人主宰了絕大多數人所能接觸的資訊，其內容往往都由入口網站或網站的擁有者所提供；但藉由部落格平台的開放內容的特性，讓資訊的產生由「一對多」的壟斷模式，逐漸演變為「多對多」的互動模式。網路上的每一個人，都可以自由地利用部落格分享自己生活或工作上的心得與想法[3]；同時，也能夠在閱讀其他人所分享資訊後，給予回應。

頻寬與數位技術的進展，使得部落格內容的呈現，從早期以文字與圖片為主的方式，演進到以影音串流為主的影音部落格。從字面上而言，影音部落格是部落格的一種，但著重於影音的分享。影音除了能更直覺地傳達資訊外，也較能夠滿足人性的表現慾望。因此，如何在現今網際網路的基礎下，設計與開發一個有效率的影音部落格平台，讓使用者能夠方便地分享影音，便是一項值得探討的議題。

本篇論文的目的，在於提出一個影音部落格平台的實作架構，而此系統目前為實際上線的系統，運作於 Xuite 影音日誌 (http://vlog.xuite.net)。整體系統的架構設計將於第二節中描述；第三節簡述系統的功能設計；而影音轉檔的必要性與轉檔模組的設計將於第四節中說明；第五節描述實驗進行的方式，並以數據資料驗證影音轉檔模組的成功率與擴充性；最後則為整篇論文的結論與參考文獻。

2. 系統架構設計

2.1 使用者介面

友善的使用介面是平台成敗的關鍵因素；當使用者利用 Internet 來存取影音部落格時，系統會根據裝置的不同，選擇下列兩種不同介面之一來提供服務；使用者的裝置資訊，包含作業系統與瀏覽器的種類，是藉由 HTTP 協定中的 User Agent 表頭[1]所提供的資訊加以判斷。

圖 1 正規瀏覽介面之首頁

- 正規瀏覽介面：當使用者是以個人電腦中的瀏

覽器來存取服務時，使用此介面來提供服務；這一個介面相容於目前大多數的瀏覽器，例如微軟的 Internet Explore 與 Mozilla 的 FireFox；圖一為此介面之首頁。

- 手持裝置瀏覽介面：當使用者使用 PDA 或手機等行動裝置時，受限於其運算資源與顯示螢幕的大小，必須提供專屬的介面[2]；圖二為此介面的影音播放頁面。

圖 2 手持裝置瀏覽介面之影音播放頁面

2.2 網路架構

網路架構的設計必須兼顧可靠性與擴充性；當使用人數與分享影音數量增多時，能夠在不影響實際運行環境的情況下增加硬體設備，以減輕平台的負擔，提供可靠的服務。系統的網路架構(圖三)包含數個元件，以下說明每一個元件的功用：

- NAS (Network-Attached Storage)：NAS 儲存技術[4]讓系統中的各個元件以網路的方式來存取資料；藉由 NAS，除了能穩定地將資料集中於同一處之外，也大大降低了元件之間溝通的複雜度。系統會將上傳影音的原始檔案、轉檔後的影音檔與影音的截圖儲存於此。
- File Upload Server：此元件負責處理使用者上傳影音的請求，除了將上傳的影音檔案放置於 NAS 中正確的位置外，也會於 Database 中新增一筆紀錄，讓 Transcode Server 能夠得知該檔案必須進行轉檔以及其位於 NAS 中的位置。
- Transcode Server：系統之核心元件，負責將各種格式的上傳影音，轉換成統一的影音格式；此元件上運行的影音轉檔模組，每隔固定的時間會詢問 Database Server，取得尚未轉檔的影音清單後，逐一進行轉檔，並將轉換後的影音檔放置於 NAS 中的正確位置，同時將此位置記錄於資料庫。當影音等待轉檔處理的時間過長時，可藉由增加此元件的數量來改善。
- Database Server：系統的資料庫；除了分享影音的資訊外，也記錄使用者的資訊與系統運作所必須參考的環境資訊。
- Web Server：標準的 HTTP 協定伺服器；提供使用者利用個人電腦上的一般瀏覽器來使用影音部落格平台的功能，此使用介面較不受存取裝置的影響，因而能提供較豐富的功能選單。當每單位時間所能處理的 HTTP 請求數目降低時，可藉由增加此元件的數量來改善。
- Web Server For Mobile：此元件功能等同於 Web Server 元件，但其服務的對象為 PDA 與手機上的瀏覽器，受限於運算資源與螢幕大小，此元件只提供最基本的功能介面。若手持裝置存取影音部落格的流量不大時，可將此元件的功能併入前述的 Web Server 元件。

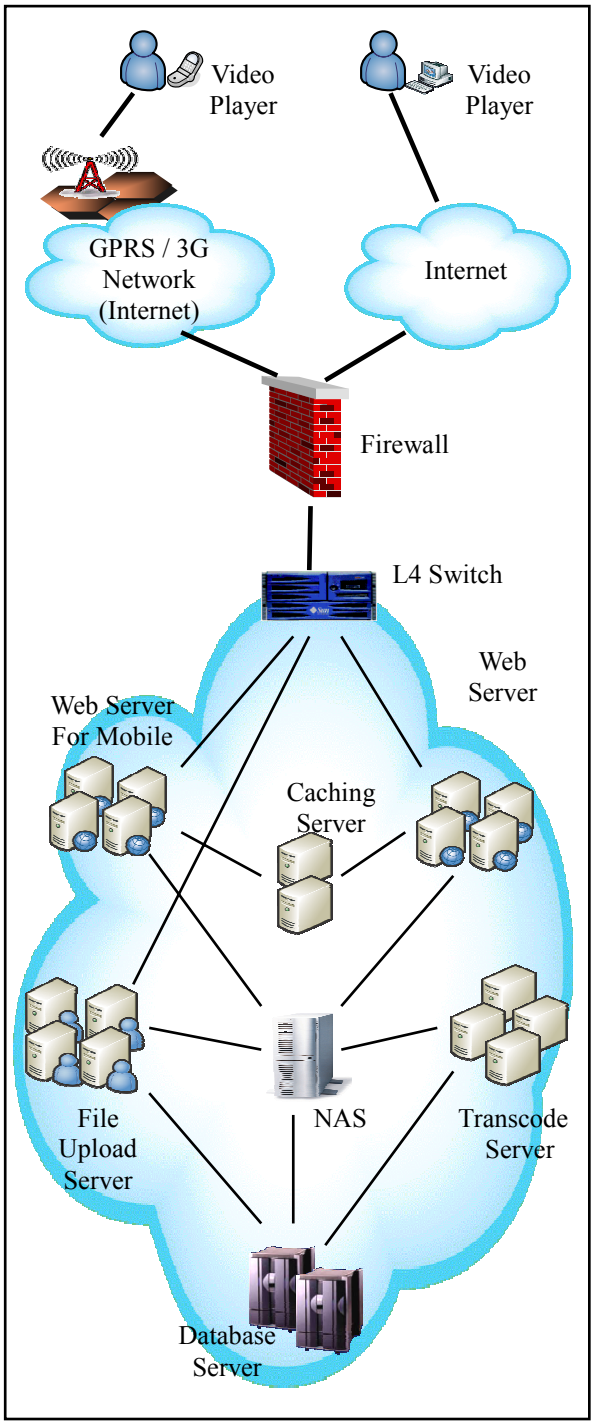

圖 3 網路架構示意圖

- L4 Switch：此元件為影音部落格平台對外的唯一入口，其作用在於平均地分派 HTTP 請求給 Web Server 處理[5]，希望使每一個 Web Sever 在任何的時間點，處理接近等量的請求，藉此減少使用者等待回應的時間。目前是以 HTTP 請求的 IP 位址作為分派的依據，相同 IP 的所有請求，由同一個 Web Server 處理。
- Caching Server：此元件完全使用記憶體來實作快取機制，藉以提升影音檔案的存取效率；同時，也用來記錄 Session 資料，排除磁碟存取的需求。
- Video Player：影音的播放器，為網頁內嵌的元件；瀏覽器必須下載影音播放器的執行檔。

3. 系統功能概述

本系統之功能主要區分為影音部落格服務模組與系統管理模組，前者的對象為一般使用者，後者則為系統管理員；藉由身份的認證，可限制所能存取的功能。圖四為系統基本功能的示意圖；以下扼要地描述各模組之功能：

- 影音上傳：經過身分驗證的使用者，能夠上傳分享影音；而在上傳過程中，除了影音檔案外，也必須提供該影音的某些資訊，包含影音名稱、影音描述、影音分類與自定的影音標籤等等；同時也能夠訂定影音的權限，決定其他使用者是否能夠觀賞此影音，也可要求系統在特定時間後才公開影音。
- 影音播放：任何使用者都可以觀看被公開的影音，也可以利用留言的方式，留下文字訊息給影音的擁有者；且在影音播放完畢後，系統會於播放器上，呈現系統管理人員所設定之推薦影音。而完成身分認證的使用者，也可以訂閱或收藏該影音。
- 影音查詢：使用者可輸入關鍵字，針對影音名稱與影音標籤等欄位進行查詢。
- 熱門影音列表：系統會記錄每一個影音被觀賞的次數，並以本週、本月、本年與全部等四種統計間隔，優先呈現次數高的影音。
- 最新影音列表：系統會以影音的上傳時間排序影音，優先呈現最近上傳的影音。
- 熱門標籤列表：列出系統中頻率較高的影音標籤；而每一個標籤的頻率為擁有該標籤的影音數目。
- 個人影音管理：使用者可管理已經上傳的影音，包含刪除影音、修改影音描述與公開權限等等功能。
- 影音檢舉：使用者完成身分認證後，可檢舉內容不當的影音；藉此讓平台上的使用者，共同維護平台影音內容的正當性。
- 影音轉檔：使用者上傳的影音必需成功地完成轉檔，才能夠被觀賞；此為系統的核心模組，將於第四節中說明。
- 用戶管理：系統管理人員必須經過身分驗證後，才能對使用者進行某些權限的設定，例如可使用的上傳空間量。
- 影音管理：系統管理人員必須經過身分驗證後，可執行推薦影音的設定、審核被檢舉的影音與刪除不適當的影音等等功能。

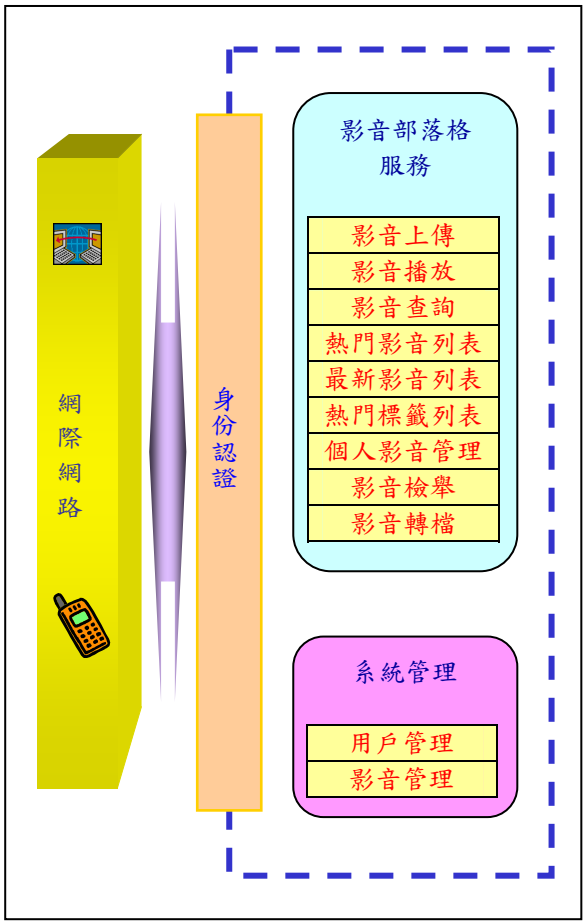

圖 4 系統基本功能示意圖

4. 影音轉檔模組

4.1 轉檔的必要性

將影音播放器嵌入於網頁中，能夠方便使用者觀賞影音；通常，要播放特定格式的影音，大都需要使用特定的編碼方式，一般的影音播放器只會具備必需的編碼方式，來播放其所支援格式的影音；因此，為了避免用戶須具備多個播放器，同時考量瀏覽器的支援度，平台選擇使用目前大多數的瀏覽器都支援的 Flash，來開發專屬的影音播放器，只要瀏覽器安裝了正確版本的 Flash Player，就能夠執行此影音播放器；而影音的格式也統一轉換成 Flash Player 所支援的 FLV 格式。

FLV 格式的影音能以串流的方式播放，也就是影音的播放與影音的下載能夠同時地進行，不需將

整個影音下載至用戶端，只要頻寬足夠，使用者不會感受到播放中斷或延遲。

當上傳影音的轉檔時間過長，或轉檔後的影像品質與轉檔前差異過大，都會造成使用者的不悅，降低系統的實用性；因此，令使用者肯定的影音部落格平台，必須具備卓越的轉檔能力。

影像品質的維持，並非能全由系統掌握，其與上傳影音的編碼方式與解析度，其判定方式也較為主觀；而轉檔所需的時間，除了受限於硬體速度與影音長度外，當上傳至平台的影音數量過多時，也會耗費時間在等待進行轉檔；因此，必須妥善地設計轉檔模組，搭配適當數量的 Transcode Server，降低等待轉檔的影音數量，盡可能地使影音完成轉檔的所需時間，趨近於真正執行轉檔所花費的時間。

4.2 轉檔模組的設計

轉檔模組的基本處理單位為一個 Process，其中包含四個 Thread，分別執行不同的工作，並使用二個 Queue 作為 Thread 之間的溝通機制。平台的轉檔模組必須執行一個以上的轉檔 Process，且所有的 Process 可以分散於不同的 Transcode Server。

原則上，轉檔 Process 數量的增加，能夠提昇單位時間內，影音轉檔的總產出，但轉檔過程的計算量不小，屬於 CPU-bound 類型的程式，若過多的 Process 反而會得到反效果；效率最佳化的 Process 數量，應取決於 Transcode Server 的 CPU 數量。論文中的第五節會藉由實驗，計算在某一個硬體設備下，效率最佳的 Process 數目。圖五為轉檔 Process 的示意圖 ，以下扼要說明其中每一個元素的作用：

- Wait Queue：記錄目前等待被 Transcode Thread 進行轉檔處理的影音資訊。
- Poll DB Thread：定期詢問資料庫，取出最多 N 筆待轉的影音，置入 Wait Queue；其 N 值會視轉檔情況自動調整；且在任何的時間點，只會有一個轉檔 Process，能對資料庫執行待轉影音的查詢。
- Output Queue：紀錄影音執行轉檔後的結果。
- Transcode Thread：負責將影音轉換為 FLV 格式的檔案，並將轉檔的結果寫入 Output Queue；而目前所能處理的影音格式包含 MP3、WMA、FLV、WMV、AVI、ASF、MPG、MPEG、MOV、MP4、M4V、3GP、3GP2 與 RM。
- Output Thread：將 Output Queue 裡的影音資料更新回資料庫，同時也把必要的統計或錯誤訊息寫入 Log 日誌檔。
- Command Thread：負責接收與處理由 Socket 介面所傳送進來的三種命令：第一、要求 Process 處理完目前正在轉檔與等待轉檔的所有影音後，結束 Process；第二、輸出 Process 的統計資訊與正在轉檔及等待轉檔的影音資訊；第三、優先執行某一個影音的轉檔，而該影音的資訊會包含在傳送進來的資料中。

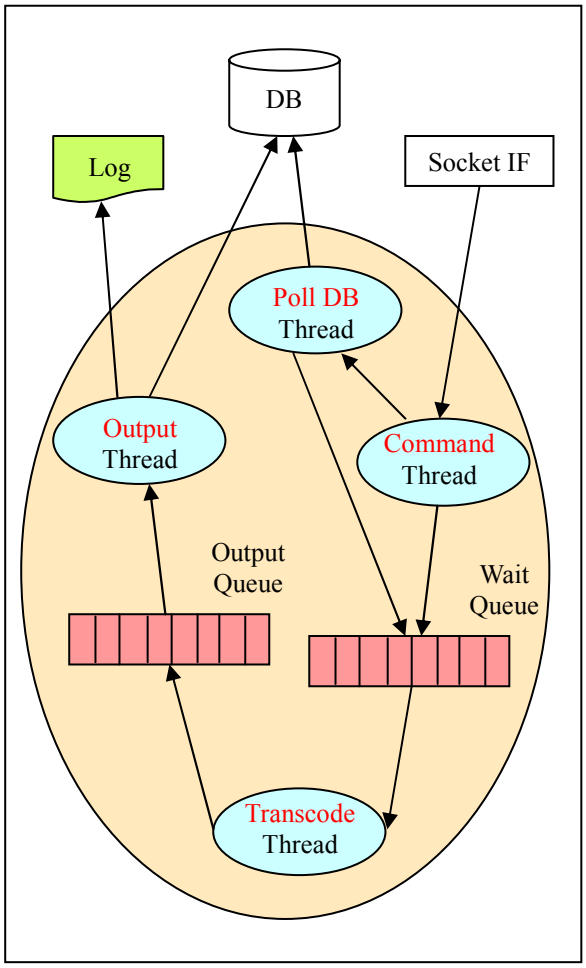

圖 5 影音轉檔 Process 的示意圖

5. 實驗數據

5.1 轉檔成功率與效率最佳的轉檔 Process 數

本實驗從使用者所上傳的影音中，隨機挑選 514 部影音，計算轉檔的成功率；同時使用相同的 Transcode Server，在不同轉檔 Process 數目的情況下，計算完成此 514 部影音轉檔所需的時間。表 1 為 Transcode Server 的資訊(此為實驗用之設備，非實際上線之設備)，表 2 為 514 部實驗影音的統計資訊。

表 1 Transcode Server 的資訊

項目	數值
CPU 規格	Intel Xeon 1.5G MP
CPU 數量	2
Memory 數量	2048 MB
OS 版本	Red Hat Enterprise Linux
Disk 空間	72 GB

表 2　514 部影音的統計資訊

項目	數值
MPEG 格式數目	2
MP3 格式數目	156
AVI 格式數目	48
FLV 格式數目	11
MPG 格式數目	55
WMA 格式數目	24
RM 格式數目	14
ASF 格式數目	8
WMV 格式數目	129
3GP 格式數目	30
MOV 格式數目	15
MP4 格式數目	22
轉檔前影音的平均檔案大小	4745 MB
轉檔後影音的平均檔案大小	2866 MB

根據表 3 的實驗結果，在 4 個轉檔 Process 同時運作的情況下，所需的整體轉檔時間較少；超過 4 個 Process 時，由於 Transcode Server 的運算能力已達飽和，無法大幅降低轉檔時間。由表 4 得知轉檔的成功率為 97%，有 17 部影音轉檔失敗，進一步分析後，得知其原因在於檔案內容的格式有錯誤，導致轉檔過程被中斷。

表 3 不同轉檔 Process 數目所需的轉檔時間

Process 數	轉檔時間 (秒)	平均轉檔時間 (秒 / MB)
2	81070	17.09
3	72371	15.25
4	66135	13.94
5	66235	13.96
6	66264	13.97

表 4 影音轉檔成功率

項目	數值
成功影音數目	497
失敗影音數目	17
成功率	97%

5.2 影音轉檔模組的可擴充性

從實驗 5.1 的結果可得之，在軟硬體規格如表 1 的 Transcode Server 下，4 個轉檔 Process 能夠擁有最佳的產出。本實驗將驗證轉檔模組的可擴充性：當系統上線運作時，若影音等待轉檔的時間過長時，只需增加 Transcode Server，就能夠有效地改善；實驗使用表 2 所描述的 514 部影音，分別以 2 部與 3 部相同的 Transcode Server(表 1)進行影音轉檔，計算所需的時間，而使用的 Transcode Sever 均執行 4 個轉檔 Process。表 5 的實驗結果驗證了影音轉檔模組的可擴充性，每增加一部 Transcode Server，都能線性地減少所需的轉檔時間。

表 5 不同 Transcode Server 數所需的影音轉檔時間

Transcode Server 數	轉檔時間 (秒)
1	66135
2	33178
3	22122

6. 結論

本篇論文提出了一個影音部落格平台的架構，並且實作整個系統。為了因應一般電腦與手持裝置的處理能力不同，需設計不同的使用者介面，方便使用者存取平台的功能；而平台的基本功能可從影音部落格服務與系統管理等兩個面向來探討。文中也說明了影音轉檔的必要性，並且描述轉檔模組的設計方式，透過實驗驗證模組的擴充性。而未來的發展工作包含了影音版權的驗證與保護。

參考文獻

[1] http://www.ietf.org/rfc/rfc2616.txt.

[2] Anne Kaikkonen, Virpi Roto, “Navigating in a Mobile XHTML Application,” *Proceedings of the SIGCHI conference on Human factors in computing systems CHI '03* , Volume No. 5, Issue No. 1, pp. 329-336.

[3] Bonnie Nardi, Diane J. Schiano, Michelle Gumbrecht, Luke Swartz, “Why we blog,” *Communications of the ACM* , Vol. 47, No. 12, pp. 41-46, December 2004.

[4] Garth A. Gibson, Rodney Van Meter, “NETWORK ATTACHED STORAGE ARCHITECTURE,” *Communications of the ACM*, Volume 43, No. 11, November 2000.

[5] MICHELE COLAJANNI, PHILIP S. YU, “The state of the art in locally distributed Web-server systems,” *ACM Computing Surveys*, Vol. 34, No. 2, pp. 263–311, June 2002.