結合 MPEG-DASH 雲端轉碼器之動態資源配置方法設計與效能分析

郭中允[a]、賴明甫[a]、蘇暉凱[b]、吳承崧[a]
國立中正大學通訊工程研究所[a]
國立虎尾科技大學電機工程系[b]

摘要 —為了提供異質網路環境下不同終端設備合適的影音串流服務，必須透過即時的影音轉碼。但是，即時的影音轉碼需要大量的運算資源很難在一般終端設備上實現，透過雲端運算技術可以將轉碼的運算工作交給雲端伺服器以提供不同終端設備的需求。為了有效的運用雲端資源，本論文提出了 Earliest Estimated Finishing Time First 此轉碼器指派演算法，此演算法可以根據 fragment 的特性及每個轉碼器的處理能力來估測出最快處理完當前工作的轉碼器並動態的將 chunk 分送到此轉碼器。未來，本研究將結合 MPEG-DASH 並設計一套可適性演算法來達成可適性影像串流傳輸。

一、 簡介

隨著寬頻網路技術的進步，加上平板、智慧型手機等行動裝置的迅速普及，多媒體串流服務成為人們日常生活中不可或缺的一部份。為了提供異質網路環境下不同終端設備合適的解析度、影像品質，必須透過即時的影音轉碼，並根據當前的網路狀況調整傳送的影像品質以符合網路連線頻寬的不同速率以達到可適性[1]。

傳統的影音轉碼方式是將儲存於終端設備中的影音檔利用現有的轉碼軟體進行轉碼，但這將導致終端設備儲存空間的浪費而且複雜的運算可能耗盡終端設備的 CPU、記憶體等資源。隨著雲端技術不斷的發展與廣泛的應用，可以利用雲端技術來進行複雜的運算處理工作同時解決使用者終端設備空間不足的問題[2]。

在傳輸串流機制方面目前可以分為兩類，即時串流傳輸以及近即時串流傳輸，即時串流的傳輸協定大都採用的是 UDP 以及 RTP，UDP 本身並無速率控制機制，且傳輸特性為盡全力傳輸，容易盲目的傳送造成網路壅塞的發生。再加上 UDP 沒有確認及重傳的機制來確保影像資料的完整性，因此，當傳輸的過程中若遺失不同重要性的資料，便會造成視訊品質不同損害程度的影響。近即時串流傳輸會將要播放的影像資料先接收暫存在緩衝區內，所以可以容忍的延遲相較於即時串流大的多，約為數秒，近年來，陸續有組織發展了使用 TCP 協定傳送近即時串流的機制，TCP 協定本身具有壅塞以及流量控制機制，會根據網路壅塞情況，改變傳輸速率以適應目前的網路狀況，而且 TCP 協定的重傳機制，可以確保串流影像資料的完整性。

近年來，MPEG 組織提出了 HTTP 可適性串流的共通標準 MPEG-DASH[3]，MPEG-DASH 定義了串流封裝架構、片段的分割，以及取得片段的方法，對於可適性的演算法，串流機制的供應商可以自行設計，對於可適性串流的開發，MPEG-DASH 擁有了較高的設計彈性。

本研究提出以 DVB-T 作為即時的影音串流來源，透過 OpenNebula 建構的 IaaS 轉碼雲來提供即時的影音轉碼服務，可以將所接收下來的影音串流即時轉成影音片段後封裝進 MPEG-DASH 的結構中並產生 MPEG-DASH 定義的清單 MPD，透過 MPEG-DASH 的請求機制用戶端利用 HTTP over TCP 的方式先向伺服端請求 MPD 清單後根據清單中的 URL 資訊來請求影音片段，在傳輸的過程中結合可適性演算法由接收端決策所要接收的影像品質，並將此決策回饋給雲端來改變雲端中 transcoder 的 codec 以轉碼出符合接收端頻寬大小的影音。

二、 相關技術

2.1 雲端運算

雲端運算(Cloud Computing) 是目前資訊技術最熱門的話題之一，雲端的基本概念，是透過網路將龐大的運算處理程式自動分拆成無數個較小的子程式，再由多部伺服器所組成的龐大系統搜尋、運算分析之後將處理結果回傳給使用者。根據美國國家標準局(NIST)於The NIST Definition of Cloud Computing[4]對於雲端所下的定義，雲端運算擁有五大特徵、四個佈署模型且依據其提供服務的方式不同可分為三種模式分別是基礎架構即服務 (Infrastructure as a Service，IaaS)、平台即服務 (Platform as a Service，PaaS)、軟體即服務 (Software as a Service，SaaS)。

OpenNebula(ONE)是歐洲研究學會發起的虛擬基礎設備和雲端運算的計畫，屬於開放原始碼的虛擬基礎設備引擎讓用戶可在一群實體資源上動態的佈署虛擬機器。目前最新版本已更新至 4.2，可支援 Xen、KVM 等虛擬化技術亦可透過 ONE 來控制 Amazon EC2 上的虛擬機。OpenNebula 特色在於將虛擬平台從單一實體機擴充到一群實體資源。OpenNebula 架構的概念為在 Virtualizer(Xen、KVM)和實體機處設備間實現一個虛擬層，在此層實作管理的機制，包含管理每台實體機器上的 Virtualizer 和實體機器上的 VM(Virtual Machine)，因此透過 OpenNebula 可以建構出屬於私人的 IaaS 私有雲。

2.2 MPEG-DASH

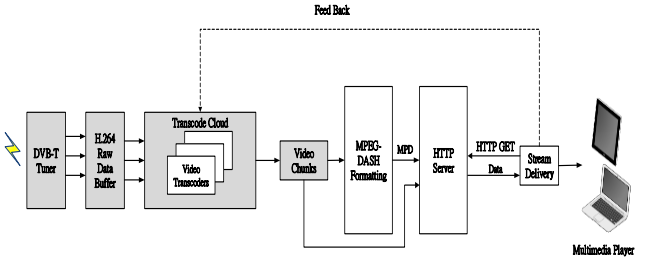
MPEG-DASH 定義了影像資料的封裝格式、片段分割，以及片段取得的方式，也定義了一種為 XML 格式的清單，稱為 Media Presentation Description (MPD)，用來描述各種不同頻寬需求的影音資料的不同片段[3]，透過 URL 來表示出每個影音片段的位址，接收端透過 URL 請求不同時間的影音片段資料，透過這種請求的方

式，使接收端可以根據自身的網路環境，無縫切換不同的影音品質，MPD 清單的架構，類似於階層式的架構，其內會包含單一或多個 Representation 資訊，一個 Representation 代表同一個媒體元件來源，具有不同的媒體資訊，每一個 Representation 將由一個或多個 Segment 所組成，每個 Segment 則是一段時間長度的媒體串流。Segment Info 方塊包括一個 Initialization Segment，以及數個 Media Segment。Initialization Segment 內含要初始化 DASH 終端裝置的媒體解碼器所需資訊，但並沒有包含實際的媒體資料。實際資料存放在 Media Segment 中，每一個 Media Segment 都具有唯一的 URL、索引 Index，以及節目起始時間。

三、 系統架構與研究方法

3.1 系統架構

本論文之系統架構如圖一所示，左端為多媒體串流資料來源，經 DVB-T Tuner 接收節目之串流資料，接收下來的 H.264 高品質 Raw Data 先儲存於 H.264 Raw Data Buffer，接著透過 轉碼雲中的 Transcoders 進行多媒體即時轉碼作業以提供即時的串流播送，將轉碼完的影音片段(chunks)儲存至 HTTP Server 中並透過 MPEG-DASH 的格式化產生 MPD 清單。在影音串流傳輸方面，本論文提出了 Pull-Based 的可適性串流機制，接收端一開始向 HTTP server 請求 MPD 清單檔案，接收端分析完 MPD 清單後再透過清單內的 URL 資訊向 HTTP server 請求影音片段，在請求傳送的同時，Client 端會不斷的監控接收緩衝區，根據緩衝區消長情況以及緩衝區暫存尚未被播放掉的影像資料(Residual Playout time)估算出目前所播放的影像品質是否合適，若發現頻寬不足需降低影像品質或頻寬充足需提升影像品質，Client 端會回饋一個訊息給轉碼雲以進行 Transcoder 之 Frame Rate 等參數動態調整以轉碼出適合 Client 端播放品質的影像。



圖一：系統架構圖

3.2 雲端多媒體串流轉碼系統

圖二為本論文之雲端多媒體串流轉碼系統架構圖，利用 OpenNebula 建構出 IaaS 的轉碼雲，其中 Front-End 提供 Transcoding Cloud 管理功能，包含系統管理 ONED，Drivers 管理、Guest OS Images 管理、實體與虛擬系統資源 Monitoring…等。Cluster Node 提供實體運算資源，並且可以動態建立 (執行) 與移除(關閉) VM Instance，每一個 VM Instance 運作獨立之 Guest OS，並且執行相關軟體提供轉碼服務。Transcoding Proxy(之後統稱 TP)則負責提供 Storage 以 NFS 服務提供資料儲存資源，並且執行 Transcoder 指派演算法即本論文提出之最早預期完成時間優先演算法，將儲存好的 H.264 Chunks 指派至最適當的 Transcoder 進行轉碼。MPEG-DASH Formatting 提供將轉碼後的 H.264 Chunks 封裝進 MPEG-DASH 結構中的服務，並產生 MPD 清單供用戶端取用。

3.3 Earliest Finishing Time First VM 指定演算法

本演算法在每個 chunk 進入 TP 時，便會利用此 chunk 的影片種類、各個 transcoder 的 Transcoding Rate 計算出每個 transcoder 將此 chunk 轉碼完成的時間，再從所有 transcoder 中挑選完成時間最快的，將此 chunk 交由完成時間最快之 transcoder 處理。

每個 transcoder 的 transcoding rate 會在完成一個 chunk 轉碼之後即時動態地更新，而如果上述之尋找最快完成轉碼之 transcoder 沒有找到符合的，也就是沒有 transcoder 能夠符合 required transcoder delay，此時代表 transcoder 的數量已不足，則會動態地增加 transcoder 以符合所需之 transcoder delay。

以下為演算法之詳細流程，首先定義會用到的參數

$$\text{Normalized Transcoder Rate:}\frac{\text{fragment playout time}}{\text{transcoding time}}$$

τ_j^i : processing time of i_{th} chunk of j_{th} transcoder

t^i : arrival time of i_{th} chunk

f_j^i : finish time of i_{th} fragment on j_{th} transcoder

D_{max} : required transcoder delay

R_j : transcoder rate of j_{th} VM

每個 Transcoder 的 Transcoder Rate 並不固定會因

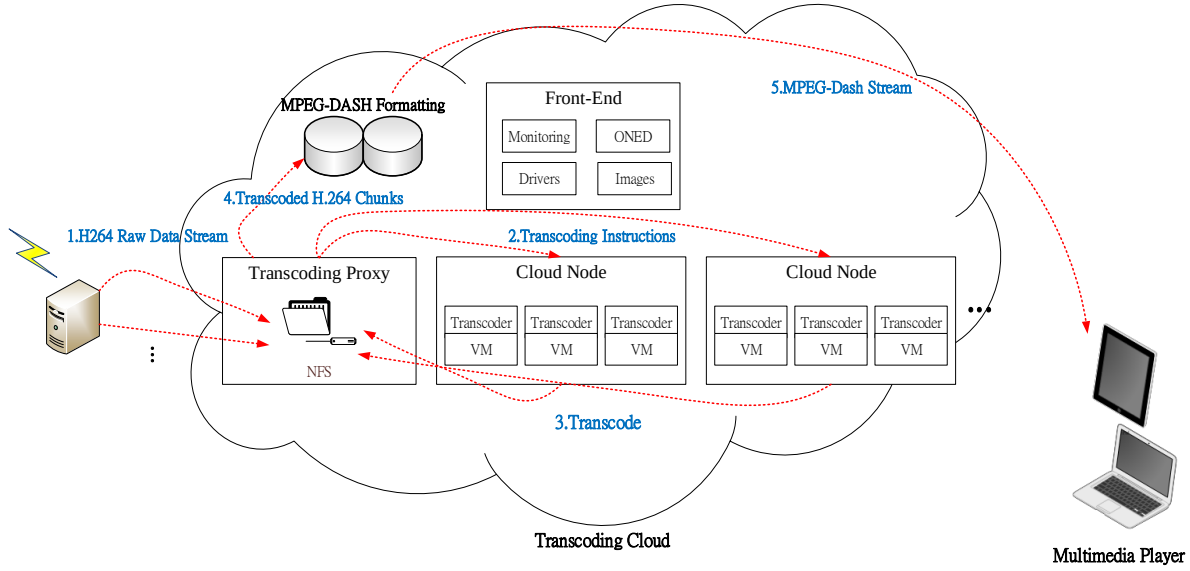
A.轉碼品質

B.影片類別

C.VM 的結構及 PM 的負載的不同而不同

Step1：首先依據 A、B、C 三項因素預估所需的 VM 數

$$K = Arg(\min_{n}(\sum_{j=1}^{n} R_j > 1)) \qquad (1)$$



圖二：雲端多媒體串流轉碼系統架構圖

(1)式所求出之 K 即為起始時 VM 使用個數

Strp2：當第 i 個 chunk 進入 TP 時預估第 i 個 chunk 在第 j transcoder 所需轉碼時間為(2)式

$$\tau_j^i = \frac{T}{R_j} \qquad (2)$$

Step3：此 chunk 在第 j 個 transcoder 經 τ 秒後完成轉碼，則以移動平均更新 第 j 個 transcoder 轉碼率為式(3)

$$R_j = (1-\alpha)R_j + \alpha\frac{T}{\tau} \qquad (3)$$

Step4：預估第 i 個 fragment 在用第 j 個 transcoder 完成的時間為假如第 j 個 transcoder 在 t^i 時沒有 workload，則為式(4)所示。假如第 j 個 transcoder 在 t^i 時尚有工作正在執行，預估在 f_j 時能完成目前工作則為式(5)所示。

$$f_j^i = t^i + \tau_j^i \qquad (4)$$

$$f_j^i = f_j + \tau_j^i \qquad (5)$$

綜合上述兩式預估完成時間為(6)式所示

$$f_j^i = \max(t^i, f_j) + \tau_j^i \qquad (6)$$

Step5：利用 Step4 的方式在所有 transcoder 中選擇最早預估完成的 transcoder 如(7)式所示

$$j = Avg(\min_j f_j^i) \qquad (7)$$

Step6：此 chunk 預估完成時間為(8)式並更 transcoder j 的完成時間為(9)式

$$f_*^i = \min_j f_j^i \qquad (8)$$

$$f_j = f_*^i \qquad (9)$$

最後增加 transcoder 的條件為所有的 transcoder 都無法符合所需的最大延遲，也就是對所有的 transcoder 都符合(10)式就必須增加 transcoder

$$f_j^i - t^i > D_{\max} \qquad (10)$$

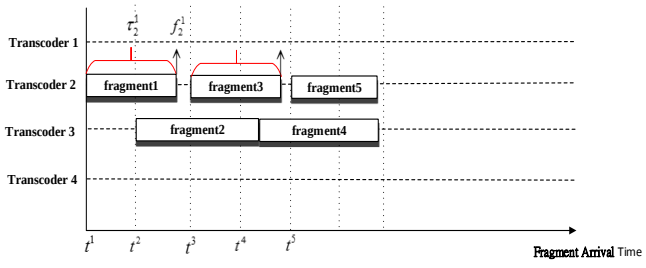
四、 效能分析

本研究在 Transcoder 指派演算法提出了 Earliest Estimated Finishing Time First 此最佳的 fragment 分送方式，Earliest Estimated Finishing Time First 可以根據 fragment 的特性及每個 transcoder 的處理能力來估測出最快處理完當前工作的 transcoder 並動態的將 fragment 分送到此 transcoder。本研究亦對此演算法與另外兩種較為直觀的演算法 Round Robin 與 Most Powerful Available Transcoder First。Round Robin 演算法為在每個 fragment 進入 Transcoding Proxy 時依照 transcoder 的順序逐一分配 fragment，而當有 fragment 進入 Transcoding Proxy，此時被輪到的 transcoder 正在工作時，代表 transcoder 數量已不足，便會動態地增加 transcoder。Most Powerful Available Transcoder First 演算法在每個 fragment 進入 Transcoding Proxy 時會從目前閒置之 transcoder 中尋找能將此 fragment 最快轉碼完成之 transcoder，將此 fragment 交由最快轉碼完成之 transcoder 進行轉碼。本研究並以此三種演算法的 fragment 指派示意圖與模擬來進行比較與分析。

4.1 三種演算法 fragment 指派示意圖

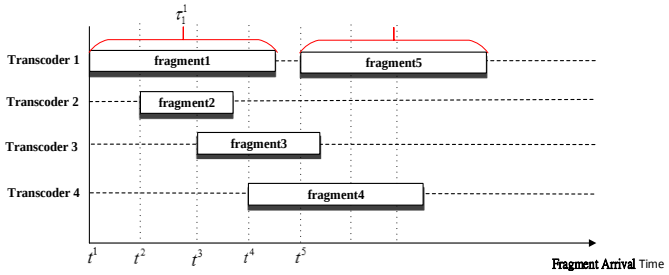
在這三個示意圖中 Transcoder1、Transcoder2、Transcoder3、Transcoder4 的 transcoder rate 分別為 1/3.2、1/1.7、1/2.4、1/3.5 且 D_{max} 為 4 個單位時間。

● Earliest Finishing Time First



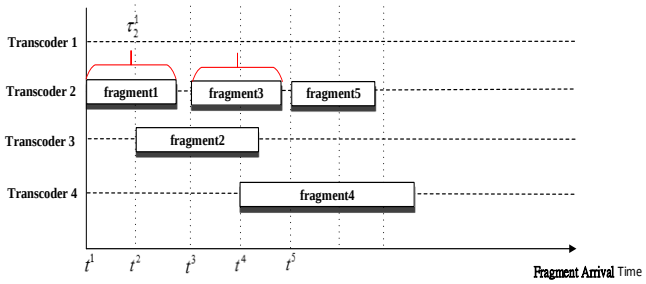
圖三：EFTF fragment 指派示意圖

● Round Robin



圖四：RR fragment 指派示意圖

● Most Powerful Available Transcoder First



圖五：MPATF fragment 指派示意圖

由這三個示意圖可以看出本研究所提出的演算法所使用到的平均 transcoder 數是最少的而且每個 transcoder 的使用率也是最高的。

4.2 模擬方式

在本研究的模擬實驗中先配置好五個不同 transcoder rate 的 transcoder 後，每單位時間會有一個播放時間為一個單位時間的 fragment 須進行指派，每個 fragment 依據其特性如動態、一般、靜態等會有其相對應的係數分別為 0.8、1.0、1.2，依據演算法來分配每隔 fragment 給合適的 transcoder 來進行轉碼。在整個模擬結束後會計算出演算法所使用到的平均 transcoder 數和每個 transcodre 的平均使用率。

4.3 模擬數據

由表一可以看出我們所提出的演算法 Earliest Estimated Finishing Time First 所使用的平均 transcoder 數是最少的而且每台 transcoder 的使用率也是最高的，所以驗證了我們所提出的方法可以使用最少的資源且資源使用率也是最多的。

表一：三種演算法模擬結果比較

	TC1 平均使用率	TC2 平均使用率	TC3 平均使用率	TC4 平均使用率	TC5 平均使用率	平均使用的 TC 數
EFTM	81.91	81.04	45.64			2.1
MPATF	63.08	64.97	18.96	2.96		2.96
RR	52.12	51.68	52.31	50.36	44.77	4.46

結論

本論文提出多媒體串流轉碼雲端系統架構，設計出一套雲端轉碼系統讓我們有效地分配雲端的運算資源來進行轉碼，並且針對雲端即時轉碼架構提出 Earliest Estimated Finishing Time First 即時轉碼工作排程演算法，此方法為最佳的即時轉碼工作排程方式，比較起傳統之 Round Robin 和 Most Powerful Available Transcoder First 演算法能夠更能有效地利用雲端資源，動態的更新轉碼率和及時增加 transcoder 使得轉碼效率達到最佳。

未來，本研究將延伸至可適性影像串流傳輸，將結合 MPEG-DASH 來設計出一套可適性串流演算法，在影音傳輸的過程中，藉由估算網路頻寬與接收端 Buffer 消長情形來決策出符合使用者網路狀況的影音品質，並動態地調整以達可適性之目標。相信本研究成果，對於未來多媒體雲端轉碼與串流傳輸領域發展具有貢獻。

參考文獻

[1] S. Lederer, C. M¨uller, and C. Timmerer, “Dynamic adaptive streaming over HTTP dataset,” in Proceedings of the 3rd Multimedia Systems Conference. ACM, 2012, pp. 89–94

[2] Yu Wu, Zhizhong Zhang, Chuan Wu, Zongpeng Li, Francis C. M. Lau, “CloudMoV: Cloud-Based Mobile Social TV”, IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 15, NO. 4, JUNE 2013

[3] Sodagar, I., “The MPEG-DASH Standard for Multimedia Streaming Over the Internet,” MultiMedia, IEEE, 62-67 April. 2011

[4] Lee Badger, Tim Grance, Robert Patt-Corner, Jeff Voas, ”Cloud Computing Synopsis and Recommendations” in NIST Special Publication 800-146, May 2012