基於雲端運算之 DVC 到 SVC 快速轉碼系統

張剛止 [a]、陳柏翰 [a]、李昌明 [a]、蘇暉凱 [b]、吳承崧 [a]
國立中正大學通訊工程學系 [a]
國立虎尾科技大學電機工程學系 [b]

摘要 —網際網路與各種個人行動設備的發展日益蓬勃，使用者多元的多媒體需求讓傳統單機編碼逐漸無法負荷，因此透過雲端運算技術，將需要高複雜度運算 DVC 的解碼端與 SVC 的編碼端集中於雲端伺服器處理，同時透過所提出的轉碼影像檔分配策略，可以進一步的提高雲端編碼效率，本論文針對 SVC 的時間與空間可調性，設計重複利用分散式視訊編碼解碼時的移動向量資訊來幫助可調性視訊編碼以加速編碼速率。所節省的時間與資源不但能節省雲端資源的使用，也能處理大量數據與更佳品質的串流服務。本論文所架設的 DVC-SVC 雲端快速轉碼系統和單機編碼相比，最多平均可加速 396.03%的編碼時間，每秒可完成 4.08 張的轉碼，且可以依據使用者需求再擴增系統以增加轉碼速率。

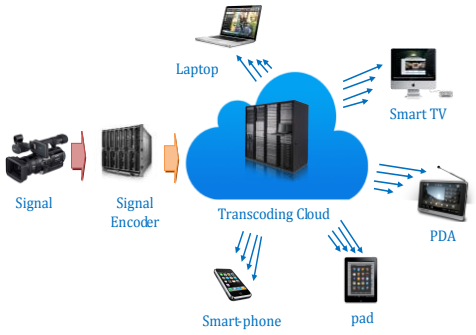
關鍵詞:雲端運算、分散式視訊編碼、可調性視訊編碼、轉碼器、空間與時間可調性

一、 簡介

在過去數年中，由於網際網路的蓬勃發展，誕生了雲端運算的新名詞，雲端運算是一種新興且旨在提供網路上的各種計算和儲存的服務 [1]。使用者不需要了解雲端伺服器的內部架構或擁有相關專業知識，而是透過網路就可以取得到相應的服務或存取資料。而應用於雲端運算上的各種服務，如搜尋引擎、輔助科學研究、多媒體運算、存儲空間和虛擬遠端桌面…等，更是充斥在我們現今的生活中，連上使用者所需要的服務雲以獲得所需的服務，成為未來使用網路的新方向。

而頻繁使用各種連網方式如 3G 與 Wi-Fi 等來獲取資訊，利用個人行動裝置觀看影片或是上傳生活影像或影片，已經成為生活中的日常習慣。但各種行動裝置擁有相異的處理能力，使用者對於多媒體影像也有著不同品質與規格的要求，因此各種影像編碼方式相應而生，可是在同一裝置上放置各種規格的編解碼器相當不符合成本，所以需要透過轉碼器將各種多媒體影像規格進行轉碼。但考慮到各種行動裝置，如智慧型手機，平板電腦等的處理能力，傳統編碼器 H.264 [2]雖然具有良好的壓縮率，但是編碼複雜度高，較不適合用於個人行動裝置上，因此需要低複雜度，低成本，低耗電且具有良好的壓縮率編碼方式。為了滿足此需求，由消息理論中的 Slepian-Wolf 理論[3]和 Wyner-Ziv 理論[4]延續發展的分散式視訊編碼 (Distributed Video Coding，DVC) [5]可供行動裝置來使用。DVC 將編碼器的複雜度轉移到解碼器上，達到使用者端低複雜度的需求，減低使用者端的負荷量。同時考慮到需要提供影像給不同的使用者使用，因此設計轉碼器將影像轉碼為編碼較為彈性的可調性視訊編碼 (Scalable Video Coding，SVC) [6]提供給各種不同的使用者。SVC 可提供多種品質的位元串流，以滿足不同行動裝置的需求。

本論文利用如圖一轉碼雲的概念，將 DVC 複雜的解碼端與 SVC 編碼端集中於雲端伺服器，並為了減低轉碼時的複雜度以及有效運用雲端資源，利用 DVC 解碼時所產生的移動向量，作為 SVC 編碼時的移動估測，以減少轉碼器的複雜度。使用者只需要上傳 DVC 格式的壓縮檔，並透過網路與轉碼雲連結，得到 SVC 格式的壓縮檔，避免了使用者端的高複雜度運算。並探討雲端轉碼器內部進行轉碼時的分配策略，並將待轉碼的壓縮檔不斷地分配給轉碼雲中的運算節點以達到最快速的處理效率，並希望提供有效的轉碼雲使用參考模型給與使用者提供服務上的參考依據。



圖一： 轉碼雲概念圖

二、 相關背景

2.1 DVC

傳統的編碼器如 H.264 雖然具有良好的壓縮率，但高複雜度的編碼端以及簡單的解碼端架構不適用於所有系統。為了滿足編碼端低複雜度、低成本、低耗電且有良好壓縮率的需求，因此分散式訊源編碼 (Distributed Source Coding, DSC) 是最佳的解決方案之一。如果將此應用於視訊處理，即為分散式視訊編碼(Distributed Video Coding, DVC) [5]，其基礎理論是由消息理論 (Information Theory) 中 Slepian-Wolf 理論以及 Wyner-Ziv 理論延續發展，將編碼器的複雜度轉移到解碼器上，達到使用者編碼端低複雜度的需求。由於某些編碼裝置放置於不同的環境，配合需求有時不只放置一個編碼裝置，甚至在各處放置編碼裝置，再將各處收集到資訊整

Corresponding author:cmlee@ccu.edu.tw
DOI:10.6159/IJSE.2014.(4-2).16

合及分析處理，最後得到使用者所想要的資訊。這類型的應用，會需要大量的編碼裝置，因此編碼端裝置成本要低，構造簡單，並且盡可能的及時傳輸資料，因此解碼端承受了整體的系統大部分的運算量，目前常見的應用例如無線感測網路 (WSN)、雲端架構、視訊監控系統等。

2.2 SVC

SVC 這個標準是由 MPEG (Motion Picture Expert Group) 委員會和國際電信聯盟 (ITU-T) 共同合作的聯合視訊團隊 (Joint Video Team，JVT) 所制定的。在 SVC 標準中，除了強調其壓縮比之外，更重要的就是它具有時間可調性 (Temporal Scalability)、空間可調性 (Spatial Scalability)、品質可調性 (SNR Scalability) 等三個維度 (Dimension) 的可調性，能夠隨著可用頻寬的增減，相當有彈性地動態調整其內容。

2.3 雲端轉碼器

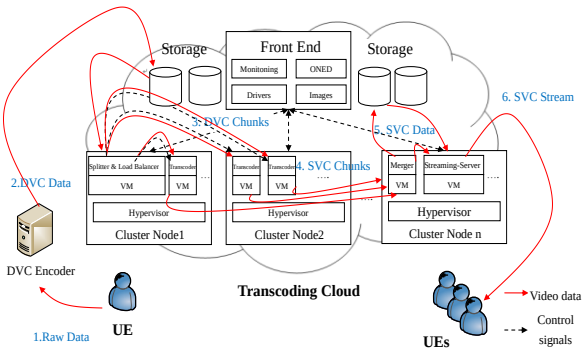
視訊轉碼器的主要任務是將訊號源的格式轉換到另一個格式上，而這個任務必須盡可能的讓視訊轉碼器的效率提高。因此，視訊轉碼器的技術上，主要專注於如何利用不同視訊壓縮標準間的資訊去改善視訊轉碼器，以得到較好的編碼效率。主要的核心問題在於如何讓視訊轉碼器的轉碼速度加快。

在目前的雲端轉碼器中，利用了分散式處理的概念來進行轉碼，與傳統叢集式運算 [7]不同之處為將運算實體機換成虛擬機器 VM，而能夠更動態且便利的雲端系統，在系統中具有三種角色的節點，分別為管理者 (Manager)，運算節點虛擬機 (VM)，以及串流伺服器 (Streaming Server)。管理者負責統籌資源的分配與負載平衡運算策略 [8]，將 Source Video 的影像壓縮檔以 GOP 為單位切割，並將各個 Part of Source Video 分配至各運算節點虛擬機中進行轉碼。每台 VM 中具有相同的轉碼器，當接受到欲轉碼的影像壓縮檔後會進行轉碼為目標格式的壓縮檔，將轉碼後的 Part of Transcoded Video 彙整後將完整的 Transcoded Video 經由串流伺服器傳遞給使用者。

三、 系統架構

3.1 雲端轉碼器整體系統架構

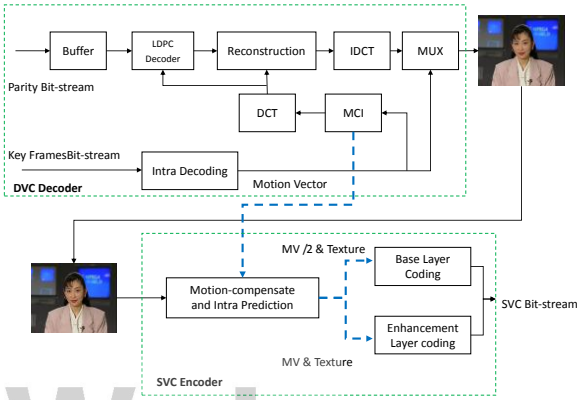
本論文所設計的雲端 DVC-SVC 轉碼系統架構如圖二，是架設於 OpenNebula 系統之上，在 OpenNebula 系統中擁有兩種角色的節點，分別為 Front-End 和 Node。Front-End 是負責掌管雲端系統的主要節點，具有管理系統中每個 Node 的能力，自由的加入與刪除 Node，透過監控也可以了解每個 Node 的狀態，可以自由部署 VM 的映像檔至每個 Node 上以提供服務，可以說 Front-End 為雲端叢集的管理者。Node 是負責提供運算資源的節點，Node 之間不知道彼此的狀態，只有 Front-End 了解整體叢集的情形。在本系統中部署的 VM 是作為運算節點使用，在每台虛擬機中設計了相同的 DVC-SVC 快速轉碼器以提供轉碼服務。使用者所上傳的原始影像首先會經過 DVC Encoder 編碼，產生 DVC 影像壓縮格式後經由儲存設備暫存或是直接進行轉碼，再經由 Splitter 以 GOP 為單位分割 DVC 影像檔為多數的 DVC Chunk，之後 Load Balancer 決定初始化所使用的運算節點個數，並以負載平衡策略分配至各運算節點來進行轉碼至 SVC 影像壓縮格式。轉碼後的 SVC Chunk 將會回傳至 Merger 中整合為完整 SVC 壓縮檔，再傳遞至儲存設備中提供往後使用者下載，或是直接經由 Streaming Server 提供給不同的使用者使用。



圖二： DVC-SVC 轉碼雲整體系統架構

3.2 DVC-SVC 快速轉碼器

最簡單形式的轉碼器為將兩種不同影像格式的解碼端和編碼端串接 (Cascade)，以得到轉換形式的影像。圖三為本論文中 DVC-SVC 快速轉碼器架構，上半部為 DVC 解碼端，下半部則為 SVC 編碼端，我們利用了 DVC 解碼端作 MCI 所得到的移動向量來給予 SVC 編碼端作為編碼的參考依據。DVC 解碼端所使用的影像大小為 CIF，而 SVC 編碼端擁有兩層的空間可調性，Enhancement Layer 為 CIF 大小，Base Layer 為 QCIF 的大小，長寬正好為 Enhancement Layer 的一半，因此 DVC 的移動向量將給予 Enhancement Layer 直接使用，而 Base Layer 則使用原本一半的移動向量。



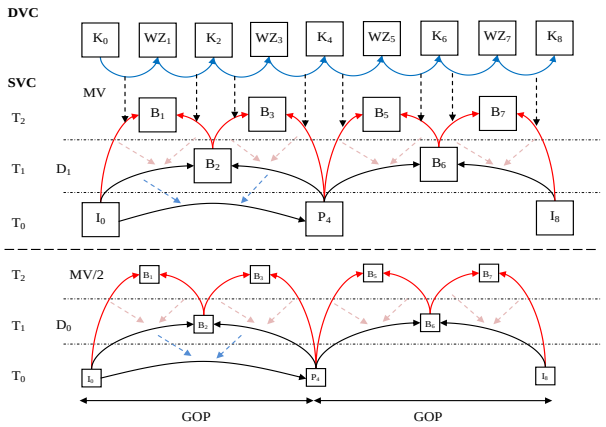
圖三： DVC-SVC 快速轉碼器架構

3.3 由上到下 (Top-down) 的 MV Prediction

本論文提出利用 DVC 的 Motion Compensated Interpolation (MCI) 做完內插後所得到的移動向量，提供給 SVC 的編碼端使用，在本論文中 SVC 使用的為時間

可調性與空間可調性。時間可調性的 SVC 其架構是階層式 B 畫面將從 DVC 獲得的移動向量對應至 SVC 階層式 B 畫面的架構，讓 DVC 的移動向量給 SVC 參考，而 DVC 的移動向量是對應給 SVC 的移動向量預測 (Motion Vector Predictor) 做改善，也就是將原先 SVC 的移動估測在做移動搜尋前，經由鄰近區塊所預測出來的移動向量預測，以 DVC 的移動向量取代，使得 SVC 的移動搜尋有更好的初始向量進行移動估測，而更甚能直接以 DVC 移動向量取代 SVC 移動向量，雖然 PSNR 會下降，但節省了更多時間。

在時間可調性方面，首先一個 GOP 的 DVC 可得到兩個向量，對應給 SVC Temporal Layer 為 2 (T_2)，如圖四所示，由 K_0 到 WZ_1 的移動向量可以對應給 I_0 到 B_1 當作參考，而 K_2 到 WZ_1 的移動向量可以對應給 B_2 到 B_1 當作參考，以上為 DVC 對應至 SVC Temporal Layer 為 2 (T_2) 的方式。接著 Temporal Layer 1 (T_1) 的部分，則是參考由 Temporal Layer 2 (T_2) 移動向量相加後的新移動向量，舉例來說，由 DVC 的 K_0 到 WZ_1 的移動向量與 K_2 到 WZ_1 的移動向量，將兩個移動向量相加後，對應給 I_0 到 B_2 的移動向量，依此類推。如此，形成了一個由上到下 (Top-down) 改善移動向量預測的架構。



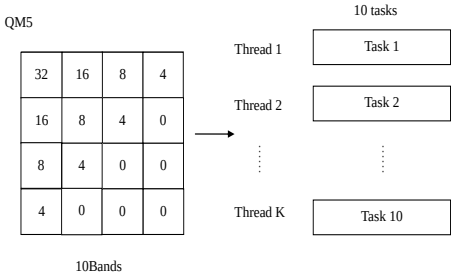
圖四： Spatial 與 Temporal 的 MV-reuse 示意圖

在空間可調性方面，SVC 使用兩層的可調性，Layer0 為 QCIF 的大小，Layer1 為 CIF 大小，將 DVC 的向量分別給予 SVC 的兩層編碼器，Layer0 給與 MV/2，Layer1 則給予原始的 MV。而在各層編碼器中則為跟時間可調性相同的由上到下的 MV Prediction 編碼方式。

另一方面，由於 DVC 所產生 Side Information 的方式是以 16x16 的巨區塊去做內插，在 DVC 的部分只能獲得 16x16 巨區塊的移動向量。因此，對應到 SVC 的部分，模式 (Mode) 16x16 以下以及模式 8x8 以上的預測移動向量會被更新改善。其餘的模式如 8x8 以下 (不包含 8x8) 的都照著原本 SVC 的架構進行編碼。

3.5 以平行化架構協助 DVC 解碼端時間改良

本論文使用 Band Level 的迴圈進行平行化。如圖五所示，以一個 DVC QM5 為例，總共會有 10 個 Bands，而且每個 Band 都是互相獨立的，因此可以利用 OpenMP 來進行平行化。而平行化的程度就必須考慮到 CPU 裡有多少執行緒可以使用。而在將迴圈平行化的時候，共有四個分割迴圈的執行方式：Dynamic、Guided、Static、Runtime，其中 Static 與 Dynamic 會讓 OpenMP 將迴圈的所有 Iteration 依序以指定 chunk_size 做切割成數個 Chunk。而 Dynamic 的 Chunk 是以動態的方式分配，能讓執行緒的負載更平衡，所以本論文使用 Dynamic 的方式進行平行化。



圖五：DVC QM5 為例的平行化架構

四、 實驗結果

4.1 MV-reuse 改良效能評估

本實驗採用了可調性視訊編碼標準軟體 JSVM 9.18 版本。分散式視訊編碼 (Distributed Video Coding, DVC) 則使用 JM 15.1 版本，相關設定則如表 I 中所示，JSVM 採用和 DVC 相同的量化參數，在實驗中進行了兩種不同動態程度的影像，並使用 DISCOVER[9]中的量化對應表進行編碼。在 JSVM 的 InterLayerPred 設定中，僅有 ILResidualPred 為開，ILMotionPred 與 ILModePred 皆為關。

表 I：實驗參數

Configuration				
Frame rate	15	Sequence		Foreman
FrameToBeEncode	8	Resolution		CIF
DVC				
QM	QM1-QM8	Foreman	QP	25、29、32、34 34、38、39、40
SVC				
Search Mode	Fast	Foreman	QP	25、29、32、34 34、38、39、40
Search Range	16	VLC		CAVLC
GOP	IBBBPBBBI	RDO		Off
Intra Period	8			
NumLayers	Spatial 2	ILMotionPred		No
	Temporal 3	ILModePred		

圖六表示 DVC-SVC 轉碼器使用 MV 作為 Predict 依據的編碼效能，使用影像為 Foreman。大小為 CIF (352x288)。橫軸代表使用不同量化矩陣與對應 QP 的編碼效能，垂直軸代表轉碼花費的時間。紫色長條代表 DVC 解碼時間，綠色長條代表原始 SVC 編碼時間，藍色長條代表改良 SVC 編碼並使用 All modes 的編碼時間，紅色長條代表改良 SVC 編碼並使用 16x16 mode 的編碼時間。可以觀察到使用了新的預測移動向量讓移動搜尋提早收斂之後，節省了編碼的時間，使用 All modes 與 16x16 mode 最多在 Foreman 影像可分別節省 5.58%與 43.92%的編碼時間，但如圖七所示，會降低影像品質 (分別為 0.027 dB 與 0.75 Db)。

4.2 DVC-SVC 雲端轉碼器效能分析

本節中將 MV-reuse 的 DVC-SVC 快速轉碼器，搭配 OpenNebula 系統和雲端轉碼的分配策略，實驗雲端

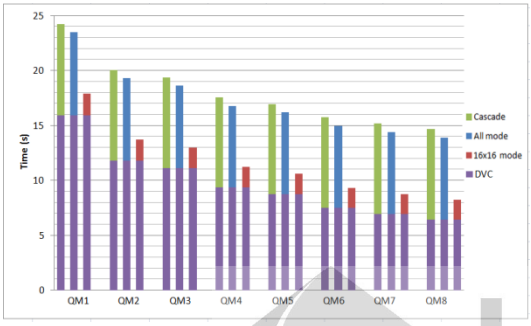
DVC-SVC 快速轉碼器的處理效能。實驗環境如表 II 所示。如圖八中所示，縱軸為每秒轉碼張數 (Frame Per Second, FPS) ，橫軸從左至右則為一台實體機上搭載著相異 processors 的虛擬機，並依序將 i7 實體機的 8 個 processors 使用完畢。右方參數 p 代表了虛擬機擁有的最大 processors 數，t 代表了在 DVC 解碼時使用了幾個 thread 來進行前述的平行 DVC 解碼。以 4px2t 為例，此 VM 最大擁有 4 processors 的能力，因此在一台 i7 實體機上最多可掛載兩台擁有 4 processors 的 VM，並在 DVC 解碼時使用了 2 threads 來幫助 DVC 解碼。由圖中可發現，使用了最小分割單位的 1px1t VM 擁有最佳的轉碼效能，而 2 processors 與 4 processors 的 VM 原本預估可以有 1processor VM 效能的 2 至 4 倍，但效能反而沒有較佳。這是因為在運行程式時，雖然擁有 2 processors 與 4 processors 的能力，但整體程式並未以平行化程式撰寫，因此仍然是由單一 processor 在運行編碼，直到運行平行化 DVC 解碼時才能有效運用其他處理器，此現象可以由圖十一中的 2px1t 與 2px2t 比較中發現。表 III 記錄了不同系統的轉碼效能，雲端 DVC-SVC 快速轉碼系統在處理 Foreman 影像時，一秒鐘約可以編碼平均 4.08 張。

表 II：DVC-SVC 雲端轉碼器實驗環境

實驗環境	Intel core i7 主機，CPU 3.4GHz，Memory 16G，500G 硬碟 Host OS: Ubuntu12.04
轉碼器	DVC-SVC Transcoder OS:Windows7

表 III：不同轉碼系統的轉碼率比較

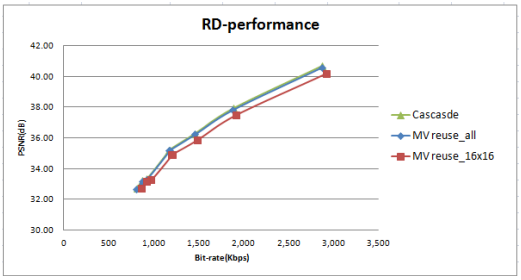
FPS	單機	1VM	2VM	4VM	8VM
QM1	0.52	0.52	1.06	2.02	3.05
QM2	0.67	0.61	1.27	2.46	3.86
QM3	0.71	0.64	1.29	2.52	3.82
QM4	0.83	0.66	1.35	2.64	4.05
QM5	0.87	0.68	1.39	2.70	4.38
QM6	0.99	0.70	1.43	2.80	4.37
QM7	1.06	0.71	1.45	2.86	4.47
QM8	1.12	0.72	1.47	3.00	4.64
平均	0.85	0.66	1.34	2.63	4.08



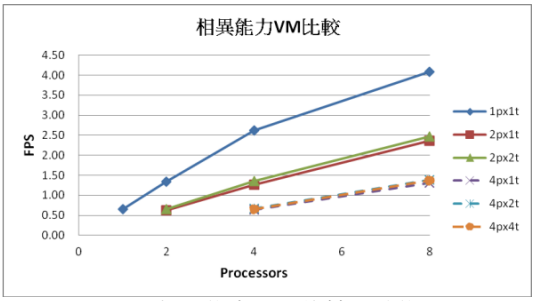
圖六： DVC-MV 作為 Predict 依據的轉碼時間長條圖

結論

本論文實現了雲端 DVC-SVC 快速轉碼系統的建置。首先改良了 DVC-SVC 轉碼器，實現了一個 Frame-based Transform Domain DVC 轉碼的包含空間與時間可調性的轉碼器，並使用 DVC 解碼端中 MCI 產生的移動向量來改善空間與時間可調性的 SVC 編碼中移動估測，而比起單純 DVC-SVC 串接的轉碼器，Foreman 影像在 16x16Mode 下最多可減少了 45.52%的編碼時間。並將此轉碼器和 OpenNebula 雲端系統搭配建置，以得到最大的吞吐量。比起在單一的 Intel core i7 的主機上進行使用 MV-reuse 的 DVC-SVC 轉碼器進行轉碼轉碼，雲端 DVC-SVC 快速轉碼系統在處理 Foreman 影像時比單機下平均可加速 396.03%的編碼時間，而可以依據使用者需求再擴增系統以增加轉碼率。



圖七：使用 DVC-MV 作為 Predict 依據的 RD 曲線圖



圖八：相異能力 VM 的轉碼效能圖

參考文獻

[1] W. Zhu, C. Luo, J. Wang, and S. Li, “Multimedia cloud computing,” IEEE Signal Process. Mag., pp. 59–69, 2011.

[2] ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC, “Advance Video Coding for Generic Audiovisual Services,” 2003.

[3] D. Slepian and J. K. Wolf, “Noiseless coding of correlated information source,” IEEE Trans. Inform. Theory, pp. 471-480, July 1973.

[4] A. Wyner and J. Ziv, “The rate-distortion function for source with side information at the decoder,” IEEE Trans. Inform. Theory, pp. 1-10, Jan 1976.

[5] B.Girod, A.M. Aaron, S. Rane, D. Rebollo-Monedero, “Distributed video coding performance” Proc. Of IEEE Int Conf. on Acoustics, Speech, and Signal Processing (ICASSP2006), Toulouse, France, May 2006.

[6] H. Schwarz, D. Marpe, T. Wiegand,“Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” IEEE Trans. on Circuits and Systems for Video Technology, pp.1013-1120, Sept. 2007.

[7] J. Guo and L. Bhuyan, “Load balancing in a cluster-based web server for multimedia applications,” IEEE Trans. Parallel Distrib. Syst., pp. 1321–1334, Nov. 2006

[8] B.A. Shirazi, A.R. Hurson, and K.M. Kavi, Scheduling and Load Balancing in Parallel and Distributed Systems. CS Press, 1995.

[9] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, M. Ouaret, “The DISCOVER Codec: architecture, techniques and evaluation, in: *Picture Coding Symposium*, ” Lisbon, Portugal, November 2007.