KIPS Tr. Comp. and Comm. Sys.
Vol.3, No.7 pp.231~240 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2014.3.7.231

A Dual Transcoding Method for Retaining QoS of Video Streaming Services under Restricted Computing Resources

Doohwan Oh† · Won Woo Ro††

ABSTRACT

Video transcoding techniques provide an efficient mechanism to make a video content adaptive to the capabilities of a variety of clients. However, it is hard to provide an appropriate quality-of-service(QoS) to the clients owing to heavy workload on transcoding operations. In light of this fact, this paper presents the dual transcoding method in order to guarantee QoS in streaming services by maximizing resource usage in a transcoding server equipped with both CPU and GPU computing units. The CPU and GPU computing units have different architectural features. The proposed method speculates workload of incoming transcoding requests and then schedules the requests either to the CPU or GPU accordingly. From performance evaluation, the proposed dual transcoding method achieved a speedup of 1.84 compared with traditional transcoding approach.

Keywords : Video Streaming, Video Transcoding, Job Scheduling, Heterogeneous Computing, GPGPU

동영상 스트리밍 서비스의 QoS유지를 위한 듀얼 트랜스코딩 기법

오 두 환† · 노 원 우††

요 약

스트리밍 서비스에 사용되는 트랜스코딩 서버는 동시에 다수의 트랜스코딩 요청을 처리한다. 하지만 트랜스코딩 연산의 부하가 높기 때문에 단일 서버가 수용할 수 있는 요청 개수는 제한적일 수밖에 없다. 본 논문에서는 단일 서버의 트랜스코딩 성능을 높이고자 CPU 기반 트랜스코더와 GPU 기반 트랜스코더를 동시에 사용하는 듀얼 트랜스코딩 방법을 제안한다. 듀얼 트랜스코딩 방법은 트랜스코딩 요청을 처리하기 전에 워크로드를 예측하여 해당 요청에 대한 QoS유지가 가능한지 판단한다. QoS유지가 가능하다고 판단되면 CPU 또는 GPU 트랜스코더 중 보다 적합한 타입의 장치에 작업을 할당함으로써 연산 자원의 효율성을 높인다. 성능 평가 결과 듀얼 트랜스코딩 기법은 기존 방식 대비 최대 1.84배의 성능 향상을 이루었다. 결과적으로 단일 서버가 보다 많은 사용자의 요청을 QoS 유지 하에 제공할 수 있게 되었다.

키워드 : 동영상 스트리밍, 동영상 트랜스코딩, 작업 스케줄링, 이종 연산 장치, GPGPU

1. 서 론

멀티미디어 데이터 압축 기술과 모바일 IT 기술의 발전으로 동영상 콘텐츠를 모바일 단말기로 전송하여 재생하는 스트리밍 서비스의 이용률이 급증하고 있다[1, 2]. 대표적인 비디오 데이터 압축 기술로는 H.263, MPEG-4 AVC, VC-2 등이 있다. MPEG-4 AVC는 일반적으로 H.264표준이라 불리며 약 1/250의 데이터 압축률을 자랑한다[3]. 이에 따라 IT전반에서 가장 인기 있는 동영상 압축 표준으로 자리 잡았다. 무선 통신 기술의 발전과 모바일 단말기의 하드웨어 성능 향상 역시 스트리밍 서비스의 인기를 높이고 있다. 네트워크 대역폭이 증가함에 따라 대용량의 동영상 데이터를 모바일 단말기로 끊김 없이 전송할 수 있게 되었으며, 모바일 단말기에는 저전력 H.264 하드웨어 디코더 칩이 탑재되어 동영상 디코딩 작업을 무리 없이 수행할 수 있게 되었다.

동영상 스트리밍의 인기에 편승해 다수의 인터넷 서비스 제공 업체들이 스트리밍 서비스를 도입하고 있다. 대부분의 업체들은 서비스 효율을 높이기 위해 원본 동영상을 사용자의 단말기에 적합한 포맷으로 변환해 주는 동영상 트랜스코딩 기술을 사용한다. 하지만 H.264압축 표준은 압축률만큼이나 높은 연산 복잡도를 요구한다[4-6]. 따라서 동영상을 H.264로 변환하기 위해서는 높은 컴퓨팅 파워가 요구된다.

※ 본 연구는 미래창조과학부가 지원한 2014년 정보통신·방송(ICT) 연구개발 사업의 연구결과로 수행되었음.
† 준 회 원 : 연세대학교 전기전자공학과 박사과정
†† 종신회원 : 연세대학교 전기전자공학부 부교수
Manuscript Received : March 11, 2014
First Revision : June 9, 2014
Accepted : June 16, 2014
* Corresponding Author : Won Woo Ro(wro@yonsei.ac.kr)

H.264의 연산을 고속으로 처리하기 위해 다양한 형태의 소프트웨어 트랜스코더가 연구 개발되고 있다. x264는 대표적인 H.264 인코딩 엔진으로 멀티코어 CPU연산을 지원한다[7]. 최근에는 NVIDIA사를 주축으로 GPU를 CPU처럼 일반적인 연산에 활용하는 연구가 활발히 이루어지고 있다[8]. 트랜스코딩 연산을 GPU에서 처리하는 연구도 진행되고 있다[9-11]. 대표적인 GPU 기반 H.264 인코딩 엔진으로는 NVIDIA사에서 출시한 cudaEncdoer가 있다[11]. cudaEncoder는 CPU외에 GPU를 추가 연산 장치로 사용하기 때문에 x264에 비해 빠른 연산 속도를 보인다.

본 논문에서는 단일 서버의 성능을 높이기 위해 CPU 기반 트랜스코더(x264 기반)와 GPU 기반 트랜스코더(cudaEncoer 기반)를 동시에 활용하는 듀얼 트랜스코딩 방법을 제안한다. 듀얼 트랜스코딩 방법은 워크로드 예측 기술과 스케줄링 기술을 포함한다. 트랜스코딩 요청에 대한 워크로드 예측을 위해 선형 회귀 분석 방법을 사용하였다. 예측한 워크로드를 근거로 해당 트랜스코딩 요청이 CPU 또는 GPU 기반 트랜스코더 중 어떤 트랜스코더로 처리해야 효율적인지 결정하는 스케줄링 정책을 제안하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 트랜스코딩 워크로드 스케줄링 정책 연구들에 대하여 살펴본다. 3장에서는 두 가지 종류의 트랜스코더의 특징과 장단점을 분석한다. 다음으로 4장에서는 본 논문에서 새롭게 제안하는 듀얼 트랜스코딩 방법을 소개한다. 5장에서는 성능 평가 결과를 분석한다. 마지막 6장에서 본 논문을 마무리한다.

2. 관련 연구

트랜스코딩 요청에 대한 부하 분산 스케줄링 연구는 대부분 복수의 서버간 로드 균형 유지를 목적으로 한다. 전통적인 스케줄링 기술로는 라운드 로빈 방식이 있다[12]. 이 방식은 사용자로부터 요청을 받으면 순서대로 서버에 할당하는 방식이다. 단순한 만큼 복잡한 연산이 필요 없지만, 서버간 부하 불균형이 발생할 여지가 큰 단점을 가진다. 그 외 최소 접속 방식과 가중치 기반 라운드 로빈 방식도 많이 사용된다[12]. 전자는 접속이 가장 적은 서버로 요청을 할당하는 방식이고, 후자는 각 서버마다 가중치를 주고 가중치에 따라 라운드 로빈으로 스케줄링 하는 방식이다.

Zhu 연구팀은 웹 서비스 요청들을 분석하여 클러스터 내 서버 중 해당 요청에 적합한 서버로 작업을 할당하는 방법을 제안하였다[13]. Li 연구팀은 Gage라 불리는 웹 서비스 요청 분산 시스템을 개발하여 서버 중 로드가 가장 적은 서버를 선택하여 할당하는 기법을 제안하였다[14]. 두 연구팀 모두 동영상 스트리밍 서비스를 포함한 모든 웹 서비스 요청에 대한 스케줄링 방법에 대하여 연구하였다. 반면에 Guo 연구팀은 동영상 스트리밍 서비스 요청에 한정한 스케줄링 기법에 대하여 연구하였다[1]. 이 기법은 워크로드를 사전에 예측하고 이를 기반으로 스케줄링 함으로써 서버 간 워크로드를 균형 있게 유지하는 방법이다. 본 논문에서 제안하는 방법은 앞선 연구들과 다르게 단일 서버 내부 연산 유닛 간의 로드 밸런싱에 대한 연구에 집중하였다.

3. 트랜스코더의 특징

본 장에서는 두 종류의 동영상 트랜스코더를 소개하고 그 특징을 관찰한다. 첫 번째 트랜스코더는 멀티코어 CPU 기반으로 동작하며, CPU-T라 한다. 다른 트랜스코더는 CPU 외에 추가로 GPU를 연산에 사용하며, GPU-T라 명명한다. Table 1은 트랜스코더 특성 분석에 사용된 시스템 사양 이다. 실험에 사용한 영상의 정보는 Table 2에 명시하였다. 총 3개의 샘플 영상이 사용되었으며 각 영상은 화면 움직임 정도(Activity)에 따라 High, Mid, 그리고 Low로 분류된다.

Table 1. System configuration

CPU	Model	Intel i7-3930K
	# of Core/Thread (clock speed)	6/12 (3.2 GHz)
GPU	Model	NVIDIA GeForce GTX 580
	# of CUDA Cores (clock speed)	512 (1544 MHz)
	GPU Memory Bandwidth	192.4 GB/sec
Main memory		32 GB
Operating System		MS Windows 7 64 bits

Table 2. Sample videos

Activity	Video	Resolution	Bitrate (Kbps)	Frame rate (FPS)
High	Dance Music Video	1280x720	3300	24
Mid	Drama (1)	1280x720	2500	24
Low	News (1)	1280x720	2700	30

3.1 CPU 기반 트랜스코더의 특징

CPU-T는 오픈 소스 미디어 프레임워크인 FFmpeg[15]의 트랜스코딩 기능을 기반으로 설계되었으며, CPU를 연산 장치로 활용한다. 입력 비디오를 H.264규격으로 압축하기 위한 인코딩 엔진으로 x264를 사용한다. x264엔진은 프레임 레벨 또는 슬라이드 레벨의 멀티쓰레딩을 지원하기 때문에 멀티코어 프로세스 시스템에서 빠른 처리 속도를 보인다. Fig. 1는 CPU-T구조에 대한 블록 다이어그램이다.

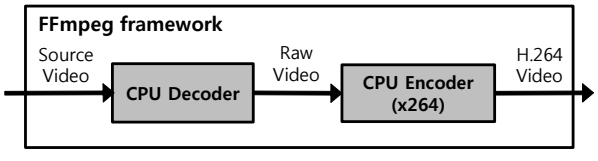


Fig. 1. System of CPU-T

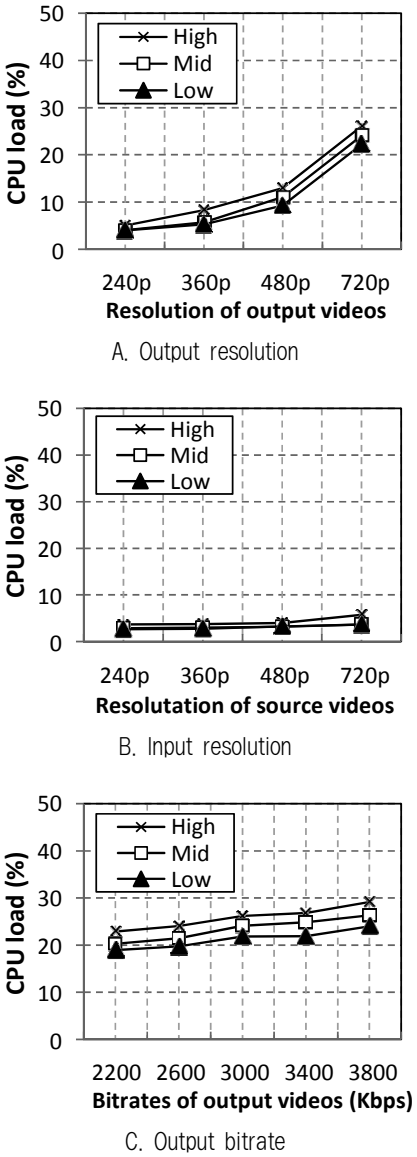


Fig. 2. CPU load on CPU-T transcoding process

CPU-T의 특징을 파악하기 위해 샘플 영상의 설정 값을 변경하면서 트랜스코딩 연산 시 발생하는 CPU로드를 측정하였다. Fig. 2A 그래프대로 출력 영상의 해상도가 증가할수록 샘플 영상 모두 CPU 로드가 증가하는 것을 확인할 수 있다. 샘플 중 화면 간 영상 변화가 가장 심한 High영상이 CPU 사용량이 제일 높았다. 240p 출력 영상에 대해서 5.0%, 720p 출력 영상에 대해서는 26.1%의 로드를 보였다. 다음으로 출력 영상의 해상도는 240p로 고정하고, 입력 영상의 해상도를 240p에서 720p로 증가시킬 경우 CPU로드 변화를 관찰하였다. 결과는 Fig. 2B에 보였다. 앞선 결과와 마찬가지로 High영상이 나머지 두 영상보다 CPU 사용량이 높았다. 입력 영상이 240p일 경우 CPU로드는 3.7%이었으나, 720p로 해상도가 증가할 경우 5.7%로 증가하였다. 마지막으로 출력 영상의 비트레이트를 2200 Kbps에서 3800 Kbps로 증가시킬 경우 CPU로드 변화를 관찰하였다. Fig. 2C에 보이듯 CPU로드는 High영상에 대해 22.2%에서 29.2%로 증가하였다. Mid와 Low영상들은 전체적으로 High보다 2% 내외 낮은 로드를 보였지만 비트레이트 변화에 따른 증감 현상은 비슷하였다.

위 결과를 통해 CPU-T 특징을 정리하면 다음과 같다.

특징 1: *출력 영상의 해상도, 즉 한 프레임 내 픽셀 수에 비례하여 트랜스코딩 연산에 대한 CPU로드는 증가한다.*

특징 2: *입력 영상의 해상도 크기에 비례하여 CPU로드가 증가한다. 단, 출력 해상도가 미치는 영향에 비해서 작다.*

특징 3: *출력 영상의 비트레이트는 CPU로드에 영향을 준다. 하지만 입력 영상 해상도의 영향에 비해서 작다.*

위 특징에 따르면 트랜스코딩 연산 처리 시 CPU로드를 결정하는 주요 속성은 출력 영상의 해상도인 것을 알 수 있다. 이는 전체 트랜스코딩 연산 시간 중 대부분이 영상을 출력 해상도 크기로 인코딩 하는 데 소비되기 때문이다.

3.2 GPU기반 트랜스코더의 특징

최근 트랜스코더들은 인코딩 연산을 GPU에서 처리하는 기능을 지원한다. GPU-T 역시 인코딩을 GPU로 처리하기 위해 cudaEncoder를 사용한다. GPU-T의 디코딩 연산은 CPU-T와 마찬가지로 FFmpeg기반의 프레임워크를 사용한다. 따라서 GPU-T는 트랜스코딩의 디코딩 연산은 CPU에서, 인코딩 연산은 GPU에서 처리한다. GPU-T의 구조는 Fig. 3에 보였다.

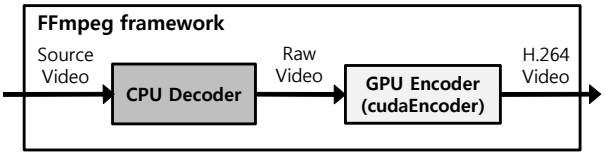
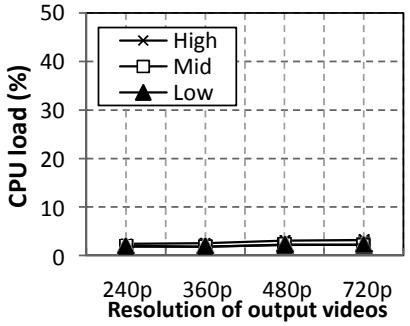


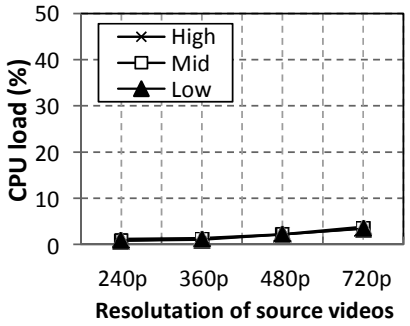
Fig. 3. System of GPU-T

GPU-T 구조에서 CPU와 GPU 사이의 메모리 대역폭은 성능을 제한하는 요소가 될 수 있다. 디코딩된 1080p 동영상의 한 프레임의 크기는 약 6.2 MB이다. 초당 30프레임이 시스템 메인 메모리에서 GPU로 전송될 경우 최소 186 MB/s 의 대역폭이 요구된다. Table 1의 GPU 사양을 보면 메모리 대역폭은 192.4 GB/s이며, PCI-E 3.0 x16 슬롯을 사용하므로 이론적으로 128 GB/s의 대역폭을 가진다. 따라서 GPU-T의 GPU는 초당 30 프레임 이상의 디코딩된 영상을 CPU로부터 지연 없이 전송 받아 처리할 수 있다.

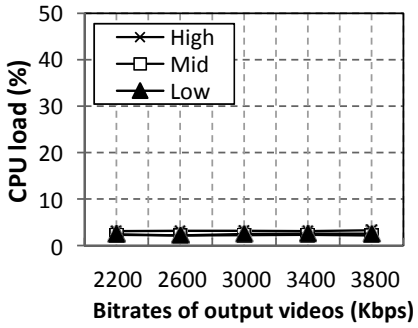
GPU-T의 특징 역시 CPU-T와 마찬가지로 출력 영상의 해상도와 비트레이트 그리고 입력 영상의 해상도 변화에 따라 연산 장치의 워크로드를 측정하여 관찰한다. 다만, GPU-T의 경우 CPU로드 외에 GPU로드가 추가로 측정된다. CPU로드 측정 결과는 Fig. 4에 보였다. 출력 영상의 해상도가 240p일 경우 CPU로드는 High가 약 2.4% 값을 보이



A. Output resolution

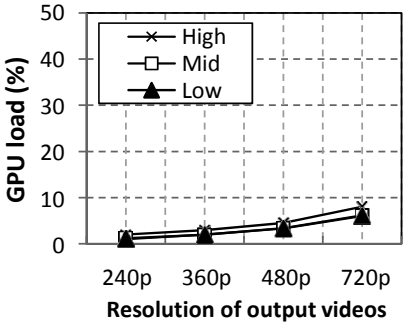


B. Input resolution

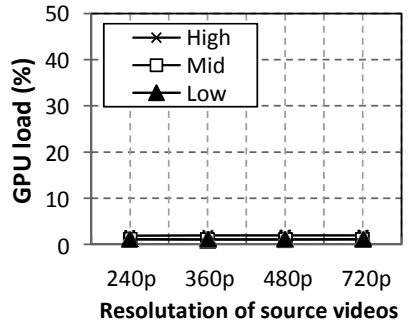


C. Output bitrate

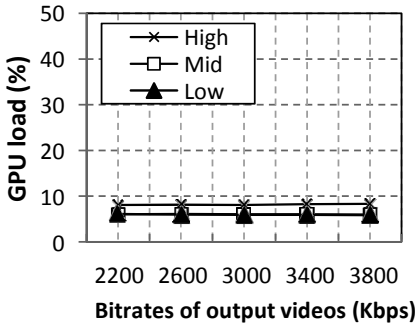
Fig. 4. CPU load on GPU-T trandcoding process



A. Output resolution



B. Input resolution



C. Output bitrate

Fig. 5. GPU load on GPU-T transcoding process

고, Mid와 Low는 각각 1.9%와 1.8%의 수치를 보였으며, 세 가지 영상 모두 해상도가 720p로 증가해도 거의 동일한 수치를 유지하고 있다(Fig. 4A 참고). 반면 입력 영상의 해상도가 240p일 경우 High의 CPU 로드는 1.2%였으나 720p해상도에 대해서는 3.8%로 증가하였다(Fig. 4B 참고). Mid와 Low는 약 0.8%에서 3.4%로 증가하였다. 마지막으로 출력 영상의 비트레이트를 2200 Kbps에서 3800 Kbps로 증가해 보았다. 이 경우 Fig. 4C 그래프에 나타나듯이 CPU 로드는 비트레이트에 상관없이 동일한 수치를 유지하였다. 세 가지 샘플 영상 중에서 High가 3.2%로 가장 높은 수치를 보이고 있다.

샘플 영상 트랜스코딩 시 GPU-T의 GPU로드 상태를 Fig. 5에 보였다. 출력 해상도가 증가할 경우 GPU로드는 증가하였다. 하지만 입력 해상도와 출력 비트레이트의 변화에는 영향을 받지 않고 동일한 수치를 유지하였다. 이 결과는 세 가지 샘플 영상 모두 동일하게 나타났다. 샘플 영상 중 가장 높은 GPU로드를 야기한 것은 High였으며 다른 두 샘플 영상에 비해 최대 2.2% 높은 수치를 보였다.

지금까지 결과로 GPU-T특징을 정의하면 다음과 같다.

특징 1: *출력 영상의 해상도가 커질수록 GPU로드가 크게 증가되며 그 증가분은 한 프레임 내의 픽셀 개수에 비례한다. CPU는 출력 해상도에 영향을 받지 않는다.*

특징 2: *입력 영상의 해상도가 커질수록 CPU로드는 증가하며 증가분은 한 프레임 내의 픽셀 개수에 비례한다. GPU는 입력 해상도에 영향을 받지 않는다.*

특징 3: *출력 영상의 비트레이트는 CPU와 GPU로드에 거의 영향을 미치지 않는다.*

정의된 특징에 따르면 출력 영상의 해상도 크기는 GPU로드에 영향을 미치며 입력 영상의 해상도 크기는 CPU로드

에 영향을 준다. 그러나 그 영향력은 CPU-T가 받는 것에 비하면 작으며 그 이유는 GPU-T는 단일 트랜스코딩 프로세스를 CPU와 GPU로 분산하여 처리하기 때문이다.

3.3 복수 트랜스코딩 요청 처리의 한계

스트리밍 서비스에 사용되는 트랜스코딩 서버는 일반적으로 동시에 한 개 이상의 트랜스코딩 요청들을 처리해야 한다. 따라서 본 장에서는 단일 트랜스코딩 서버가 다수의 트랜스코딩 요청들을 처리할 경우 CPU와 GPU로드의 변화 추이를 관찰하고 이를 통해 CPU-T와 GPU-T의 한계점을 분석한다.

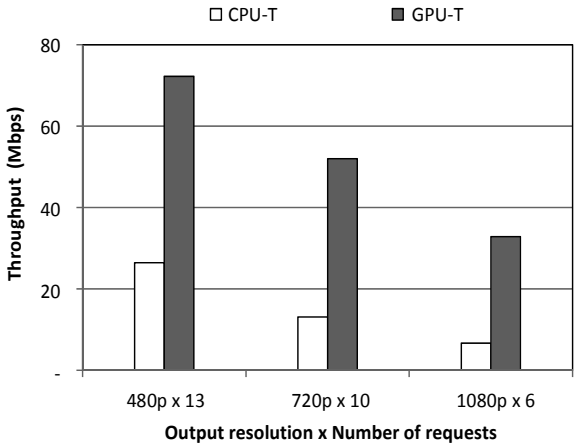


Fig. 6. Throughput for multiple requests

첫 번째 실험은 출력 해상도별로 서버가 복수의 트랜스코딩 요청을 처리하는 데 걸린 시간을 측정하였다. 결과는 Fig. 6에 보였다. 출력 해상도가 480p이고 요청 개수가 13개일 때 GPU-T는 약 71 Mbps의 처리 속도를 보인 반면 CPU-T는 약 24 Mbps를 보였다. 출력 해상도가 커질수록 두 트랜스코더의 성능은 떨어졌고, 성능 차이는 점점 더 벌어졌다. 1080p 출력 영상 6개를 처리할 때 GPU-T는 약 32Mbps, CPU-T는 약 6Mbps를 보였다.

GPU-T가 CPU-T보다 단위시간당 데이터 처리 성능이 높지만, CPU와 GPU자원을 동시에 활용하기 때문에 두 연산 자원 간에 사용량 불균형을 초래할 수 있다. Fig. 7은 출력 해상도와 요청 개수별 GPU-T의 CPU와 GPU로드 상태를 보였다. 출력 해상도 720p 이하일 경우 CPU자원 부족으로 병목현상이 일어난다. 하지만 1080p에서는 오히려 GPU 자원 부족으로 병목현상이 발생한다. 따라서 CPU와 GPU로드간에 균형을 유지할 수 있는 정책이 필요하다.

4. 듀얼 트랜스코더의 스케줄링 기법

듀얼 트랜스코더는 CPU만 사용하는 CPU-T와 CPU와 GPU를 동시에 사용하는 GPU-T를 단일 시스템에 설치하여 트랜스코더로 사용한다. 트랜스코딩 요청을 받으면 워크로

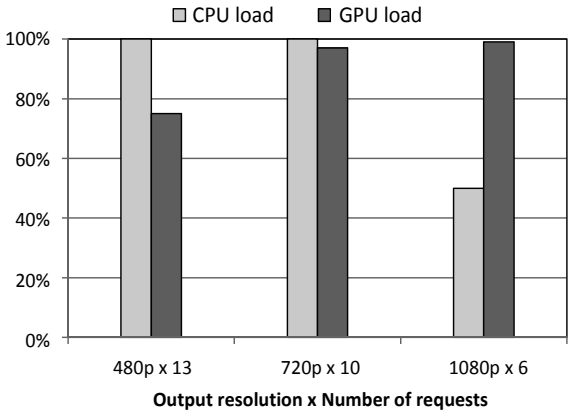


Fig. 7. CPU and GPU load on multiple requests

드 예측을 통해 두 트랜스코더 중 한 곳에 작업을 할당하여 처리한다. 본 장에서는 듀얼 트랜스코더 기법에 대하여 설명한다. 우선 CPU-T와 GPU-T의 워크로드를 선형회귀분석을 통해 예측하는 방법을 설명한다. 이어서 예측한 워크로드를 기반으로 트랜스코딩 요청을 스케줄링 하는 정책을 설명한다.

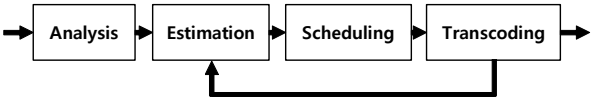


Fig. 8. Control flow of dual transcoding method

제안하는 스케줄링 정책의 동작흐름은 Fig. 8과 같다. 서버가 트랜스코딩 요청을 받으면 우선 입력 영상의 속성과 출력 영상의 설정 값을 분석한다. 다음으로 분석한 수치를 바탕으로 워크로드를 예측한다. 이때 현재 서버의 CPU와 GPU로드 상태를 피드백으로 받아 참고한다. 현재 요청에 대한 로드가 예측되면 CPU-T에서 구동할지 아니면 GPU-T에서 구동할지 스케줄링 된다. 이후 결정된 트랜스코더를 통해 트랜스코딩 작업이 처리된다. 작업이 시작되면 CPU와 GPU의 로드 상태를 워크로드 예측 단계로 피드백 하여 다음 요청에 대한 워크로드 예측 시 활용한다.

4.1 트랜스코딩 워크로드의 선형회귀분석

본 논문은 트랜스코딩 요청의 워크로드를 예측하기 위해 선형 회귀 분석을 사용한다. 이 기법은 다른 기법들에 비해 연산 복잡도가 낮아 적용에 대한 오버헤드 부담이 낮은 장점을 가진다. 선형 회귀 모델의 기본 수식은 다음과 같다.

$$Y = mX + b \tag{1}$$

X는 독립변수, Y는 종속변수를 의미한다. 변수 m과 b는 선형 회귀 분석을 통해 얻어진다. 독립변수로부터 예측되는 종속변수의 분산 비율을 의미하는 결정계수는 R^2라 하며 다음과 같다.

$$R^2 = (\frac{1}{n}) \times \sum [(X_i - X) \times (Y_i - Y)] / (\sigma_X \times \sigma_Y)^2 \quad (2)$$

우리는 3장에서 트랜스코더의 특징을 정의하였다. 총 3종의 샘플 영상을 사용하였으며 모두 동일한 경향을 보였다. 그 중 High샘플이 모든 경우에 있어 가장 높은 연산 로드를 야기하였다. 그 이유는 영상 내의 움직임이 많아 계산복잡도가 다른 샘플들에 비해 높기 때문이다. 본 논문의 목적은 트랜스코딩 서버가 스트리밍 서비스의 QoS유지하에 최대 성능을 보이는 것이다. 따라서 연산복잡도가 높은 샘플을 기준으로 예측 모델을 정의할 것이다. 즉, High샘플을 대표 샘플로 선정하고 선형회귀모델을 정의할 것이다. 이와 같은 접근법은 워크로드를 방어적으로 예측하여 성능 측면에서 손해를 볼 수 있지만 안정적인 시스템 운영을 보장할 수 있는 이점을 가진다.

앞서 언급했듯이 출력 영상의 프레임 내 픽셀 수에 비례하여 CPU-T와 GPU-T 워크로드가 결정된다. 따라서 수식 (1)에서 독립변수 X는 출력 영상의 해상도가 되며, Y는 종속변수로 CPU또는 GPU의 워크로드가 된다. 입력 영상의 해상도와 출력 영상의 비트레이트는 영향력이 상대적으로 적어 차후에 오차로 처리한다.

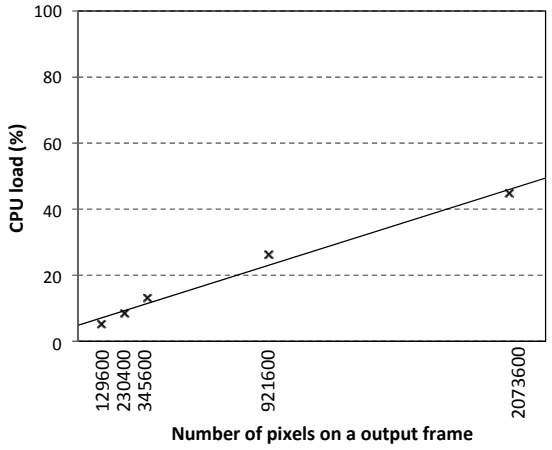


Fig. 9. Linear regression line for CPU load on CPU-T transcoding

우선 CPU-T의 워크로드 모델을 정의하기 위해 3.1장에서 실험한 결과를 수집하고 분석하였다. 이를 통해 m은 19.95 x 10-6, b는 4.6366의 값을 얻을 수 있다. 따라서 출력 해상도 크기에 따른 CPU-T의 CPU로드 선형 회귀 모델은 다음과 같다.

$$Y = (19.95 \cdot 10^{-6} \cdot X) + 4.6366 \quad (3)$$

선형 회귀 수식 (3)의 결정계수 R2는 0.98의 값을 갖는다. 이는 CPU-T의 워크로드와 출력 영상의 픽셀 수의 관계가 수식 (3)를 통해 작은 오차를 가지고 모델링 되었음을 의미한다. Fig. 9에 수식 (3)의 선형회귀모델 그래프를 보였다.

다음으로 출력 영상의 해상도 크기에 대한 GPU-T의 워크로드를 선형회귀모델로 정의하였다. GPU-T의 경우 GPU만 출력 영상의 해상도에 영향을 받기 때문에 GPU로드에 대한 모델을 정의하였다. 이 역시 CPU-T의 경우와 마찬가지로 수식 (1)을 기반으로 정의된다. 3.2장의 실험결과를 토대로 회귀계수를 구하면 m은 7.09x10-6, b는 1.4644가 된다. 따라서 출력 영상의 해상도에 따른 GPU워크로드의 선형회귀모델은 다음과 같다.

$$Y = (7.09 \cdot 10^{-6} \cdot X) + 1.4644 \quad (4)$$

선형회귀모델 수식 (4)의 결정계수 R2는 0.99로 CPU-T보다 작은 오차 범위를 보인다. GPU-T의 경우 CPU워크로드는 입력 영상의 프레임 내 픽셀 수에 비례하기 때문에 이에 대한 선형회귀모델 역시 정의된다. 3.2장의 실험결과를 토대로 GPU-T의 CPU워크로드 모델을 정의하며 수식은 다음과 같다.

$$Y = (1.64 \cdot 10^{-6} \cdot X) + 0.9569 \quad (5)$$

이 선형회귀모델의 결정계수는 1.0000을 보였다. GPU-T의 선형회귀모델 그래프는 Fig. 10에 보였다.

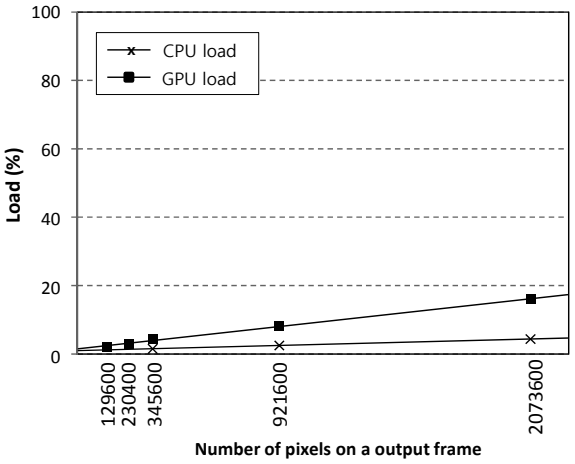


Fig. 10. Linear regression line for CPU and GPU load on GPU-T transcoding

4.2 트랜스코더 워크로드 예측 모델

듀얼 트랜스코딩 기법은 트랜스코딩 요청에 대하여 CPU-T와 GPU-T모두의 워크로드를 예측한다. CPU-T의 CPU로드 예측 모델은 다음과 같다.

$$P_{CPU-C} = L_{CPU} + E_{CPU-C} \quad (6)$$

L_{CPU}는 현재 서버의 CPU로드 상태를 의미한다. E_{CPU-C}는

해당 요청이 처리될 때의 CPU로드 예측 값이다. E_{CPU-C}는 선형 회귀 모델 수식 (3)을 기초로 산출된다. 다만 수식 (3)도 자체 오차가 있기 때문에 이를 고려한 예측이 필요하다. 예측 결과에 오차가 발생하여 서버의 가용치 이상의 요청이 할당되는 것을 방지하기 위해 최대오차로 예측한다.

CPU-T의 CPU로드에 대한 선형회귀모델의 결정계수는 0.98 이다. 따라서 최대 2%의 오차가 발생할 수 있다. 발생할 수 있는 최대오차를 고려하여 CPU-T의 CPU로드 예측하면 다음과 같다.

$$E_{CPU-C} = Y \bullet 1.02 \tag{7}$$

Y는 수식 (3)를 통해 얻어진다.

GPU-T의 경우 CPU와 GPU로드를 예측해야 한다. CPU로드 예측 모델은 다음과 같다.

$$P_{CPU-G} = L_{CPU} + E_{CPU-G} \tag{8}$$

E_{CPU-G}경우 입력 영상의 해상도에 영향을 받으므로 수식 (5)을 통해 예측한다. 수식 (5)의 결정계수는 1.0000으로 측정 범위 내에서 오차가 없다고 할 수 있다. 따라서 GPU-T의 CPU로드 예측치 E_{CPU-G}는 수식 (5)의 Y가 된다.

GPU-T의 GPU로드의 예측 모델은 다음과 같다.

$$P_{GPU} = L_{GPU} + E_{GPU} \tag{9}$$

L_{GPU}는 현재 서버의 GPU로드 상태를 의미한다. E_{GPU}는 해당 요청에 대한 GPU로드 예측 결과이다. E_{GPU}는 출력 영상의 해상도에 영향을 받는다. 따라서 수식 (4)을 기반으로 예측 가능하다. 수식 (4)의 결정계수는 0.9959이므로 최대 0.41%의 오차가 발행할 수 있다. 결과적으로

$$E_{CPU-C} = Y \bullet 1.0041 \tag{10}$$

Y는 수식 (4)을 통해 구한다.

4.3 트랜스코딩 요청 스케줄링 정책

본 논문에서 제안하는 스케줄링 정책의 목표는 CPU와 GPU 두 연산 자원의 로드 밸런싱이 유지될 수 있도록 트랜스코딩 요청을 스케줄링 하는 것이다. 따라서 CPU와 GPU 예측 결과인 P_{CPU}와 P_{GPU}의 상태를 토대로 트랜스코딩 요청을 스케줄링 한다.

다음은 세 가지로 구분한 CPU와 GPU로드 상태이다.

상태 1)

P_{GPU} > 100%,
P_{CPU-C} <= 100%.

상태 2)

P_{GPU} <= 100%,
P_{CPU-G} <= 100%,
P_{CPU-C} > 100%.

상태 3)

P_{GPU} <= 100%,
P_{CPU} <= 100%

상태 1)은 현재 트랜스코딩 요청을 CPU-T가 처리한다. 상태 2)는 GPU-T가 처리한다. 상태 3)의 경우 다음 조건에 따라 트랜스코더를 결정한다.

$$|P_{CPU-C} - L_{GPU}| < |P_{CPU-G} - P_{GPU}| \tag{11}$$

수식 (11)이 참일 경우 CPU-T가 처리한다. 거짓일 경우 GPU-T가 처리한다. 상태 1부터 3에 해당되지 않을 경우 트랜스코딩 요청을 현재 서버가 처리할 수 없는 상황이다.

제안하는 예측 모델은 스케줄링의 연산 오버헤드를 최소화하기 위해 간단한 수식으로 구성하였으며 오차가 발생하더라도 실제 결과가 예측 결과보다 크지 않도록 설계하였다. 이를 통해 서버가 가용한 워크로드 이상의 요청을 할당받는 일이 없도록 방지 하였다. 또한 오차는 다음 트랜스코딩 요청을 분석하기 전에 보정되기 때문에 차후의 예측에 영향을 주지 않는다.

5. 실험 결과 및 분석

본 장에서는 듀얼 트랜스코딩 방법의 성능을 검증하고자 단일 서버에서 복수의 트랜스코딩 요청을 처리할 때 단위 시간당 데이터 처리량을 측정하였다. 트랜스코딩에 사용한 샘플 영상에 대한 정보는 Table 3에 보였다.

Table 3. Sample videos

ID	Video (Codec)	Resolution	Bitrate (Kbps)	Frame rate (FPS)
A	Drama2 (H.264)	1280x720 (HD)	1500	24
B	News2 (H.264)	1280x720 (HD)	2500	30
C	Movie (H.264)	1920x810 (Full HD)	5200	24
D	TV show (XVID)	1280x720 (HD)	3300	30
E	Sports (WMV)	1280x720 (HD)	3000	30

5.1 워크로드 예측 결과 평가

본 논문에서 제안하는 워크로드 예측 기법을 평가하기 위해 다섯 가지 영상에 대한 CPU와 GPU로드 측정값과 예측 결과(Est.)를 비교하였다. Fig. 11A는 CPU-T의 CPU로드

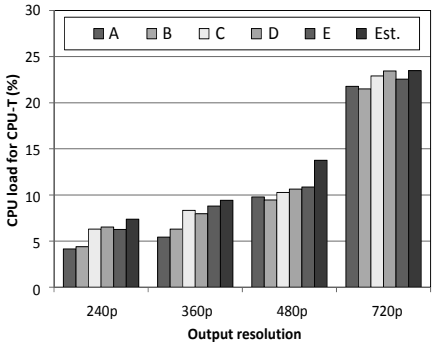
예측 모델에 대한 비교 결과이다. 다섯 가지 샘플 영상의 CPU로드치가 예측 결과인 Est.과 최대 3%의 오차를 보였지만 예측 결과를 넘지는 않았다. 다음은 GPU-T의 CPU로드 예측 모델의 성능 비교 결과이다. Fig. 11B에 보였듯 실제 측정 결과와 예측 결과가 최대 1.9% 차이가 났지만, 예측 결과 이상의 수치를 보이지 않는 것을 확인하였다. 다음으로 Fig. 11C는 GPU-T의 GPU로드 예측 결과 비교 그래프이다. 샘플 A가 720p 해상도로 변환될 때 예측값보다 3.5% 낮은 수치를 보였다. 이는 전체 실험 중 가장 큰 오차였다. 하지만 이 역시 측정값이 예측값을 넘지 않으므로 트랜스코딩 서버 가용치 이상의 작업을 할당 받는 경우는 발생하지 않는다.

5.2 트랜스코딩 요청 처리 성능 비교

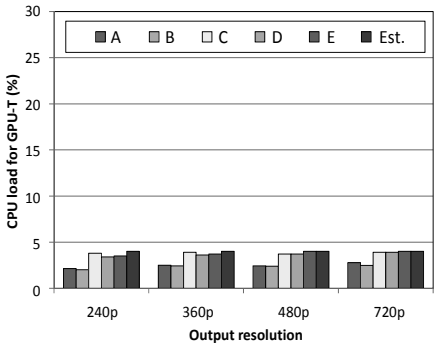
복수 트랜스코딩 요청에 대한 듀얼 트랜스코딩 기법(Dual-T)의 처리 성능을 평가하기 위한 실험을 진행하였다. 비교 대상으로는 CPU-T만 사용한 서버(CPU-T)와 GPU-T만을 사용한 서버(GPU-T)를 사용하였다. 성능 비교를 위해 측정한 항목은 throughput 및 CPU와 GPU로드 상태이다. 총 두 가지 트랜스코딩 상황에 대한 성능 평가를 진행하였으며 결과를 Fig. 12에 보였다. 첫 번째 상황은 1080p 영상으로 변환하는 요청 6개와 240p 영상 변환 요청 6개를 동시에 처리한 경우이다. 단, 첫 번째 상황(S.1)의 요청 순서는 1080p 영상 변환 6개 후 240p 영상 변환이 요청된다. 두 번째 상황(S.2)의 경우 반대로 240p 변환 요청 이후 1080p 영상 변환이 요청된다.

CPU-T는 S.1에 대해 1080p변환 1개 이상을 할당 받지 못하고 대신 이어지는 240p 변환 요청 6개를 할당 받아 총 1080p 요청 1개와 270p 요청 6개를 동시에 처리 가능했다. 처리 성능은 37 Mbps를 보였다. 이때 CPU의 로드는 94.7%이고 GPU는 사용하지 않는다. GPU-T의 1080p 출력에 대한 요청 6개를 동시에 처리할 수 있으며, 이후 240p 요청은 처리할 수 없었다. 처리 성능은 38 Mbps에 근접하다. 이때 CPU로드 51%, GPU 로드는 99%를 보인다. Dual-T는 동시에 1080p 출력 요청 6개, 270p 출력 요청 6개를 처리할 수 있다. 즉 동시에 12개의 요청을 처리할 수 있으며 처리 성능은 68 Mbps를 보인다. 이때 CPU로드는 95.8%, GPU로드는 99%로 가용한 모든 연산자원을 거의 대부분 사용하고 있다. 따라서 처리 성능도 기존 두 트랜스코더 대비 1.8배 높게 나타나고 있다.

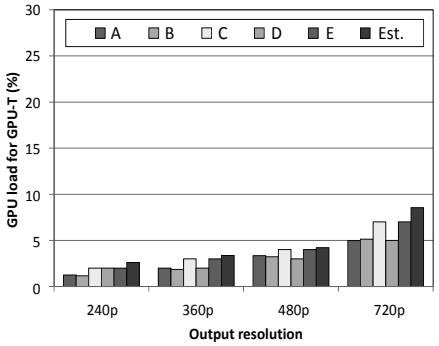
두 번째 상황에서 CPU-T는 240p 변환 요청 6개를 모두 할당 받았으나 이후 1080p 변환 요청은 1개까지만 할당 받아 처리할 수 있었다. GPU-T는 240p 변환 요청 6개와 1080p 변환 요청 3개를 처리할 수 있었다. 총 9개의 요청을 처리할 수 있었으며 처리 성능은 57 Mbps를 보였다. 이는 S.1보다 50% 향상된 수치이다. Dual-T의 경우 S.1과 동일하게 12개의 요청을 모두 처리하였으며, 성능은 69 Mbps를 보인다. S.2에서 GPU-T대비 Dual-T의 성능 향상 폭은 1.20배로 S.1에 비해 낮은 수치를 보였다. 이는 240p 요청에 대한 GPU-T의 CPU와 GPU로드 간 균형이 1080p 요청보다 적절했기 때문이다.



A. CPU load for CPU-T



B. CPU load for GPU-T

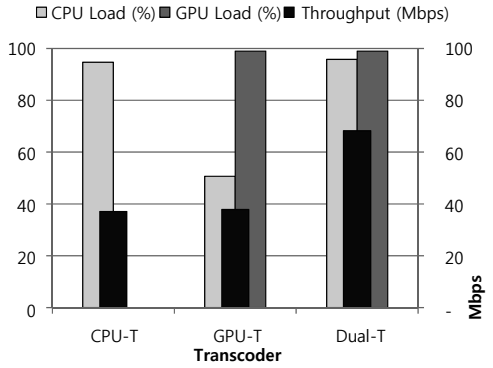


C. GPU load for GPU-T

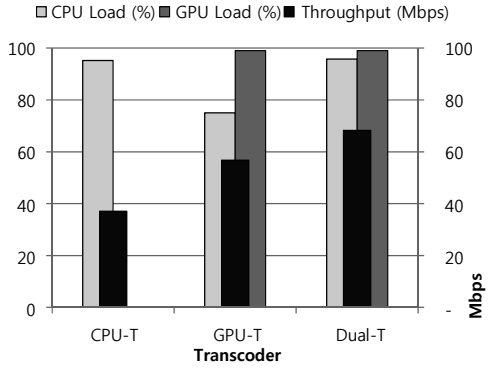
Fig. 11. Comparison estimation results and measured results

6. 결 론

본 논문에서는 듀얼 트랜스코딩 기법을 새롭게 제안하여 단일 서버의 단위시간당 데이터 처리량을 높였다. 이 기법은 크게 두 단계로 이루어진다. 첫 단계는 요청받은 트랜스코딩 연산을 실행하기 전에 선형회귀분석을 통해 워크로드를 예측한다. 이를 통해 해당 서버의 컴퓨팅 파워 이상의 작업이 할당되는 것을 방지하였으며, 결과적으로 트랜스코딩 연산 지연으로 스트리밍 서비스의 질이 떨어지는 것을



A. S.1



B. S.2

Fig. 12. Transcoding performance comparison

막았다. 다음으로 예측 결과를 근거로 트랜스코딩 요청을 CPU 기반 트랜스코더 또는 GPU기반 트랜스코더로 스케줄링 한다. 이때 가능한 CPU와 GPU자원의 사용률이 균형 있게 유지될 수 있도록 스케줄링 하여 자원 활용률을 최대화 하였다.

듀얼 트랜스코딩 기법의 예측모델은 연산 부담이 적지만 서버의 연산 유닛이 변경될 경우 다시 정의해야 하는 단점이 있다. 후속 연구에서는 연산 장치의 성능 또한 고려하여 하드웨어 시스템에 독립적인 예측 모델 정의에 대하여 진행할 계획이다.

Reference

[1] Jiani, Guo and Laxmi Narayan Bhuyan, "Load Balancing in a Cluster-Based Web Server for Multimedia Applications," *IEEE Transactions on Parallel and Distributed Systems*, Vol.17, No.11, pp.1321-1334, November, 2006.

[2] C. Sampath Kannangara, Iain. E. Richardson, and A. J. Miller, "Computational Complexity Management of a Real-Time H.264/AVC Encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.18, No.9, pp.1191-1200, September, 2008.

[3] Recommendation H.264: Advanced Video Coding for Generic Audiovisual Services ITU-T, 2003.

[4] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance and complexity," *IEEE Circuits Syst. Mag.*, Vol.4, pp.7-28, 2004.

[5] Tung-Chien Chen et. al., "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder," *IEEE T. CIRC. SYST. VID.*, Vol.16, No.6, pp.673-688, June, 2006.

[6] Shenggang Chen et. al., "Mapping of H.264/AVC Encoder on a Hierarchical Chip Multicore DSP Platform", in *HPCC'10*, pp.465-470, 2010.

[7] Laurent Aimar et. al., x264, [Online]. Available: http://www.videolan.org/developers/x264.html

[8] NVIDIA Corporation, NVIDIA CUDA Programming Guide 4.2, Edition 2012.

[9] Wei-Nien Chen and Hsueh-Ming Hang, "H.264/AVC Motion Estimation Implementation on Compute Unified Device Architecture (CUDA)", in *ICME'08*, pp.697-700, 2008.

[10] "Video Capture, Encoding, And Streaming in a Multi-GPU System", NVIDIA Technical Brief, 2010.

[11] NVIDIA Corporation, NVIDIA CUDA VIDEO ENCODER LIBRARY, Edition 2010.

[12] B. A. Shirazi, A. R. Hurson, and K. M. Kavi, "Scheduling and Load Balancing in Parallel and Distributed Systems," *Wiley-IEEE Computer Society Press*, 1995.

[13] H. Zhu, H. Tang, and T. Yang, "Demand-Driven Service Differentiation in Cluster-Based Network Servers," in *Proc. IEEE INFOCOM*, 2001.

[14] C. Li, G. Peng, K. Gopalan, and T. Chiueh, "Performance Garantees for Cluster-Based Internet Services," in *Proc. 23rd Int'l Conf. Distributed Computing Systems (ICDCS '03)*, May, 2003.

[15] F. Bellard et al., FFmpeg. [Online]. Available: http://www.ffmpeg.org

오 두 환

e-mail : ohdooh@yonsei.ac.kr

2007년 경희대학교 전자공학과(학사)

2010년 연세대학교 전기전자공학과(석사)

2010년~현 재 연세대학교 전기전자공학과 박사과정

관심분야 : 컴퓨터 시스템, 병렬 프로세싱, 동영상 스트리밍

노 원 우

e-mail : wro@yonsei.ac.kr

1996년 연세대학교 전기공학과(학사)

1999년 University of Southern California (석사)

2004년 University of Southern California (박사)

2003년~2004년 Apple Computer Inc. 인턴 연구원

2004년~2007년 California State University 전기 및 컴퓨터공학과 조교수

2006년~2007년 ARM Inc. 소프트웨어 엔지니어

2007년~현 재 연세대학교 전기전자공학부 부교수

관심분야 : 고성능 마이크로프로세서 디자인, 컴파일러 최적화, 임베디드 시스템 디자인 등