프레임 분할 기반 병렬화 H.264/AVC 디코더

*김원진 *박주열 **정기석

*한양대학교 전자컴퓨터통신공학과 **한양대학교 융합전자공학부

*kwonjin97@gmail.com *radarpark@hanyang.ac.kr **kchung@hanyang.ac.kr

Frame Partition based Parallelization of H.264/AVC decoder

*Won-Jin Kim *Joo-Yul Park **Ki-Seok Chung

Hanyang University

요 약

고해상도의 동영상 서비스가 보편화 되면서 동영상을 빠르게 처리를 위한 연구가 활발히 이루어 지고 있다. 그리고 멀티코어 프로세서의 사용이 증가 하고 멀티코어 시스템에서 H.264/AVC 디코더를 구현하기 위하여 다양한 병렬화 방법이 제안되고 있다. 하지만 H.264/AVC디코더의 병렬화를 진행하는 과정에서 각 스레드에서 처리하는 데이터의 처리 시간 차이로 인하여 스레드의 동기를 확인 해야 한다. 이로 인하여 병렬화를 통한 성능 향상의 걸림돌이 된다. 우리는 이러한 병렬화 과정에서 발생하는 문제점을 고려하여 효과적으로 H.264/AVC 디코더를 병렬화 하는 방법에 대하여 연구하였다. 우리가 제안하는 Frame Partition based Parallelization (FPP) 방법은 프레임을 매크로 블록 묶음으로 나누어 병렬화 한다. 그리고 병렬화 과정에서 스레드를 처리하는 방법을 개선하여 성능을 향상 시켰다. 본 논문에서는 FFmpeg H.264/AVC 디코더를 이용하여 실험 하였고 인텔 쿼드 코어 기반의 멀티코어 시스템에서 멀티 스레드로 구현하였다. 우리는 FPP 방법을 적용하여 병렬화 방법 적용 전 H.264/AVC 디코더와 비교하여 최대 53%의 성능 향상을 보였다.

1. 서론

디지털TV가 보편화 되고 사용자들은 고해상도의 영상 서비스를 요구하고 있다. 대표적으로 H.264/AVC는 현존하는 가장 압축률이 우수한 성능의 비디오 부호화 표준으로, 디지털 방송, 멀티미디어 플레이어, 화상회의 등 멀티미디어 서비스 분야에서 많이 사용되고 있고 다양한 연구가 이루어 지고 있다. 이러한 H.264/AVC 코덱을 처리하기 위해서는 고성능의 프로세서 사용이 필요하다. 기존의 싱글 코어 기반의 프로세서는 클럭 속도를 올려서 성능을 향상시키기 때문에 프로세서의 발열 및 소비전력의 증가로 인하여 성능 향상의 한계에 직면 하였다. 이러한 문제를 해결하기 위하여 프로세서 코어 수를 늘리고 병렬 처리 이용하는 멀티 코어 시스템의 연구가 지속적으로 이루어 지고 있다. 그러나 기존의 싱글 코어 프로세서 시스템에서 사용한 프로그래밍 모델은 멀티 코어 시스템의 성능을 충분히 활용할 수 없다. 따라서 프로그램을 멀티 코어 시스템을 효과적으로 사용하기 위하여 병렬프로그래밍으로 바꾸어야 한다. 그러므로 고해상도에서 H.264/AVC디코더의 성능을 향상 시키기 위하여 효과적인 병렬화 방법이 필요하다.

대표적인 H.264/AVC 디코더 병렬화 방법으로 태스크 레벨 병렬화 방법과 데이터 레벨 병렬화 방법이 있다. 태스크 레벨 병렬화 방법은 하나의 작업을 여러 개의 작업으로 나누어 각 스레드에서 나누어서 수행하는 방법이다. 그리고 데이터 레벨 병렬화 방법은 H.264/AVC 데이터를 병렬화가 가능하도록 나누어 여러 스레드에서 동시에 처리하는 방법이다. 그런데 태스크 레벨 병렬화 방법은 H.264/AVC 디코더 기능 별로 처리 시간 다르기 때문에 성능 향상에 어려움이 있다. 그리고 데이터 레벨 병렬화 기법은 H.264/AVC의 데이터 의존성을 지키면서 병렬화를 진행해야 한다. 이러한 문제점으로 해결하기 위하여 새로운H.264/AVC 디코더 병렬화 방법이 필요하다.

본 논문은 고해상도 동영상을 고속으로 처리하기 위한 H.264/AVC 디코더 병렬화 방법을 연구 한다. 우리가 제안하는 Frame Partition based Parallelization (FPP)방법은 병렬화 과정에서 발생하는 문제점을 해결하여 H.264/AVC 디코더 성능을 향상 하였다. 본 논문은 다음과 같은 순서로 이루어져 있다. 2장에서는 기존의 H.264/AVC 디코더 병렬화에 대한 관련 연구를 알아 본다. 그리고 3장에서는 본 논문에서 제안하는 FPP 방법에 대하여 설명하였다. 4장에서 실험 환경 및 실험 결과에

이 논문은 서울시 산학연 협력사업(10560)과 2009년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구(No.2009-0083601)입니다. 그리고 2010년도 두뇌한국21사업에 의하여 지원되었습니다

대하여 서술하였고 마지막으로 논문의 결론과 향후 계획으로 마무리 하였다.

2. 관련연구

최근 H.264/AVC 병렬화 하는 다양한 연구가 이루지고 있다.[2,3,4,5] 우리는 H.264/AVC디코더 병렬화 하는 방법인 태스크 레벨 병렬화 방법과 데이터 레벨 병렬화 방법으로 나누어 구체적으로 살펴 보았다.

가 태스크 레벨 병렬화 방법

H.264/AVC디코더는 Entropy Decoding (ED), Inverse Quantization/Inverse Transformation (IQ/IT), Intra prediction (IP), Motion Compensation (MC), Deblocking Filter (DF) 기능으로 이루어져 있다. 태스크 레벨 병렬화 방법은 H.264/AVC디코더의 각 기능별로 나누어 스레드에 할당하여 성능을 높이는 병렬화 방법이다. 일반적인 병렬화 방법인 파이프라인 병렬화 방법과 같다. 그림 1은 태스크 레벨 병렬화인 매크로 블록 단위로 파이프라인 병렬화 방법을 나타내고 있다. 그런데 H.264/AVC 디코딩 과정에서 각 기능별로 데이터를 처리하는 시간이 다르다.

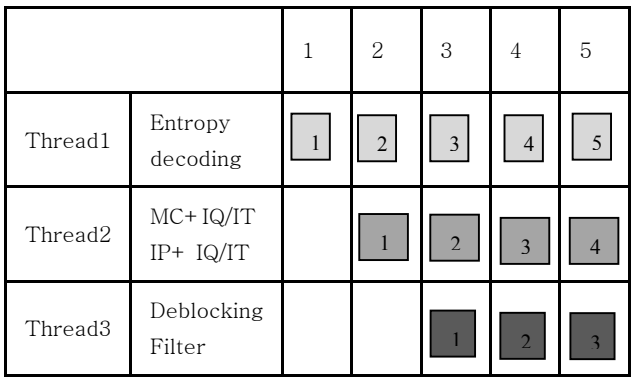

그림 1. 파이프 라인 병렬화 방법

H.264/AVC 디코딩 과정에서 각 기능별로 매크로 블록을 처리 시간이 달라 지는 이유는 매크로블록 타입에 따라서 처리 시간이 다르기 때문이다. 그러므로 매크로 블록의 타입에 따라 처리 속도의 차이가 발생하고 파이프 라인의 한 단계의 처리 시간은 가장 시간이 오래 걸리는 매크로 블록의 처리 시간이 된다. 그래서 전체적인 병렬화 처리 시간이 증가 하게 된다.

나 데이터 레벨 병렬화 방법

데이터 레벨 병렬화 방법은 H.264/AVC 데이터를 병렬화가 가능하게 나누어 처리하는 방법이다. H.264/AVC 데이터 단위에 따라 프레임 단위, 슬라이스 단위, 매크로 블록 단위 병렬화 방법으로 나누어 진다. 대표적으로 매크로블록 단위의 H.264/AVC 디코더를 병렬화에 대한 다양한 연구가 이루어 지고 있다. 매크로 블록 단위 데이터 레벨 병렬화 방법은 동시에 처리 할 수 있는 매크로 블록을 각 스레드에 할당하여 병렬화 한다. 그런데 H.264/AVC에서는 그림 2와 같이 매크로 블록 사이의 의존성이 발생 한다. 예를 들어 현재 매크로 블록이 IP을 처리 하기 위해서 1,2,3,4 매크로 블록이 IP가 처리 되어야 한다.

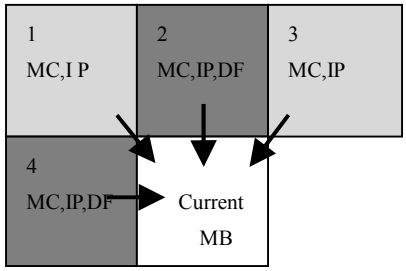

그림 2. 매크로 블록의 데이터 의존성

대표적인 데이터 레벨 병렬화 방법인 2Dwave 병렬화 방법이 있다. 그림 3은 H.264/AVC 데이터 의존성을 지키면서 데이터 레벨 병렬화 진행 하는 방법을 보여주고 있다. 그림 3에서 MB(4,0), MB(2,1), MB(0,2)는 데이터 의존성을 지키면서 5번째 시간에서 동시에 처리된다. 2Dwave 병렬화 방법은 그림 3에서 각 화살표에 스레드를 할당하여 화살표를 따라서 매크로 블록을 처리한다. 즉 2Dwave 병렬화 방법은 화살표 방향으로 수평적으로 스레드를 할당하여 병렬화 한다. 하지만 데이터 레벨 병렬화를 진행하기 위해서는 Entropy Decoding과정이 선행 되어야 한다. 이렇게 처리하는 이유는 Entropy Decoding은 데이터를 나누어 병렬화가 되지 않기 때문이다.

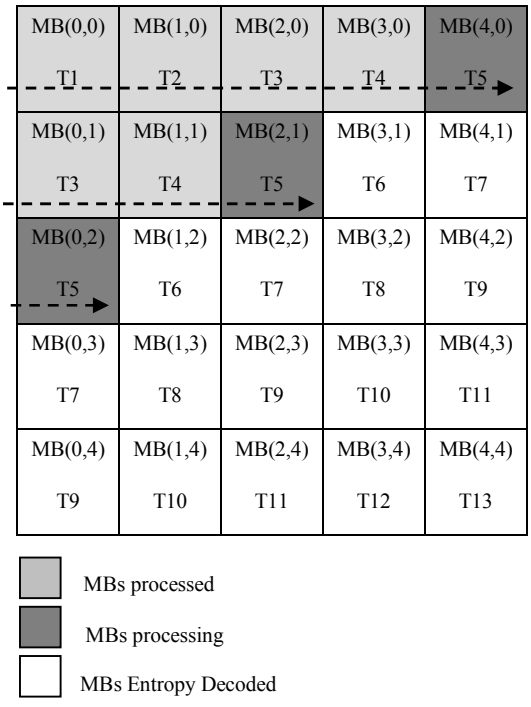

그림 3. 데이터 레벨 병렬화 방법

우리는 H.264/AVC 디코더 병렬화 방법을 살펴 보았다. 하지만 병렬화를 통한 성능 향상에 오버헤드가 있음을 알 수 있다. 우리는 이러한 병렬화 과정에서 발생하는 오버헤드를 해결 하기 위하여 새로운 H.264/AVC디코딩 병렬화 방법을 제안하다.

3. Frame Partition based Parallelization

본 논문은 고해상도 영상에서 H.264/AVC 디코더를 빠르게 처리 하기 위한 병렬화 방법 제안 한다. 우리가 제안하는 H.264/AVC 디코딩 병렬화 방법은 Frame Partition based Parallelization (FPP)방법이다. 프레임을 매크로 블록 묶음으로 나누어 병렬화 하고 독립적으로 스레드를 생성하여 처리하였다. 우리는 이러한 병렬화 방법을 사용하여 H.264/AVC 디코더 병렬화 과정에서 발생하는 스레드 사이의 동기화 오버헤드를 줄일 수 있다. FPP 방법은 태스크 레벨 병렬화 방법인 파이프 라인 방식 병렬화 방법과 유사하다. 파이프 라인 병렬화 방법은 H.264/AVC디코더를 기능별로 나누어 스레드에 할당하여 병렬화를 처리 한다. 우리는 ED, MC+IQ/IT IP+IQ/IT, DF 부분으로 나누었다. 나누어진 기능을 스레드에 할당하여 파이프 라인 병렬화를 진행한다. 기존의 파이프 라인 병렬화 방법은 매크로 블록 단위로 파이프 라인을 처리하고, 파이프 라인 단계마다 매크로 블록 단위로 스레드의 동기화를 확인 한다. 그러나 이러한 스레드의 동기화는 병렬화를 통한 성능의 걸림돌이 된다. 그리고 해상도가 높아지면 프레임당 처리하는 매크로블록의 개수가 증가하고, 따라서 스레드의 동기화 횟수도 늘어난다. 이러한 스레드의 동기화로 인한 문제를 해결 하기 위하여 프레임을 분할하여 매크로 블록 묶음으로 병렬화 하는 방법을 적용 하였다.

본 논문에서 제안하는 FPP방법은 프레임을 매크로 블록 묶음으로 나누어 파이프 라인 병렬화 처리 한다. 그림 4은 프레임 분할하여 매크로 블록 묶음으로 처리하는 과정을 보여준다. 그리고 효과적으로 스레드를 처리하기 위하여 파이프 라인 단계 마다 독립적으로 스레드를 생성하였다. 파이프 라인 단계에서 생성된 스레드는 매크로 블록 묶음을 처리 하고 종료 한다.

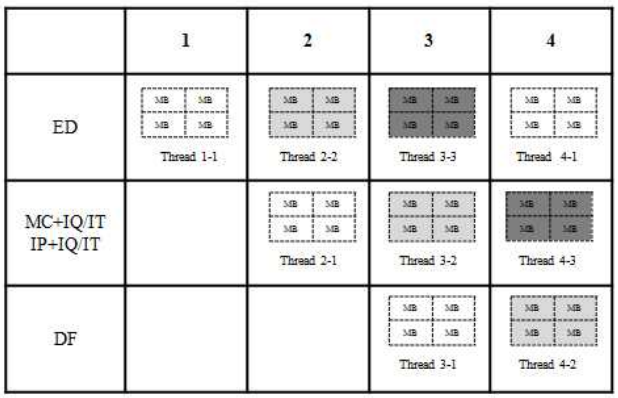

그림 4. 매크로블록 묶음 단위 병렬화

그림 4는 FPP 방법에서 스레드 처리를 보여 준다. 그림 4의 3번째 파이프 라인 단계에서는 thread3-1, thread3-2, thread3-3 를 생성하고, 생성된 스레드는 매크로블록 묶음을 처리하고 종료 한다. 동일하게 4번째 파이프 라인 단계에서 thread4-1, thread4-2, thread4-3을 생성하고, 생성된 스레드는 매크로블록 묶음을 처리하고 종료한다. 파이프 라인 단계에서 독립적으로 스레드가 생성하고 종료하기 때문에 파이프 라인 단계 사이의 스레드 동기화를 줄 일 수 있다.

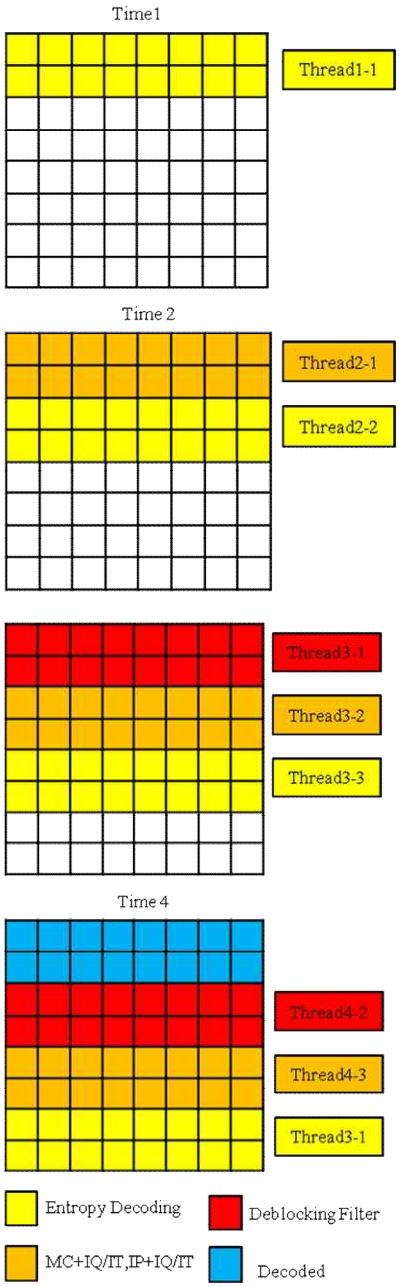

그림 5. Frame Partition based Parallelization

그림 5은 전체적인 FPP방법을 나타내고 있다. FPP방법은 그림 5의 time 1 에서 thread1-1은 ED를 수행하여 매크로 블록 생성한다. time 2에서 thread2-1은 생성된 매크로 블록을 이용하여 MC+IQ/IT IP+IQ/IT 처리한다. 그리고 thread2-2에서는 ED을 수행하여 매크로 블록을 생성한다. time 3에서 thread3-1은 생성된 매크로 블록을 사용하여 DF를 수행 한다. 그리고 threa3-2에서는 생성된 매크로 블록을 이용하여 MC+ IQ/IT IP+ IQ/IT 처리한다. 그리고 thread3-3에서는 ED을 수행하여 매크로 블록을 생성한다. time 3에서 thread 3-1이 끝나면 디코딩이 완료된 영상을 얻을 수 있다. 이러한 과정으로 FPP를 수행하여 프레임이 끝날 때까지 병렬화 한다.

4. 실험 및 결과

본 논문에서 오픈 소스 프로젝트로 개발 중인 FFmpeg H.264/AVC 디코더를 사용하여 병렬화 방법을 적용 하여 실험 하였다. 실험 결과의 사용한 영상은 JM-v16을 사용하여 인코딩 하였다. 인코딩 환경은 JM-v16 에서 제공하는 H.264/AVC baseline profile를 기반으로 하였다. 병렬화 방법은 OpenMP[6]를 사용하였다. 우리는 인텔 쿼드 코어 기반의 멀티 코어 시스템에서 병렬화 방법 적용 전 H.264/AVC 디코더와 병렬화 방법을 적용한 H.264/AVC 디코더를 비교 하였다. 표 1은 H.264/AVC 디코더 병렬화 방법을 적용 하였을 때 프레임을 처리하는 최소 시간을 나타낸다. 표 2은 H.264/AVC 디코더 병렬화 방법을 적용 하였을 때 성능 향상율을 나타낸다.

FHD 1920X1088	Basic (μs)	Pipeline (μs)	2Dwave (μs)	FPP (μs)
rush_hour	14746	11873	9998	6993
blue_sky	13694	11018	9558	6735
pedestrain_area	13498	12126	9604	8431
sunflower	11034	9481	8507	6718

표 1. H.264/AVC 디코더 병렬화 방법 비교(μs)

FHD 1920X1088	Pipeline (%)	2Dwave (%)	FPP (%)
rush_hour	19%	32%	53%
blue_sky	20%	30%	51%
pedestrain_area	10%	29%	38%
sunflower	14%	23%	39%

표 2. H.264/AVC 디코더 병렬화 방법 비교(%)

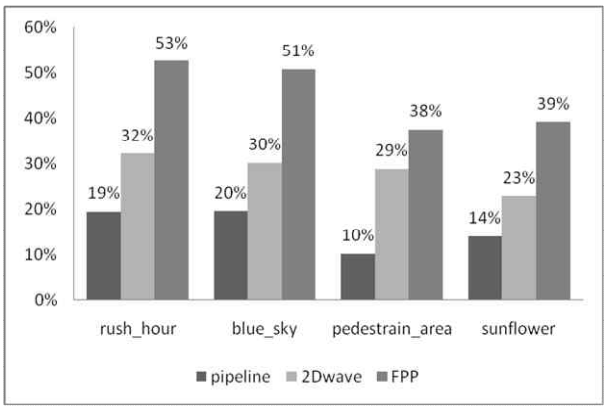

그림 6. H.264/AVC 디코더 병렬화 성능 비교

표 1,2 에서 적용한 병렬화 방법은 다음과 같다. 태스크 레벨 병렬화 방법인 파이프라인 병렬화 방법을 실험 하였다. 그리고 데이터 레벨 병렬화 방법인 2Dwave 병렬화 방법을 실험 하였다 마지막으로 본 논문에서 제안한 FPP 방법을 실험하였다. 표 2에서 파이프 라인 병렬화 방법은 12~32%, 2Dwave 병렬화 방법은 23~32% 성능 향상하였다. 우리가 제안한 FPP 방법은 38~53% 성능 향상 하였다. 그림 6는 표 2의 결과를 그래프로 나타내었다. 본 논문에서 제안하는 FPP 방법이 다른 병렬화 방법에 비교하여 효과적임을 알 수 있다.

5. 결론

본 논문에서 고해상도에서 H.264/AVC 디코더를 빠르게 처리하기 위하여 새로운 H.264/AVC 디코더 병렬화 방법인 FPP 방법을 제안 하였다. FPP 방법은 기존의 H.264/AVC 디코더를 병렬화 과정에서 발생하는 문제점을 해결하여 효과적으로 H.264/AVC 디코더를 병렬화 하였다. 우리는 FPP 방법을 인텔 쿼드 코어 시스템에서 실험 하였고 다른 병렬화 방법과 성능을 비교하였다. 향후 FPP 방법을 확장하여 다양한 멀티 코어 시스템에서 적용하여 하려고 한다.

참고문헌

[1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, Senior Member, " Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003

[2] E. van der Tol, E. Jaspers, and R.Gelderblom, " Mapping of H.264 decoding on a multiprocessor architecture," Image and Video Communications and Processing, pp.707-718, May, 2003.

[3] J. Chong, N. R. Satish, B. Catanzaro, K. Ravindran, and K.Keutzer,"Effcient parallelization of h.264 decoding with macro block level scheduling," in 2007 IEEE International Conference on Multimedia and Expo, July 2007.

[4] Kosuke Nishihara, Atsushi Hatabu, Tatsuji Moriyoshi, "Parallelization of H.264 video decoder for embedded multicore processor," In Proceedings of ICME'2008. pp.329~332.

[5] A. Azevedo, C. Meenderinck, B. Juurlink, A. Terechko, J. Hoogerbrugge, M. Alvarez, and A. Rammirez, "Parallel H.264 Decoding on an Embedded Multicore Processor," in Proceedings of the 4th International Conference on High Performance and Embedded Architectures and Compilers -HIPEAC, Jan 2009.

[6] Chunhua Liao, Zhenying Liu, Lei Huang, and Barbara Chapman. "Evaluating OpenMP on Chip MultiThreading Platforms," In First international workshop on OpenMP, Eugene, Oregon USA, June 2005.