Journal of Korea Game Society 2014 Dec; 14(6): 59-68
http://dx.doi.org/10.7583/JKGS.2014.14.6.59

게임 프로그래밍

DPCM-GR 방식을 이용한 CUDA 기반 초고해상도 게임 영상 무손실 비동기 압축

김영식
한국산업기술대학교 게임공학과
kys@kpu.ac.kr

CUDA based Lossless Asynchronous Compression of Ultra High Definition Game Scenes using DPCM-GR

Youngsik Kim
Dept. of Game & Multimedia Engineering, Korea Polytechnic University

요 약

초고해상도 UHD(4096x2160) 게임 영상의 메모리 대역폭 요구량은 기하급수적으로 늘어난다. 본 논문에서는 화질 저하 없이 메모리 대역폭 문제를 해결하기 위하여 CUDA 환경에서 비트 병렬 파이프라인을 지원하는 논문 [4]의 DDPCM-GR 압축 알고리즘을 변형한 DPCM-GR 방식을 적용한 무손실 압축을 구현하였다. CUDA 공유메모리 사용을 통한 효율성을 증대하였으며, paged-locked 호스트 메모리 비동기 전송을 통한 커널과 데이터 전송 중첩의 다양한 구성을 구현하였다. 실험을 통하여 CPU 방식에 비하여 최대 31.3배 속도 향상을 이루었으며, 비동기 전송 구성의 변화를 통하여 최대 30.3% 수행 시간이 감소하였다.

ABSTRACT

Memory bandwidth requirements of UHD (Ultra High Definition 4096x2160) game scenes have been much more increasing. This paper presents a lossless DPCM-GR based compression algorithm using CUDA for solving the memory bandwidth problem without sacrificing image quality, which is modified from DDPCM-GR [4] to support bit parallel pipelining. The memory bandwidth efficiency increases because of using the shared memory of CUDA. Various asynchronous transfer configurations which can overlap the kernel execution and data transfer between host and CUDA are implemented with the page-locked host memory. Experimental results show that the maximum 31.3 speedup is obtained according to CPU time. The maximum 30.3% decreases in the computation time among various configurations.

Keywords : CUDA, Lossless Compression(무손실 압축), Asynchronous Transfer(비동기 전송), DPCM-GR (Differential Pulse Code Modulation - Golomb Rice)

Received: Nov. 13, 2014 Accepted: Dec. 11, 2014
Corresponding Author: Youngsik Kim(Korea Polytechnic Univ.)
E-mail: kys@kpu.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

최신 게임에서 고해상도 모니터, 디지털 TV, 스마트폰 등의 다양한 기기의 고화질 요구에 따라서 디스플레이 해상도는 HD(1920x1080)를 넘어서 초고해상도 UHD(4096x2160, 7680x4320)로 지속적으로 증가하는 추세이다. 특히, 해당 플랫폼에서 초고해상도 게임 영상 해상도를 맞추기 위해서 메모리 대역폭의 요구량이 기하급수적으로 늘어난다. 이러한 초고해상도 디스플레이를 위해서 메모리 대역폭을 줄이기 위한 게임 영상 압축 연구가 필요하게 되었다. 초고해상도 영상의 압축은 화질 저하를 막기 위해서 일반적으로 무손실(lossless) 압축 알고리즘이 적합하다.

무손실 압축 알고리즘은 화질 저하는 없는 대신에 가변 길이(variable length) 압축 데이터를 가지므로 압축률이 고정이 아니다. 손실 압축(Lossy compression) 알고리즘은 고정 길이 압축 데이터를 가지며 압축률이 고정이라서 메모리 접근 패턴이 단순하고 비트 병렬 처리하기 쉽니다. 그러나 무손실 압축을 적용하기 위해서는 시각적으로 무손실 해야 하는 제약이 있다. 손실 압축을 적용하여 가변 길이 데이터를 사용하면 메모리 인터페이스가 해결되어야 하고 비트 병렬 처리가 가능해야 한다. 이와 같은 무손실 압축 하드웨어 구조에 대한 연구가 많이 진행되었다[1,2,3,4,5,6].

GPU(Graphics Processing Unit) 내부에 존재하는 대규모 프로세서들을 병렬적으로 수행가능하게 하는 NVIDIA CUDA(Computer Unified Device Architecture)[7] 컴퓨팅 환경을 이용한 많은 연구가 진행되었다[8,9,10,11].

논문 [8]에서는 합산영역테이블 계산을 위해서 입력 데이터를 정방의 서브 이미지로 분할하고 매개 데이터를 이들 간에 파급시켜서 전역 메모리 접근 량을 거의 반으로 줄임으로써 주어진 메모리 대역폭을 효율적으로 사용하였다. 논문 [9]에서는 메모리 정렬이 되지 않은 웹캠 영상의 색상 형식 변환을 CUDA를 이용하여 가속하는 최적화 기법을 제안하였다. 논문 [10]에서는 CUDA를 이용하여 GPU에서 최대-최소 8진트리의 생성을 가속화하는 방법을 제안하였다. 논문 [11]에서는 CUDA 프로그래밍 기술을 활용하여 DXT(DirectX Texture) 압축의 저지연(low-latency) 실시간성을 개선하기 위한 기법들을 제안하였다. 그러나 DXT 압축은 손실 압축 기법이므로 화질 저하를 막을 수 없다.

본 논문에서는 초고해상도 UHD(4096x2160) 게임 영상의 메모리 대역폭 문제를 해결하기 위하여 CUDA 환경에서 비트 병렬 파이프라인을 지원하는 논문 [4]의 DDPCM-GR 압축 알고리즘을 변형한 DPCM-GR 방식을 적용한 무손실 압축을 구현하였다. 이를 위하여 공유메모리(shared memory) 사용을 통한 메모리 효율성을 고려하였다. 또한 성능 최적화를 위하여 paged-locked 호스트 메모리 비동기 전송을 통한 커널과 메모리 전송 중첩의 다양한 구성을 구현하고 실험을 통하여 최적의 구성을 성능 검증하였다.

2. 무손실 이미지 압축 관련연구

HMD-ExpG(Hierarchical Minimum Difference + ExpGolomb)[1] 알고리즘은 예측(prediction)을 위해서 계층적으로 2x2 블록, 4x4 블록, 8x8 블록 각각 최소값을 구하고 그 최소값과 해당하는 블록의 다른 픽셀 값과의 차이를 구한 후 엔트로피 코딩(entropy coding)은 지수(exponential) Golomb 코딩을 한다. 압축률을 높이고 랜덤 액세스를 위해서 64비트 양자화 주소 테이블(Quantized Address Table)을 갖는다. 이 알고리즘의 장점은 랜덤 액세스를 고려하였으며 압축률이(compression ratio) 2.1:1로 비교적 높다. 그러나

단점으로 테이블 오버헤드 (6.1KB/frame)를 가지며 지수 Golomb 코딩의 하드웨어 레이턴시 (ExtG = 12cycle)가 크다. 또한 계층적으로 최소값을 구하고 그 차이를 계산하는 방식은 하드웨어 구현 시에 고속 파이프라인 및 비트 병렬적으로 동작하기 어려우며 버스트 접근(burst access)을 구현하기 어렵다.

HMD-NBS[2] 알고리즘은 HMD-ExpG 알고리즘의 디코딩 레이턴시를 줄이기 위해서 예측은 기존과 같이 HMD(hierarchical minimum difference) 방식을 사용하고 엔트로피 코딩을 새로운 방식인 NBS (Nonzero Bit Selection)를 사용한다. 차이 값을 갖는 블록을 5개의 그룹으로 나누고 각각의 그룹 당 최소값과의 차이 값을 표현해야 하는 최소 비트 수를 구해서 각각의 차이 값 그룹 의 시작 위치를 미리 알 수 있다. 이 방식의 장점은 디코딩 사이클을 3 사이클로 줄일 수 있다. 그러나 여전히 테이블 오버헤드를 가지며 HMD-ExpG를 단점과 마찬가지로 고속 파이프라인과 비트 병렬적으로 동작하기 어렵다.

HACP-SBT[3] 알고리즘은 예측을 위해서 HACP(hierarchical average copy prediction)이라고 해서 수평적 평균 예측(horizontal average prediction), 수직적 평균 예측(vertical average prediction), 수평적 직접 예측(horizontal direct prediction), 수직적 직접 예측(vertical direct prediction), 4개 픽셀 평균 예측(four pixel average prediction) 등의 5가지 방식을 사용한다. 그리고 엔트로피 코딩을 위해서 HMD-NBS 알고리즘과 유사하게 각각의 그룹 별로 표현할 수 있는 최소 비트 수를 정해서 표현하는 SBT (significant bit truncation) 방식을 사용한다. 이 알고리즘의 장점은 압축률이 2.56:1 높아서 대역폭 감소가 61%에 이른다. 그러나 랜덤 액세스가 안되고 인코딩 사이클 오버헤드가 있으며 예측 방식이 복잡해서 파이프라인이나 비트 병렬 하드웨어 구현이 어렵다.

DDPCM-GR[4] 알고리즘은 예측으로 ATI 회사에서 고안한 알고리즘 DDPCM[5,6] (differential differential pulse code modulation)을 사용하고 엔트로피 코딩은 Golomb-Rice(GR) 코딩 스킴을 사용한다. 이 압축 알고리즘의 장점은 간단한 prediction 방법을 사용하여 비트 병렬 완전 파이프라인 하드웨어 구조를 고안한 것이다. 이를 통하여 2 사이클 인코딩/디코딩 파이프라인 구조를 제공한다. 압축률은 8x8 블록에 대해서 1.64:1 을 제공한다.

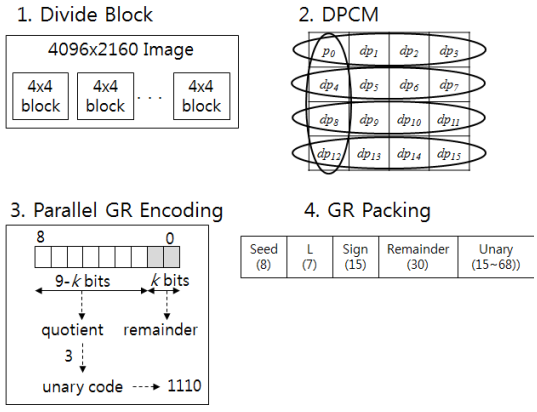
[Fig. 1] ATI DDPCM [5,6]

ATI에서 고안한 DDPCM은 3D 그래픽 데이터 중에서 Z 데이터 압축을 위한 것이다. 기본적인 DDPCM을 사용하는 것은 깊이 값인 Z 데이터는 선형 보간 (linearly interpolated) 되었다고 가정한다. DDPCM 이후에는 깊이 값을 -1, 0, 1, escape value 로 구분한다. 구분된 값을 엔트로피 코딩을 수행해서 0은 00, 1은 01, -1은 10, escape value는 11로 코딩한다. 그러나 DDPCM은 원래 Z데이터 압축을 위해서 고안한 방식이므로 게임 장면 컬러 픽셀들은 DDPCM을 사용하지 않고 좀 더 간단한 DPCM (differential pulse code modulation)을 적용해도 압축 효율이 떨어지지 않는다.

본 논문에서는 CUDA 환경에서 비트 병렬 파이프라인을 지원하기위해서 논문 [4]의 DDPCM-GR 압축 알고리즘을 변형한 DPCM-GR 방식을 적용한 무손실 압축을 구현하였다. 또한 CUDA 사용의 최적화를 위해서 공유메모리 사용을 통한 메모리 사용을 최적화하였고, 성능 극대화를 위하여 paged-locked 호스트 메모리 비동기 전송을 통한 커널과 메모리 전송 중첩의 다양한 구성을 구현하고 실험을 통하여 성능을 검증하였다.

3. DPCM-GR 무손실 압축

이번 절에서는 논문 [4]의 DDPCM-GR 압축 알고리즘을 변형한 DPCM-GR 방식을 적용하여 비트 병렬 파이프라인을 지원하는 무손실 압축 알고리즘을 설명한다.



[Fig. 2] DPCM-GR lossless compression algorithm.

초고해상도 게임 영상을 위한 전체 무손실 압축 알고리즘은 [Fig. 2]와 같이 크게 1. 원본 이미지 데이터를 4 x 4 데이터 블록으로 나누는 부분, 2. 데이터 블록에 대한 DPCM 계산, 3. 병렬 GR 인코딩(Golomb-Rice encoding), 4. GR 포장(GR Packing)으로 구성된다. 각 데이터 블록에 대하여 DPCM 계산을 위해서 하나의 시드(Seed) 데이터와 15개의 차분 데이터가 도출된다. 15개의 차분 데이터에 대하여 GR 알고리즘을 비트 병렬적으로 적용하여 전체적으로 무손실 압축 데이터를 도출한다.

시드 데이터는 압축하지 않은 원본 데이터이다. GR 인코딩을 위해서 DPCM 차분 데이터를 2^k 으로 나눈다. 이 논문에서는 GR 매개변수 k를 2로 가정한다. 그러면 나머지(remainder) 부분은 고정된 비트 수로써 15x2 =30 비트가 된다. 그리고 15개의 몫(quotient)들은 unary code 로 표현한다. 예를 들어서 몫이 3이면 1110 (0은 구분비트)로 표현한다. 구분 비트의 위치(t_i)들을 동시에 계산하기 위해서 다음 (eq. 1)과 같이 비트 병렬적으로 계산한다.

$$t_i = \sum_{j=1}^{i} q_j + i - 1, (1 \le i \le 15) \qquad \text{(eq. 1)}$$

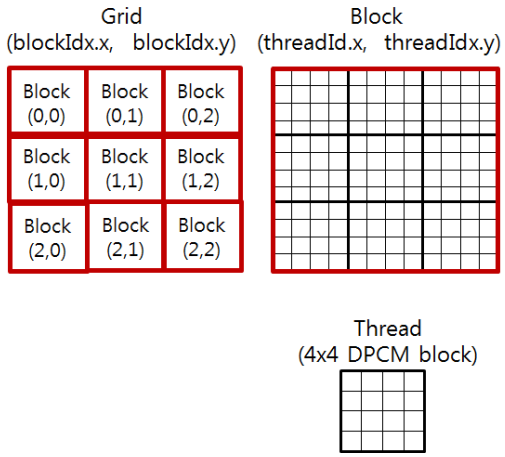
예를 들어서 q_1=1, q_2=3, q_3=5 이면 unary code는 101110111110 이고 구분 비트들은 t_1=q_1=1, t_2=q_1+q_2+1=5, t_3=q_1+q_2+q_3+2=11 이다.

GR 포장에서는 압축 비트 길이가 원본 비트 길이보다 길면 그냥 원본 비트로 포장하고 아니면 [Fig. 2]에서처럼 시드 데이터 (8비트), 길이 (L 7비트), 나머지 (30비트), unary code (15~68비트)로 포장한다.

4. CUDA를 이용한 비동기 압축 구현

이번 절에서는 무손실 DPCM-GR 비동기 압축 알고리즘을 CUDA 환경에 최적화하도록 구현하였다. 우선 UHD(4096x2160) 원본 이미지를 호스트에서 CUDA로 전송해야 한다. 첫 번째, [Fig. 3]과 같이 CUDA 프로세서들을 UHD(4096x2160) 이미지의 4x4 블록으로 나눠서 DPCM-GR 계산을 수

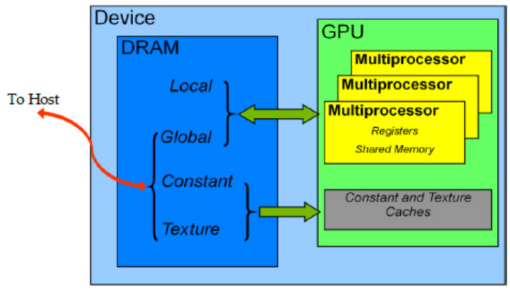
행하는 thread, thread (threadIdx.x, threadIdx.y) 들이 모여서 block, block (blockIdx.x, blockIdx.y)들이 모여서 grid를 이룬다.



[Fig. 3] CUDA grid, block, and thread for DPCM-GR compression.

두 번째, 4x4 블록으로 나눠서 DPCM-GR 계산을 수행하는 thread들은 동시에 DPCM을 수행한다. 세 번째, 각 thread 들은 DPCM을 계산한 4x4 블록에 대해서 GR 인코딩을 수행한다 ([Fig. 2] 3. Parallel GR Encoding 참조). 마지막으로 각 thread 들은 GR 포장을 한다. 이렇게 모든 UHD(4096x2160) 이미지의 4x4 블록마다 생성된 GR 포장 데이터를 호스트로 전송하면 된다.

CUDA 내부에는 [Fig. 4]와 같이 다양한 메모리가 존재한다. 가장 크고 느린 메모리가 Global 메모리 (디바이스 메모리) 이며 이를 통해서 호스트와 CUDA 사이에 데이터를 주고받을 수 있다. 그리고 같은 Global 메모리 내에 약간의 영역을 할당 받아 GPU가 직접 관장하는 Local 메모리가 있다. 그리고 Global 메모리 일부에 상수와 텍스쳐 메모리가 있다. 제일 빠른 메모리는 공유 메모리와 레지스터이다. DRAM으로 구성되는 Global 메모리와 공유 메모리 사이의 접근 속도 차이는 약 100배 정도 차이가 난다고 한다.



[Fig. 4] Memory Spaces on a CUDA Device [7].

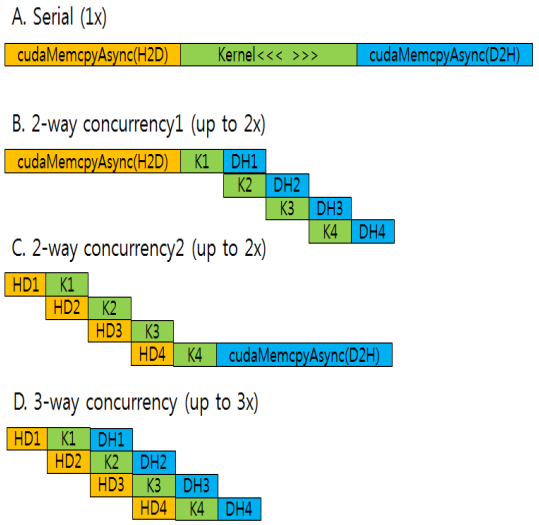
본 논문에서는 공유 메모리의 빠른 메모리 접근 속도를 활용하기 위해서 CUDA block 내의 thread 가 4x4 이미지 블록을 공유 메모리로 읽어와서 DPCM 계산과 병렬 DPCM-GR 인코딩 등을 처리하도록 한다. CUDA Compute Capability 1.2 기준으로 본 논문에서 구현한 DPCM-GR CUDA 커널을 "-ptxas-options=-v" 옵션으로 컴파일하여 CUDA block 당 공유 메모리와 thread 당 레지스터 사용량을 [Table 1]과 같이 구하였다. thread 당 레지스터 개수 14개와 block 당 공유 메모리 124바이트는 적당한 크기이다.

[Table 1] Shared memory and register usage

Memory Type	Usage
Shared Memory Per Block	124 bytes
Register Per Thread	14 registers

성능을 극대화하기 위해서 커널 실행과 호스트와 CUDA 디바이스 사이의 데이터 전송은 비동기적 함수 사용을 통하여 중첩할 수 있다. 중첩하는 데이터 전송을 위해서는 page-locked(pinned) 호스트 메모리를 사용해야 한다. page-locked 호스트 메모리는 일반적인 호스트 메모리보다 전송 속도가 빠르다. 또한 CUDA 커널 함수의 비동기적인 실행과 데이터 전송을 CUDA 스트림(stream)을 사용하여 파이프라인으로 동작하게 할 수 있다.

본 논문에서는 성능을 극대화하기 위해서 CUDA DPCM-GR 커널 실행과 호스트와 CUDA 디바이스 사이의 원본 이미지 데이터 전송 및 압축 결과 데이터 전송을 비동기적 함수 사용을 통하여 중첩하였다. [Fig. 5]와 같이 CUDA 커널과 데이터 전송에 관한 비동기 실행에 관한 구성을 4가지로 하여 A (Serial: 모든 데이터 전송과 커널 수행은 시리얼 실행), B (2-way concurrency 1: 호스트에서 디바이스 데이터 원본 이미지 전송은 시리얼, 커널 실행과 디바이스에서 호스트 결과 데이터 전송은 파이프라인 비동기 실행), C (2-way concurrency 2: 호스트에서 디바이스 데이터 원본 이미지 전송과 커널 실행은 파이프라인 비동기 실행, 디바이스에서 호스트 결과 데이터 전송은 시리얼), D (3-way concurrency: 호스트에서 디바이스 데이터 원본 이미지 전송, 커널 실행, 디바이스에서 호스트 결과 데이터 전송 모두 파이프라인 비동기 실행) 로 구성하고 실험을 통하여 성능을 검증하였다.



[Fig. 5] Four concurrency configurations according to asynchronous transfers

5. 실험 및 평가

본 논문에서는 4가지 CUDA 커널과 데이터 전송에 관한 비동기 실행 구성에 대한 성능 평가를 위하여 실험 환경을 Intel(R) Core(TM) i5 CPU M520 @ 2.40GHz 2.40GHz 8GB RAM, GPU는 NVIDIA GeForce 330M (Compute Capability 1.2, 48 Cores, 1.1GHz, 256MB Global Memory) 과 같이 구성하였다. ([Table 2] 참조)

비동기 실행을 위하여 필요한 page-locked 호스트 메모리 환경에서 호스트-메모리 대역폭과 메모리-호스트 대역폭은 각각 12170.3MB/s와 12228.3 MB/s로 우수하다.

본 논문에서는 초고해상도 UHD(4096x2160) 게임 영상 벤치마크 이미지를 [Fig. 6]과 같이 6가지 준비하였다. 6가지 벤치마크 이미지에 대한 베이스라인 CPU 기반 DPCM-GR 압축을 수행하고 [Table 3]과 같이 성능을 측정하였다. 평균 CPU 수행시간은 1686.76ms 이고 압축 비율은 0.68이다.

[Table 2] GeForce GT330M Spec

GeForce GT330M	Description
Driver version	6.5
Computer Capability	1.2
Total Global Memory	256MB
Cores	48 Cores = 6 Multiprocessors x 8 CUDA cores
GPU Clock rate	1.1 GHz
Shared Memory/block	16KB
Warp size	32
Max threads per block	512
H2D Pinned memory bandwidth	12170.3 MB/s
D2H Pinned memory bandwidth	12228.3 MB/s


Image 1 [12]


Image 2 [13]


Image 3 [14]


Image 4 [15]


Image 5 [16]


Image 6 [17]

[Fig. 6] Benchmark UHD (4096x2160 resolution) game images

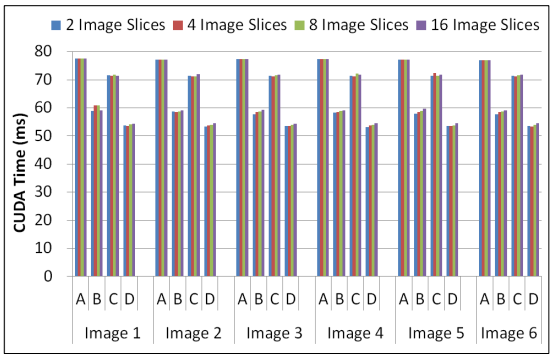
[Table 3] CPU time and compression ratio for benchmark images

	CPU Time (ms)	Compression Ratio
Image 1	1699.39	0.621
Image 2	1702.93	0.743
Image 3	1704.84	0.652
Image 4	1697.37	0.847
Image 5	1666.89	0.615
Image 6	1649.12	0.601

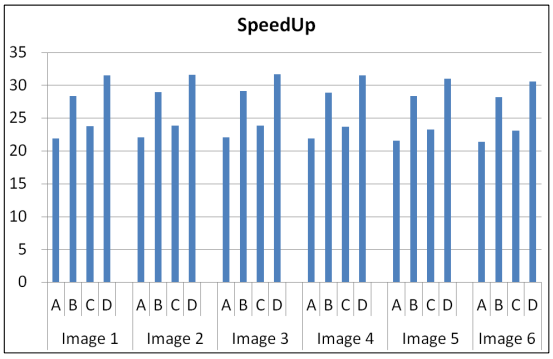
[Fig. 7]은 6가지 벤치마크 이미지에 대해서 4가지 비동기 실행 구성(A,B,C,D)을 적용한 CUDA 실행 시간을 측정한 것이다. 비동기 파이프라인 동작을 위하여 원본 이미지를 각각 2, 4, 8, 16 슬라이스로 잘라서 적용하였다. 벤치마크 이미지 6가지와 이미지 슬라이스 4가지는 크게 유의미한 차이를 보이지 않았다. 벤치마크 이미지와 이미지 슬라이스 크기 따른 CUDA 실행시간 차이가 유의미하지 않은 이유는 데이터 전송과 커널 실행사이의 준비 오버헤드 시간이 더 크기 때문이다.

비동기 파이프라인 구성 4가지에 대해서는 확실한 성능 차이를 보이고 있다. D(3-way concurrency: 호스트에서 디바이스 데이터 원본 이미지 전송, 커널 이미지 전송, 커널 실행, 디바이스에서 호스트 결과 데이터 전송 모두 파이프라인 비동기 실행) 구성이 가장 성능이 우수하였으며, A (Serial: 모든 데이터 전송과 커널 수행은 시리얼 실행) 구성에 비하여 최대 CUDA 실행 시간을 30.3% 향상 시켰다. B (2-way concurrency 1: 호스트에서 디바이스 데이터 원본 이미지 전송은 시리얼, 커널 실행과 디바이스에서 호스트 결과 데이터 전송은 파이프라인 비동기 실행) 구성과 C (2-way concurrency 2 : 호스트에서 디바이스 데이터 원본 이미지 전송과 커널 실행은 파이프라인 비동기 실행, 디바이스에서 호스트 결과 데이터 전송은 시리얼) 구성도 각각 A구성에 비하여 23.8%와 7.4% 성능 향상을 얻었다. C 구성의 성능 향상이 적은 이유는 압축 결과 데이터 크기가 원본 이미지 데이터 크기에 비하여 작고 불규칙하기 때문이다.

[Fig. 8]은 CPU 실행 시간 대비 CUDA 실행 시간의 속도 향상 (SpeedUp)을 측정한 것이다. A, B, C, D 구성은 각각 CPU 실행 시간에 비하여 21.8배, 28.7배, 23.6배, 31.3배 속도 향상으로 D 구성이 가장 우수하였다.



[Fig. 7] CUDA time according to changing benchmark images, the number of image slices, and concurrency configurations.



[Fig. 8] SpeedUp (CPU time / CUDA time) according to benchmark images, the number of image slices, and concurrency configurations.

논문 [11]에서 제시한 DXT방식의 CUDA 기반 초고해상도 영상 압축 및 전송은 메모리 저지연 실시간성 측면에서 우수한 성능을 보이지만 DXT 압축은 손실 압축 기법이므로 화질 저하를 막을 수 없다. 본 논문에서는 화질 저하 없는 DPCM - GR 방식을 이용한 CUDA 기반 무손실 초고해상도 게임 영상 압축 기법을 제시하였다.

6. 결 론

초고해상도 디스플레이를 위해서 메모리 대역폭을 줄이기 위한 게임 영상 압축 연구가 필요하게 되었다. 게임 영상의 압축은 화질 저하를 막기 위해서 무손실 압축 알고리즘이 적합하다.

본 논문에서는 초고해상도 UHD(4096x2160) 게임 영상의 메모리 대역폭 문제를 화질 저하 없이 해결하기 위하여 CUDA 환경에서 커널과 메모리 전송 중첩을 지원하는 비트 병렬 파이프라인 DPCM-GR 방식을 적용한 무손실 압축을 구현하였다. CUDA 공유메모리 사용을 통한 메모리 효율성을 증대시켰으며, paged-locked 호스트 메모리 비동기 전송을 통한 커널과 메모리 전송 중첩의 다양한 구성을 구현하고 실험을 통하여 최적의 구성을 성능 검증하였다.

REFERENCES

[1] S. Lee, M. Chung, S. Park, and C. Kyung, “Lossless frame memory recompression for video codec preserving random accessibility of coding unit”, IEEE Trans. Consumer Electro., Vol. 55, No. 4, pp.2105-2113, Nov. 2009.

[2] S. Lee, N. Eum, M. Chung, and C. Kyung “Low Latency Variable Length Coding Scheme For Frame Memory Recompression”, 2010 IEEE International Conference on Multimedia and Expo (ICME), pp.232-237, IEEE, 2010.

[3] J. Kim and C. M. Kyung, “A lossless embedded compression using significant bit truncation for HD video coding”, IEEE Transaction on Circuit and System for Video Technology, Vol. 20, No. 7, pp. 848-860, June 2010.

[4] Hong-Sik Kim, Joohong Lee , Hyunjin Kim, Sungho Kang, and Woo Chan Park, “A Lossless Color Image Compression Architecture Using a Parallel Golomb-Rice Hardware Codec”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 21, No. 11, pp.1581-1587, Nov. 2011.

[5] Steve Morein, “ATI Radeon HyperZ

Technology", ACM/Eurographics Symposium on Graphics Hardware, Aug 22, 2000.
[6] DEROO J., MOREIN S., FAVERA B., WRIGHT M., : Method and Apparatus for Compressing Parameter Values for Pixels in a Display Frame. In US Patent 6,476,811 (2002).
[7] NVIDIA CUDA, http://developer.nvidia.com/object/cuda.html.
[8] Sang-Won Ha, Moon-Hee Choi, Tae-Joon Jun, Jin-Woo Kim, Hye-Ran Byun, and Tack-Don Han, "Bandwidth Efficient Summed Area Table Generation for CUDA", Journal of Korea Game Society, Vol. 12, No. 5, pp.67-78, 2012.
[9] Jin-Woo Kim, Yunhye Jung, Jinhong Park, Yong-Jin Park, and Tack-Don Han, "Optimization of Color Format Conversion of WebCam Images Using the CUDA", Journal of Korea Game Society, Vol. 11, No. 1, pp.147-157, 2011.
[10] Jong-Hyeon Lim and Byeong-Seok Shin, "Min-Max Octree Generation Using CUDA", Journal of Korea Game Society, Vol. 9, No. 6, pp.191-196, 2009.
[11] Namgon Lucas Kim and Jong-Won Kim "GPU-based Low-latency DXT Compression and Transport for 4K Ultra-high-definition Media Sharing", KIISE Transactions on Computing Practices, Vol. 18, No. 8, pp.573-581, 2012.
[12] Image 1 Asura's Wrath from http://www.giantbomb.com/
[13] Image 2 Tom Clancy's the Division from http://www.dlh.net/
[14] Image 3 Tom Clancy's the Division from http://www.dlh.net/
[15] Image 4 Devil May Cry 5 from http://-www.gamehdwall.com/
[16] Image 5 ForgeMaster from http://www.over3000.net/
[17] Image 6 China Lake Grenade Launcher Pop Gun from http://www.polycount.com/

김 영 식 (Youngsik Kim)

1993년 연세대학교 컴퓨터과학과 학사
1995년 연세대학교 컴퓨터과학과 석사
1999년 연세대학교 컴퓨터과학과 박사
1999년-2005년 삼성전자 System LSI 책임연구원
2013년 University of Pittsburgh 방문교수
2005년-현재 한국산업기술대학교 부교수

관심분야 : 게임기구조, 컴퓨터구조, 3차원 그래픽가속기, 임베디드 시스템 등