GPGPU를 활용한 PNG 압축 필터링 가속화

강준범[1], 박능수[1]
[1]건국대학교 컴퓨터공학과
aopko@konkuk.ac.kr, neungsoo@konkuk.ac.kr

Acceleration of PNG Compression Filtering Using GPGPU

Junbeom Kang[1], Neungsoo Park[1]
[1]Dept. of Computer Science and Engineering, Konkuk University

요 약

기존 병렬 프로그래밍을 활용한 이미지 압축 관련 알고리즘 연구는 Joint Photographic Experts Group(JPEG) 관련 알고리즘이 주를 이루고 있었다. 따라서 Portable Network Graphics(PNG) 관련 연구가 필요하다고 판단하여, PNG 압축 알고리즘에 병렬프로그래밍과 GPGPU 개념을 도입해 더욱 빠른 압축 커널 설계 필요성을 생각했다. 이번 연구는 이미지 필터링, Lempel-Ziv 77 알고리즘, 허프만 코딩에 걸친 세 단계의 절차 중 이미지와 GPGPU의 특징 중 픽셀 단위와 다중 코어로 구성된 구조적 유사성을 고려하여 우수한 병렬화를 기대할 수 있는 이미지 필터링을 병렬 알고리즘으로 설계했다. 병렬 필터링 알고리즘은 CUDA 프로그래밍 모델의 블록 구조를 사용해 구현했으며 직렬 필터링 대비 약 190배의 가속화를 보였다. 추후 연구로는 이번 연구에서 다루지 않은 나머지 두 단계를 고려하여 포괄적인 병렬 PNG 압축 알고리즘을 설계하고자 한다.

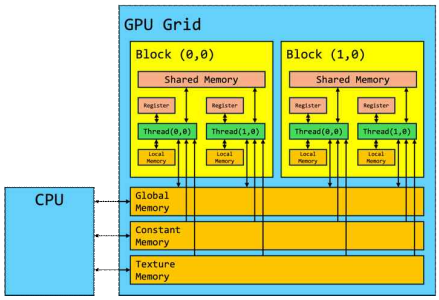
1. 서론

최근 다양한 연구 분야에서 병렬프로그래밍을 이용한 알고리즘 가속화에 중점을 두고 있다. 그 중, 이미지 압축과 관련한 연구로는 과거부터 비교적 현재까지 대부분 Joint Photographic Experts Group(JPEG) 표준에 대한 압축 알고리즘 병렬처리를 통한 가속화에 집중되어 있음을 확인할 수 있었다[1-3]. 또한 General Purpose Graphics Processing Unit(GPGPU)를 잘 활용할 수 있도록 설계된 프로그래밍 모델 중 하나인 NVIDIA사의 Compute Unified Device Architecture(CUDA)에서도 nvJPEG과 같은 GPU 기반 JPEG 표준 디코더, 인코더, 트랜스코더 가속화 라이브러리만 제공하고 있다. 따라서 본 논문에서는 투명 채널을 활용할 수 있으며 다양한 그래픽 디자인 환경에서 사용할 수 있는 Portable Network Graphics(PNG) 형식에 접근하여 PNG 압축 알고리즘에 대한 병렬처리를 진행하고 기존 직렬 구조에 비해 빠른 압축 커널을 제안하고자 한다. 논문의 구성은 다음과 같다. 2장에서는 CUDA와 PNG에 대해서 설명하고 3장에서는 PNG 압축 알고리즘을 설명하고 병렬 PNG 필터링 알고리즘을 제안하며 4장에서는 실험을 통해 병렬 압축 알고리즘의 가속화 수준을 평가하고 제안한 알고리즘의 실효성을 검증하며, 마지막으로 5장에서는 기대효과와 향후 연구 주제를 제안하고 논문을 마무리한다.

2. 연구배경

2.1 CUDA

CUDA는 GPGPU를 이용한 병렬 프로그래밍을 C++ 환경에서 구현하기 위해 개발된 수준 높은 프로그래밍 모델이다. 처음 출시된 때에는 C++만 지원했으나 현재는 Python, Java 등 다양한 언어를 통해 구현할 수 있도록 지원하고 있다. 개발 당시에는 CUDA C라는 기존 C언어에 사용자 정의 프로그래밍 언어를 추가하여 GPGPU의 매니코어에 각각 연산을 할당할 수 있도록 설계되었다. CUDA 프로그래밍 모델의 강점은 하드웨어의 병렬성을 극대화하여 활용할 수 있다는 점이다. 그러나 이를 효율적으로 운영하기 위해선 알고리즘 간 의존성을 잘 해결해야 하고 대부분의 프로세스 과정을 작은 프로세스로 나눠 설계해야 한다는 점이다.



(그림 1) CUDA Architecture

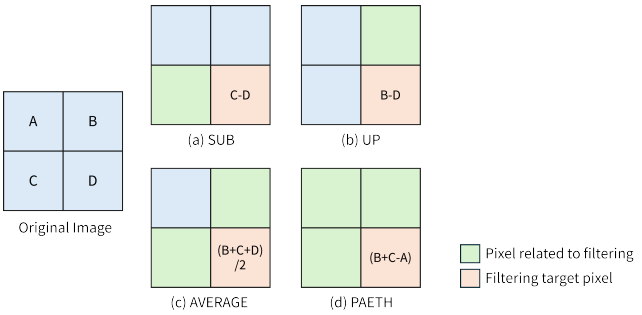
2.2 PNG

PNG는 ETF RFC 2083과 ISO/IEC 15948 상에 표준으로 등재되어있는 이미지 파일 형식 중 하나이다. W3C에서 PNG 파일 형식에 대한 전반적인 특징을 문서화하여 제공하고 있다[6]. PNG 파일 형식의 전반적인 특징으로는 무손실 압축 기법을 활용하는 파일 형식으로 기존 많이 사용되었던 JPEG 표준과 동일한 크기의 이미지를 압축했을 때 비교적 큰 용량을 가지게 된다. PNG 파일 형식의 장점으로는 웹 브라우저나 이미지 편집 프로그램 등에서 광범위적으로 지원되고 있다는 이식성, 기존 Graphics Interchange Format(GIF)의 압축률보다 우수한 압축률을 가지며 여러 단점들을 개선했다는 점, 구성되어있는 여러 알고리즘들의 오픈소스로 공개되어 있다는 점을 들 수 있다.

파일 헤더는 8 바이트 크기의 시그니쳐로 구성되어 있으며 이후 연속적인 청크로 이어져 이미지의 정보들을 포함하고 있다. 청크의 대표적인 예시들로는 IHDR(이미지 크기/색상 정보 등), PLTE(색상 목록), IDAT(실제 이미지 데이터) 등이 있다. 색상 종류로는 그레이스케일, RGB, 인덱스, 그레이스케일과 알파채널, RGB와 알파채널로 5개를 제공하고 있으며 알파 채널은 투명도를 조절할 수 있는 채널로서 사용된다.

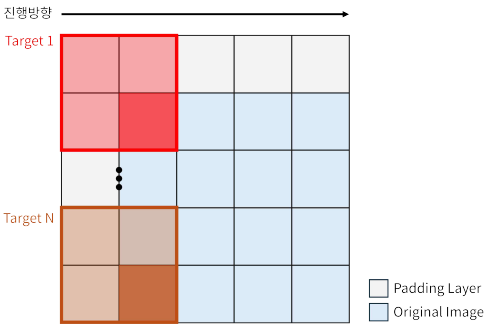
3. 알고리즘 제안

PNG 압축 알고리즘은 그림 2와 같이 총 3단계에 걸쳐 진행된다. 우선적으로 PNG 데이터가 입력값으로 들어오게 되면 다섯 종류의 필터링 기법 중 하나를 선택하여 압축 전 예측 과정이 진행된다. 필터링은 후에 서술할 Deflate 알고리즘 이전에 데이터를 효율적으로 압축이 가능하도록 하는 단계다. 필터링의 종류로는 NONE(0), SUB(왼쪽 픽셀 값을 현재 픽셀 값에서 뺌), UP(위쪽 픽셀 값을 현재 픽셀 값에서 뺌), AVERAGE(처리중인 현재 픽셀과 왼쪽과 위쪽 픽셀값의 평균로 정함), PAETH(위 픽셀과 왼 픽셀을 더하고 왼쪽 대각 상단 픽셀 값을 뺌)이 있다.



(그림 2) PNG Compression Filtering Algorithm

필터링 단계가 종료되고 나면 본격적인 압축이 시작된다. LZ77 알고리즘을 통해 데이터를 압축하고 중복되는 내용을 허프만 코딩을 활용해 재압축한다. 위 논문은 이미지 필터링 방식과 GPGPU 구조의 유사성을 고려하여 병렬 PNG 필터링 알고리즘을 제안한다.



(그림 3) Parallel Filtering

인공지능에서 주로 사용되는 컨볼루션 연산에서 아이디어를 채용해 기존 입력 데이터에 패딩 구역을 추가하여 GPGPU 코어들이 병렬적으로 필터링을 진행하도록 구현했다. 그림 3에서는 4x4 이미지에 패딩 구역을 추가하여 기존 이미지 데이터에 손실이 발생하지 않고 필터링이 가능하도록 구현한 예시를 보여주고 있다. 수행 원리는 다음과 같다. GPU의 각 코어들은 Target Block을 하나씩 담당하여 필터링 계산을 진행하며 왼쪽에서 오른쪽 방향으로 이미지 필터링을 진행한다. 사용자가 설정한 블록 수 파라미터 값과 이미지 크기에 맞게 N 값이 계산된다. 만약 FHD(1920x1080) 크기의 이미지를 대상으로 필터링을 수행하게 된다면, 세로 픽셀 수인 1080을 사용자가 설정한 파라미터 값으로 나눴을 때 N 값에 대한 정보를 알 수 있다.

4. 실험

4.1 실험 설계

실험은 다음과 같이 설계했다. FHD의 이미지를 랜덤하게 생성한다는 가정으로 2차원의 uint8 자료형을 가진 1920x1080 크기의 배열을 4개 생성하였다. 이는 PNG 형식의 이미지의 특징 중 하나인 RGBA 채널에 대응하도록 하기 위해서다. 자료형을 uint8로 선택하게 된 이유는 각 픽셀이 저장할 수 있는 값은 이미지 파일 자체에서도 0에서 255 이기 때문에 잔여 메모리 공간이 발생하지 않도록 했다.

이후 생성된 데이터를 대상으로 직렬 PNG 필터링 연산을 우선 진행하고 이후에는 병렬 PNG 필터링 연산을 종류별로 각 10회 실시, 커널 실행 시간의 평균 시간을 토대로 가속화 정도를 평가했다.

4.2 실험 환경

표 1은 실험을 진행한 하드웨어 사양과 소프트웨어 환경을 나타내고, 표 2는 GPGPU의 주요 세부 성능 지표를 나타내고 있다.

<표 1> Experimental Environment

Hardware Specification	
CPU	AMD Ryzen 7 7800x3D
GPU	NVIDIA RTX 4070Ti Super
RAM	DDR5 32GB
Software Environment	
OS	Windows 11 / WSL(Ubuntu)
CUDA	8.9 version
CUDA Toolkit	12.6 version
측정 프로그램	CUDA Nsight System

<표 2> GPGPU Specification

GPGPU Specification	
CUDA Cores	8,448
Boost/Base Clocks (GHz)	2.61/2.34
CUDA Cores per SM	128
L1 Cache Size per SM	128KB

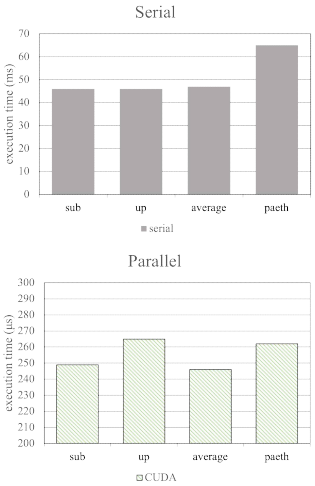
4.3 실험 결과

실험 결과는 그림 4와 같다. Nvidia사에서 제공하는 정밀 측정 프로그램인 Nsight System을 사용해 각 커널 시간을 측정했으며. 블록 사이즈는 Warp 사이즈를 고려하여 32로 설정했다.

병렬 필터링 알고리즘의 커널 실행시간은 직렬 필터링 알고리즘 대비 가장 빠른 가속화는 약 192배, 가장 느린 가속화는 약 170배 정도의 성능 향상이 발생했다. Up 필터링의 경우에는 필터링 알고리즘 자체적인 이유로 캐시 미스에 의한 성능 향상이 제한적으로 발생했다.

5. 결론

본 논문은 병렬 PNG 필터링 알고리즘을 제안하였으며 이는 PNG 압축 알고리즘의 가장 처음 단계인 직렬 필터링 알고리즘 대비 약 190배의 향상을 보임을 확인할 수 있었다.



(그림 4) 실험 결과

이를 통해 PNG 형식 또한 병렬처리의 효과를 획기적으로 볼 수 있음을 확인할 수 있었다. 향후 연구 주제로는 데이터에 의존성을 받지 않기 때문에 필터링 알고리즘에 Multi-Stream 환경을 도입하여 알고리즘을 개선하는 것과 블록 구조를 2D 환경으로 바꿔 Cache 적중률을 향상하는 방안을 고려해 추가적인 가속화가 가능하도록 구현하며 LZ77과 허프만 코딩 알고리즘의 병렬화를 구현해 온전한 병렬 PNG 압축 알고리즘을 제안하고자 한다.

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 정보통신방송혁신인재양성(메타버스융합대학원)사업 연구 결과로 수행되었음
(IITP-2025-RS-2023-00256615)

참고문헌

[1] D. Liu and X. Y. Fan, "Parallel program design for JPEG compression encoding," 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China, pp. 2502-2506, 2012

[2] Y. Nishikawa, S. Kawahito and T. Inoue, "A parallel image compression system for high-speed cameras," IEEE International Workshop on Imaging Systems and Techniques, 2005, Niagara Falls, Ontaro, Canada, pp. 53-5, 2005

[3] Fushan Zhu, Hua Yan, “An efficient parallel entropy coding method for JPEG compression based on GPU,” The Journal of Supercomputing, Volume 78, Issue 2, 2022, pp.2681-2708