GPGPU 프로그래밍 모델의 기술 동향

이현진, 정유나, 이성길
성균관대학교 컴퓨터공학과*
e-mail: {dy7aa, jeongyuna, sungkil}@skku.edu

Survey on GPGPU Programing Models

Hyunjin Lee, Yuna Jeong, and Sungkil Lee
Dept. of Computer Engineering, Sungkyunkwan University

요 약

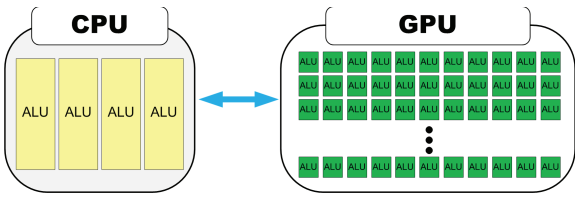
대용량 영상 데이터 처리를 위한 GPU 는 많은 코어들을 이용한 병렬 작업을 통해 결과를 도출한다. 단순 수치 연산에 특화된 이러한 GPU 의 계산 능력을 다른 분야로 확장시켜 적용하고자 하는 시도인 GPGPU 는 이전부터 꾸준히 시도되고 있다. 그러나 GPU 의 난해하고 생소한 프로그래밍으로, 작성이 쉽지 않고 현격한 성능 향상을 기대하기 어렵다. 이에, 이러한 GPGPU 프로그래밍의 어려움을 해결하고자 여러 프로그래밍 모델들이 등장하였다. 본 논문에서는 GPGPU 프로그래밍을 위한 대표적인 모델인 CUDA, OpenCL, C++ AMP, 그리고 OpenACC 에 대해 살펴본다.

1. 서론

GPU (Graphics Processing Unit)란 컴퓨터의 그래픽스 연산을 전문적으로 담당하는 하드웨어로, 3D 그래픽스 렌더링을 하는데 주목적이 있다. GPU 는 부동 소수점 연산을 하는 소형의 코어 수천 개로 구성되어 있어 병렬로 데이터를 처리하므로, 소수의 코어로 구성된 CPU 에 비해 그 성능이 매우 뛰어나다 (그림 1).

근래 고정된 형태의 GPU 를 사용하던 방식에서 프로그램 가능한 보다 유연한 형태의 GPU 로 넘어오면서, 3D 그래픽스 분야를 위해서만 사용되던 GPU 에 대한 새로운 접근 시도되었다. 이는 GPU 가 그래픽 렌더링에 주로 쓰이는 행렬과 벡터 연산에 높은 계산 성능을 보이는 것에서 기인한 것으로, 그래픽스 연산뿐 아니라 일반 컴퓨팅 영역에서도 GPU 를 활용하고자 하는 시도이다. 이러한 컴퓨팅 체계는 GPGPU (General-Purpose GPU)라 부른다.

마이크로소프트사에서는 2006 년 11 월 배포된 DirectX 10.0 SDK 에 다이렉트컴퓨트 (DirectCompute)라는 GPGPU 플랫폼을 추가했다. 하지만 쉐이더 기반의 다이렉트컴퓨트는 GPU 사용자들이 범용적인 목적으로 사용하기에는 다소 생소한 방법을 제공하고 있었다. 이에, GPGPU 프로그래밍을 위해 제공되는 여러 프로그래밍 모델들이 등장하게 되었다. 대표적으로는 NVIDIA 의 CUDA 와 OpenACC, 크로노스 그룹의 OpenCL, 마이크로소프트사의 C++ AMP 등이 있다. 본 논문에서는 대표적인 GPGPU 프로그래밍 모델들의 특징 및 기술 동향에 대해서 살펴보도록 한다.



(그림 1) CPU 와 GPU 의 구조 차이.

2. 기술동향

수천 개의 코어로 이루어진 그래픽카드를 범용적으로 이용하는 GPGPU 는 병렬 처리에 적합한 연산에서 뛰어난 연산 속도를 낼 수 있다는 잇점이 있어 상당히 매력적인 기술이나, GPU 병렬 프로세싱을 위한 프로그래밍 작성을 위해 넘어서야 할 진입장벽이 높다. 이에, GPU 관련 업체들은 개발자들이 보다 쉽고 편리하게 개발할 수 있도록 프로그래밍 모델을 제시하고 있으며, 보다 편리한 GPGPU 프로그래밍 작성을 위해 다양한 접근방법들이 시도되고 있다.

2.1 CUDA (Compute Unified Device Architecture)

2006 년 NVIDIA 에서는 GPU 개발을 위한 툴로 CUDA 를 소개했다. CUDA 는 병렬 컴퓨팅 플랫폼으로,

* 이 논문은 2012 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구, 중견연구자 사업 지원을 받아 수행된 것임 (No. 2011-0014015, 2012R1A2A2A01045719).

대량의 GPU 코어로 컴퓨팅 속도를 현격히 향상 시킬 수 있는 프로그래밍 모델이다. 이는 C 언어 기반의 직관적인 GPI 프로그래밍을 제공하며, 공유 메모리를 사용하여 빠른 연산을 가능하게 한다. CUDA 는 CUDA Runtime API 와 Driver API 2 가지로 구성되어 있다. Runtime API 는 설정을 비롯한 여러 필요한 값들을 자동으로 할당해 놓음으로써 사용자의 편의를 돕고자 제공되는 것이고, 실제로 Runtime API 가 동작할 수 있도록 돕는 Driver API 는 메모리나 장치 등 직접적인 관리가 필요한 경우, Runtime API 를 사용하지 않고 프로그래밍을 하고자 하는 경우를 위해 제공되는 것이다. 이러한 CUDA 는 시뮬레이션과 같이 대량의 연산을 요구하는 다양한 분야의 병렬 처리 연산에 적합한 작업을 수행할 경우에 적용하여 뛰어난 성능 향상을 기대할 수 있다. 그러나 제조사에 특화되어, NVIDIA 의 GPU, 그 중에서도 G8X GPU 로 구성된 Geforce 8 시리즈 급 이상만 지원한다는 한계가 존재하며, 특정한 소프트웨어를 요구하는 제한된 호환성을 지닌다 [1, 2].

2.2 OpenCL (Open Computing Language)

크로노스 그룹에서 유지 및 관리되고 있는 OpenCL 은 애플, AMD, IBM, 인텔, 그리고 NVIDIA 에서 개발한 개방형 범용 병렬 컴퓨팅 프레임워크로, GPU 뿐만 아니라 CPU 와 기타 프로세서로 이루어진 이기종 (heterogeneous) 컴퓨터 시스템을 위한 산업 표준 프로그래밍 모델이다. OpenCL 은 데이터와 태스크 기반의 병렬 프로그래밍 모델이며, 오픈 스펙의 특징을 가져 GPU, CPU, 그리고 DSP 등도 함께 사용할 수 있다 [3].

OpenCL 은 C, 런타임, API 세 부분으로 구성되어 있다. C 기반의 크로스플랫폼 프로그래밍 인터페이스를 제공하고 방대한 계산에 대한 하드웨어 추상화를 제공해주는 API 를 사용하며, 스케쥴과 계산, 메모리를 관리한다. OpenCL 은 CUDA 에서도 도입되어 개발자들이 NVIDIA GPU 상에서 이기종 컴퓨팅 환경을 제공해주는 역할을 하고 있다.

OpenGL 과 완벽하게 연동 되는 OpenCL 은 모바일 임베디드 분야에서도 사용이 가능하지만, 제조사인 NVIDIA 와 인텔의 지원 부족으로 인해 이기종 플랫폼 구현이 제한된다는 한계점이 있다. 또한, C99 스탠다드 헤더를 사용할 수 없다는 제한이 있다.

2.3 C++ AMP (C++ Accelerated Massive Parallelism)

마이크로소프트사가 개발한 C++ AMP 는 CPU 와 GPU 를 사용한 이기종 컴퓨팅을 위한 개방형 프로그래밍 언어이다. Visual Studio 2012 에 추가된 기능인 C++ AMP 는 GPU 를 이용하여 C++ 코드의 실행 속도를 높이고자 하는 목적을 가지고 있다. C++ 기반의 GPU 활용을 가능하게 하는 C++ AMP 는 DirectX API 에 대한 이해 수준과 응용 능력에 상관 없이 GPU 를 활용한 범용 프로그래밍을 돕고자 하였다. C++ AMP 는 기존 개발자들에게 친숙한 GPGPU 프로그래밍 작성을 위해, C 언어 대신 C++ 기반으로, STL 과 비슷하게 설계하였다. C++ AMP 는 람다 (Lamda) 식과 병렬 반복문을 결합한 형태의 구조로, 병렬 반복문을 통해 C++ AMP 가 얼마나 많은 스레드를 생성할 것 인지와 생성한 스레드들을 식별하기 위한 ID 를 부여한다 [4].

C++ AMP 는 C++ 개발자에게 익숙한 형태로 제안되어 GPGPU 개발을 보다 쉽게 하고, Visual Studio 2012 IDE (Integrated development environment)로 기존의 디버깅이 어려운 문제를 해결해주었다는 이점이 있지만, Windows 7 에서는 디버깅이 불가하고, Windows 8 운영체제와 DirectX 11.0 이상을 필요로 한다는 플랫폼 의존적인 점과 Visual Studio 2012 에서만 사용이 가능하다는 한계가 있다.

2.4 Open ACC

CUDA 와 같은 플랫폼의 등장으로 GPU 프로그래밍의 개념과 유용성은 널리 알려졌지만, 병렬 컴퓨팅을 잘 활용하여 현격한 성능향상을 얻으며 프로그램을 작성하기란 쉽지 않다. NVIDIA 는 이러한 프로그래밍의 어려움을 해결하기 위하여, CUDA 를 좀 더 추상화시킨 컴파일러 지시문 (directive) 기반 프로그래밍 모델인 OpenACC 를 소개하였다. 지시어 기반의 Open ACC 는 개발자에게 비교적 간편한 프로그래밍 환경을 제공하기 때문에 더 높은 생산성을 기대할 수 있는 프로그래밍 모델이다. 앞서 언급한 다른 프로그래밍 모델과 달리 운영체제와 플랫폼에서 Cross-platform 의 성격을 가져 의존도가 매우 낮다는 이점 또한 가지고 있다. 그렇다고 해서 OpenACC 가 최적의 솔루션이라고 보기에는 이르다. CUDA 를 추상화한 결과물인 OpenACC 는 최적화된 CUDA 프로그램과 비교해보면 성능 면에서 한계를 발견할 수 있다 [5].

3. 비교

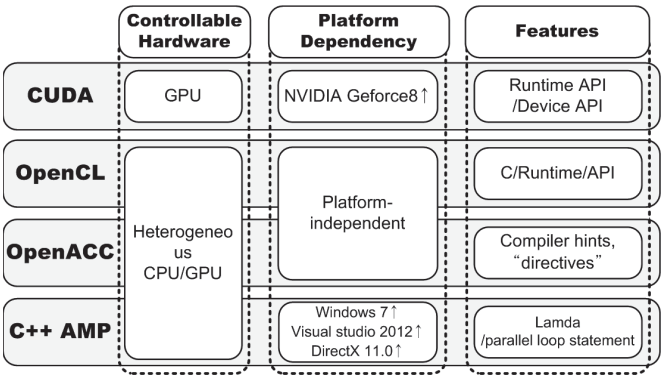
CUDA, OpenCL, C++ AMP, 그리고 OpenACC 등의 API 들은 서로 대응 되는 것을 목적으로 설계된 것이 아니기 때문에, 이를 성능 면에서 비교하기는 어렵다. 따라서 앞서 설명한 각 API 환경의 특징을 기반으로 이를 비교해보고자 한다 (그림 2).

먼저 API 를 통해 제어할 수 있는 하드웨어를 살펴보면, OpenCL, C++ AMP, OpenACC 는 이기종 컴퓨팅 환경을 제공하지만, CUDA 는 GPU 관리만을 목적으로 한다.

다음으로 프로그래밍 환경의 의존도를 살펴보면, CUDA 는 자사의 특정 GPU 상에서만 동작하고, 마이크로소프트의 C++ AMP 는 자사의 OS 와 프로그램이 전제되어야 한다는 한계를 지니고 있다. 이러한 플랫폼 의존적인 특징은, 해당 프로그래밍 환경으로 개발된 솔루션의 동작여부에 불확실성을 주기 때문에, 개발자의 언어선택에 있어 신중히 고려되어야 한다.

GPGPU 를 처음 접하는 개발자에게 있어, 디바이스와 메모리 관리가 필요한 OpenCL 과 CUDA device API 는 학습이 용이하지 못하다. 이에 반해 Visual studio 2012 의 디버깅 기능 사용이 가능한 C++ AMP 나, 컴파일러 힌트를 통해 CUDA 를 추상화시킨 OpenACC

는 비교적 진입장벽을 낮춘 API 라고 볼 수 있다. OpenCL 의 경우 직관적인 API 이름을 사용하여, 다른 언어에 비해 CPU 와 GPU 사용이 용이하다. OpenCL 이 다양한 설정을 요하는 것에 비해, CUDA 의 런타임 API 는 비교적 적은 API 호출을 통해 setup 과 release 가 가능하다.



(그림 2) CUDA, OpenCL, C++ AMP, OpenACC 비교.

4. 결론

본 논문에서는 GPU 의 자원을 3D 그래픽스 이외의 분야에 사용하는 GPGPU 프로그래밍의 대표적인 모델들을 살펴보았다. GPU 의 발전이 3D 그래픽스 관련 분야인 게임이나 엔터테이먼트 산업을 발전하게 했다면, 이제는 더 나아가 CPU 의 영역으로 그 영향이 확장되어 그 활용범위는 매우 넓다. 이에, 꾸준히 향상되고 있는 그래픽 하드웨어의 성능 아래, 최대의 효율을 이끌어 내는 런타임 프로그램을 만들기 위해서는, GPU 리소스를 연산 영역에 사용하는 GPGPU 기능에 대한 이해와 적용이 필수적이라 할 수 있다. 이에, 이러한 GPGPU 프로그래밍 자성을 돕는 프로그래밍 모델의 활용은 필수적이며, 이에 대한 연구 또한 활발히 진행될 것으로 전망된다.

참고문헌

[1] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High Performance Network Intrusion Detection Using Graphics Processors. In Proc. of Recent Advanced in Intrusion Detection (RAID), 2008

[2] NVIDIA. 2013. CUDA; http://www.nvidia.com/CUDA.

[3] Macworld. 2013. Open CL; http://www.macworld.com/article/1136921/opencl.html

[4] msdn. 2013. C++ AMP; http://msdn.microsoft.com/en-us/library/vstudio/hh265137.aspx

[5] OpenACC. 2013. Open ACC; http://www.openacc-standard.org