백서

NVIDIA의 차세대 CUDA™ 컴퓨팅 아키텍처:

Kepler™ GK110

*최고 수준의 속도와 효율성을 자랑하는 **HPC** 아키텍처*

목차

Kepler GK110 – 차세대 GPU 컴퓨팅 아키텍처 3
Kepler GK110 – 극한의 성능과 극한의 효율성 4
- 동적 병렬 처리(Dynamic Parallelism) 5
- Hyper-Q 5
- 그리드 관리 유닛 5
- NVIDIA GPUDirect™ 5

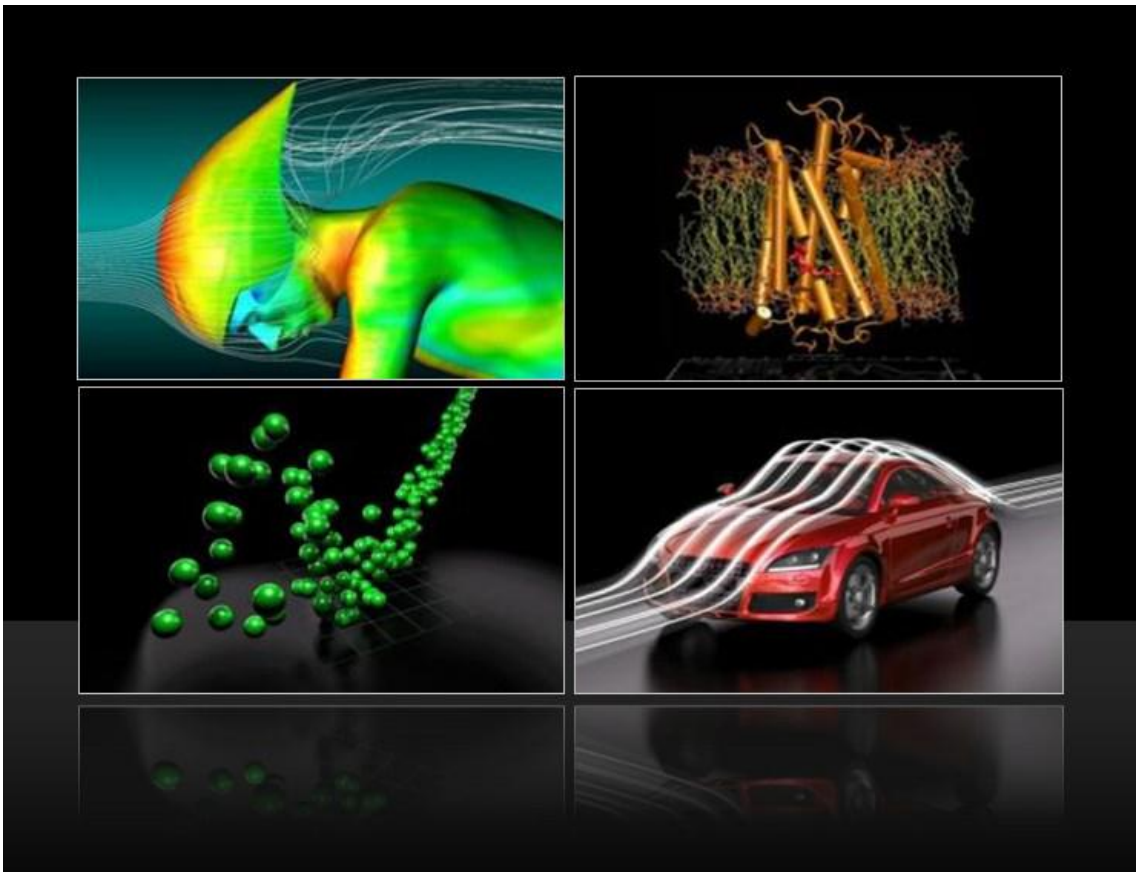
GK110 Kepler 아키텍처 개요 6
와트당 성능 7
SMX(스트리밍 멀티프로세서) 아키텍처 8
SMX 프로세싱 코어 아키텍처 9
쿼드 와프 스케줄러(Quad Warp Scheduler) 9
새로운 ISA 인코딩: 스레드당 255개 레지스터 11
셔플 명령 11
원자적 연산 12
텍스처 향상 13
Kepler 메모리 서브시스템 – L1, L2, ECC 14
64 KB 구성 가능 공유 메모리 및 L1 캐시 14
48KB 읽기 전용 데이터 캐시 15
향상된 L2 캐시 15
메모리 보호 지원 15
동적 병렬 처리 16
Hyper-Q 19
그리드 관리 유닛 - 효율적인 GPU 활용도 관리 21
NVIDIA GPUDirect™ 22

맺음말 24
부록 A - CUDA 요약 정보 25
CUDA 하드웨어 실행 26

Kepler GK110 – 차세대 GPU 컴퓨팅 아키텍처

과학계, 의료계, 엔지니어링, 금융계 등 다양한 분야에 걸쳐 고성능 병렬 컴퓨팅에 대한 수요가 증가함에 따라 엔비디아는 차원이 다른 강력한 성능을 갖춘 GPU 컴퓨팅 아키텍처로 지속적인 혁신을 통해 수요에 대응하고 있다. 엔비디아의 기존 Fermi GPU는 이미 탄성파 처리, 생화학 시뮬레이션, 날씨 및 기상 모델링, 신호 처리, 금융 공학, 기계 공학, 계산 유체 역학, 데이터 분석 등 다방면에서 HPC(고성능 컴퓨팅) 기능을 재정의하고 그러한 기능의 도입 속도를 높이고 있다. 엔비디아의 새로운 Kepler GK110 GPU는 이러한 병렬 컴퓨팅의 기준을 한 차원 높여 아무리 까다로운 컴퓨팅 과제도 거뜬히 해결하도록 지원하게 된다.

Kepler GK110은 GPU에서 병렬 워크로드 실행을 최적화하고 처리량을 늘릴 수 있는 새로운 방식을 통해 이전 GPU 세대보다 훨씬 높은 수준의 처리 성능을 제공함으로써 병렬 프로그램의 개발을 간소화하고 고성능 컴퓨팅의 일대 혁신을 불러올 것이다.

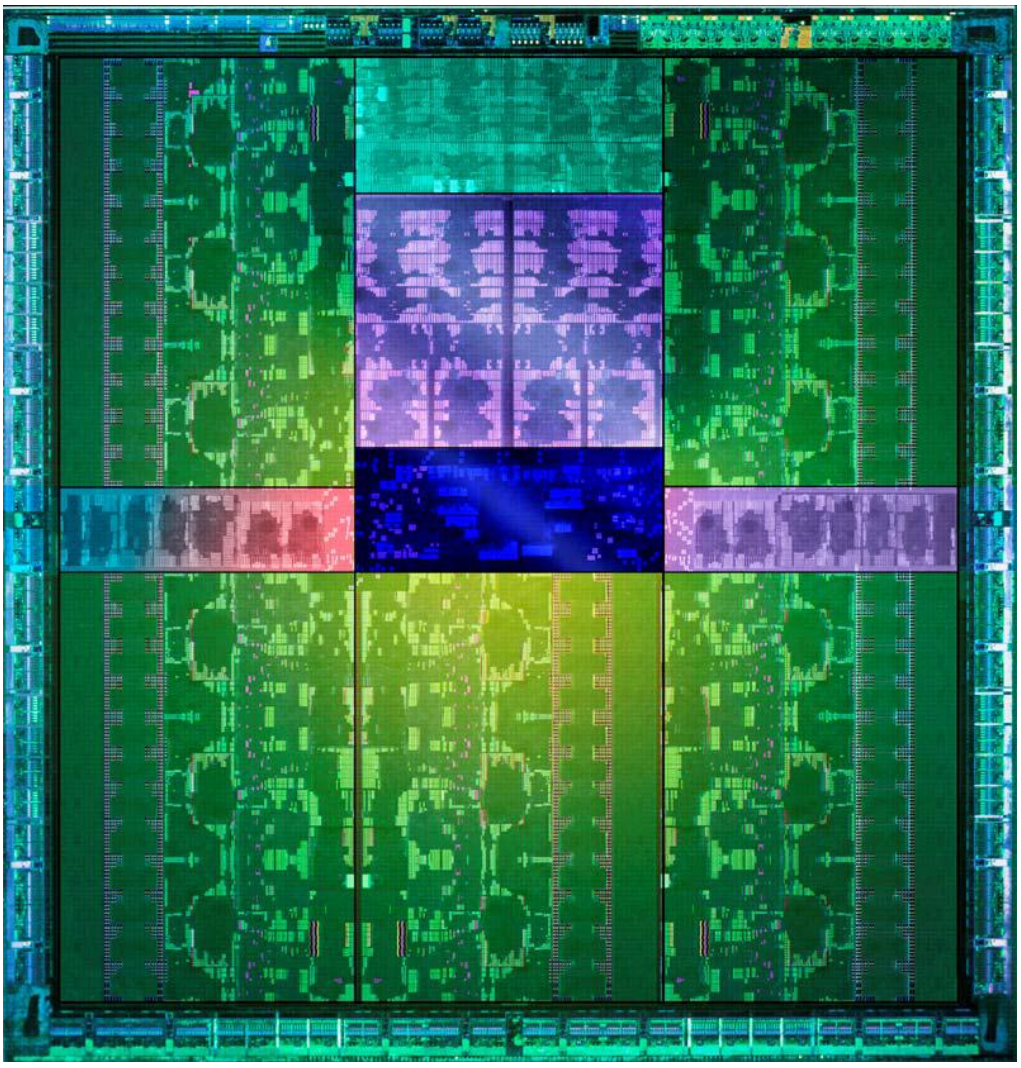


Kepler GK110 - 극한의 성능과 극한의 효율성

71억 개의 트랜지스터로 구성된 Kepler GK110은 업계에서 속도가 가장 빠를 뿐만 아니라 설계가 가장 복잡한 마이크로프로세서이기도 하다. GK110은 Tesla® 및 HPC 시장에서 병렬 처리 분야를 견인할 대표 제품이 될 수 있도록 컴퓨팅 성능에 주안점을 두고 새롭고 혁신적인 기능을 많이 추가했다.

Kepler GK110은 이전 Fermi 아키텍처의 60-65%보다 크게 향상된 80% 이상의 DGEMM(배정도 행렬연산) 효율성으로 1TFlop 이상의 배정밀도 처리량을 제공한다.

대폭 향상된 처리 성능과 함께, Kepler 아키텍처는 전력 효율성에 있어서도 Fermi보다 3배 향상된 와트당 성능을 제공한다.



Kepler GK110 다이 사진

Kepler GK110은 GPU 활용도를 높이고, 병렬 프로그램 설계를 간소화하며, 개인용 워크스테이션에서 슈퍼컴퓨터에 이르기까지 폭넓은 컴퓨팅 환경 스펙트럼에 걸쳐 GPU를 구축할 수 있는 다음과 같은 새로운 기능을 제공한다.

- **동적 병렬 처리(Dynamic Parallelism)** – CPU와는 별개로 GPU에서 자체적으로 처리할 새로운 작업을 생성하고, 결과를 동기화하고, 가속화된 전용 하드웨어 경로를 통해 해당 작업의 스케줄링을 제어하는 기능을 추가한다. 프로그램 실행의 전 과정에서 작업량과 형태에 맞춰 병렬 처리를 실행하는 유연성을 제공하므로, 프로그래머가 보다 다양한 병렬 작업을 노출시키고 GPU의 연산 효율성을 극대화할 수 있다. 이러한 기능 덕분에 정형화되지 않고 복잡한 작업의 쉽고 효율적인 실행이 가능해지면서 애플리케이션의 더 많은 부분을 전적으로 GPU에서 실행할 수 있게 된다. 또한 프로그램을 작성하기도 더 쉬워지고, 확보된 CPU 성능을 다른 작업에 더 활용할 수 있다.

- **Hyper-Q** – Hyper-Q는 여러 CPU 코어에서 단일 GPU의 작업을 동시에 실행하여 GPU 활용도를 높이고 CPU 유휴 시간을 크게 줄이는 기능이다. Hyper-Q는 호스트와 GK110 GPU 간의 총 연결(작업 대기열) 수를 늘려 하드웨어에서 관리되는 동시 연결 수를 32개(Fermi의 경우 단일 연결만 가능)까지 지원한다. Hyper-Q는 여러 CUDA 스트림, 여러 MPI(메시지 전달 인터페이스) 프로세스 또는 프로세스 내의 여러 스레드에서 각각 연결할 수 있도록 하는 유연한 솔루션이다. 이전에 여러 작업에 걸쳐 직렬화 오류가 발생해 GPU 활용에 한계가 있었던 애플리케이션도 기존 코드를 변경하지 않고 성능이 대폭 개선되는 효과를 볼 수 있다.

- **그리드 관리 유닛** – 동적 병렬 처리를 지원하려면 유연하고 수준 높은 그리드 관리 및 디스패치 제어 시스템이 필요하다. 새로운 GK110 GMU(그리드 관리 유닛)는 GPU에서 실행할 그리드를 관리하고 우선 순위를 지정한다. 새로운 그리드의 디스패치를 일시 중지하고 대기 중이거나 일시 중단된 그리드를 실행할 준비가 될 때까지 대기열에 넣을 수 있기 때문에 동적 병렬 처리와 같은 강력한 런타임을 구현하는 데 필요한 유연성이 확보된다. 이러한 GMU는 CPU와 GPU에서 생성된 워크로드가 정상적으로 관리되고 디스패치되도록 보장한다.

- **NVIDIA GPUDirect™** – NVIDIA GPUDirect™는 단일 컴퓨터의 GPU 또는 네트워크에 분산된 여러 서버의 GPU가 CPU/시스템 메모리를 거치지 않고 직접 데이터를 교환할 수 있게 하는 기능이다. GPUDirect의 RDMA 기능은 SSD, NIC, IB 어댑터 등의 타사 디바이스가 단일 시스템에 설치된 여러 GPU의

메모리에 직접 액세스할 수 있도록 하여 GPU 메모리에서 메시지를 주고받는 MPI의 지연 시간을 크게 줄여 준다. 또한 시스템 메모리 대역폭 요구량을 줄이고 다른 CUDA 작업에 사용할 수 있도록 GPU DMA 엔진의 작업 부하도 줄인다. Kepler GK110은 P2P, 비디오용 GPUDirect 기능 등 다른 GPUDirect 기능도 지원한다.

GK110 Kepler 아키텍처 개요

주로 Tesla용으로 제작된 Kepler GK110은 세계 최고 성능의 병렬 컴퓨팅 프로세서를 제공하는 것을 목표로 한 제품이다. GK110은 Fermi에서 본래 제공되는 컴퓨팅 성능을 상회할 뿐 아니라 소비 전력이나 발열도 훨씬 적게 효율적으로 작동한다.

Kepler GK110의 정식 구현 모델에는 SMX 유닛 15개와 64비트 메모리 컨트롤러 6개가 포함된다. GK110의 구성은 제품마다 달라진다. 예를 들어 SMX가 13개 또는 14개 구축되는 제품도 있다.

아래에서는 다음과 같은 아키텍처의 주요 특징을 좀 더 자세히 설명한다.

- 새로운 SMX 프로세서 아키텍처
- 서브시스템 추가 캐싱 기능, 계층 구조의 각 레벨별 추가 대역폭, 완전히 새로 설계되어 속도가 크게 개선된 DRAM I/O 구현 등을 제공하는 향상된 메모리
- 새로운 프로그래밍 모델 기능 지원을 위해 설계 전반에 적용된 하드웨어 지원

Kepler GK110 전체 칩 블록 표

Kepler GK110은 새로운 CUDA Compute Capability 3.5를 지원한다(CUDA의 개요는 ***부록 A - CUDA 요약 정보*** 참조). 다음 표에서는 Fermi GPU 아키텍처와 Kepler GPU 아키텍처의 서로 다른 컴퓨팅 기능을 비교하고 있다.

	FERMI GF100	**FERMI GF104**	**KEPLER GK104**	**KEPLER GK110**
컴퓨팅 성능	**2.0**	**2.1**	**3.0**	**3.5**
스레드/와프	**32**	**32**	**32**	**32**
멀티프로세서당 최대 와프수	**48**	**48**	**64**	**64**
멀티프로세서당 최대 스레드 수	**1536**	**1536**	**2048**	**2048**
멀티프로세서당 최대 스레드 블록 수	**8**	**8**	**16**	**16**
멀티프로세서당 32비트 레지스터 수	**32768**	**32768**	**65536**	**65536**
스레드당 최대 레지스터 수	**63**	**63**	**63**	**225**
스레드 블록당 최대 스레드 수	**1024**	**1024**	**1024**	**1024**
공유 메모리 크기 구성(바이트)	**16K** **48K**	**16K** **48K**	**16K** **32K** **48K**	**16K** **32K** **48K**
최대 X 그리드 크기	2^16-1	2^16-1	2^32-1	2^32-1
Hyper-Q	**X**	**X**	**X**	O
동적 병렬 처리	**X**	**X**	**X**	O

Fermi GPU와 Kepler GPU의 컴퓨팅 기능

와트당 성능

Kepler 아키텍처의 주된 설계 목표 중 하나는 전력 효율을 높이는 것이었다. NVIDIA 엔지니어들은 Kepler를 설계하면서 Fermi 개발을 통해 얻은 노하우를 총동원하여 고효율로 작동하도록 Kepler 아키텍처를 최적화했다. 전력 소비를 줄이는 데 있어서 TSMC의 28nm 제조 공정이 큰 역할을 했지만, 우수한 성능을 유지하면서 소비 전력을 최소화하기 위해서는 GPU 아키텍처 자체를 여러 가지로 수정해야 했다.

Kepler의 모든 하드웨어 유닛은 뛰어난 와트당 성능을 제공하도록 설계되고 다듬어졌다. 뛰어난 와트당 성능을 보여 주는 가장 좋은 예로 Kepler GK110의 새로운 SMX(스트리밍 멀티프로세서)를 들 수 있다. 최근에 Kepler GK104에서 선보인 SMX 유닛과 많은 부분이 유사하지만 컴퓨팅 알고리즘을 처리할 배정밀도 유닛이 훨씬 많이 포함되어 있다.

SMX(스트리밍 멀티프로세서) 아키텍처

Kepler GK110의 새로운 SMX는 혁신적인 기능의 아키텍처 적용으로 이전의 어떤 멀티프로세서보다 뛰어난 성능과 최고의 프로그래밍성 및 전력 효율성을 제공한다.

SMX: 단일 정밀도 CUDA 코어 192개, 배정밀도 유닛 64개, SFU(특수 기능 유닛) 32개 및 LD/ST(로드/저장 유닛) 32개

SMX 프로세싱 코어 아키텍처

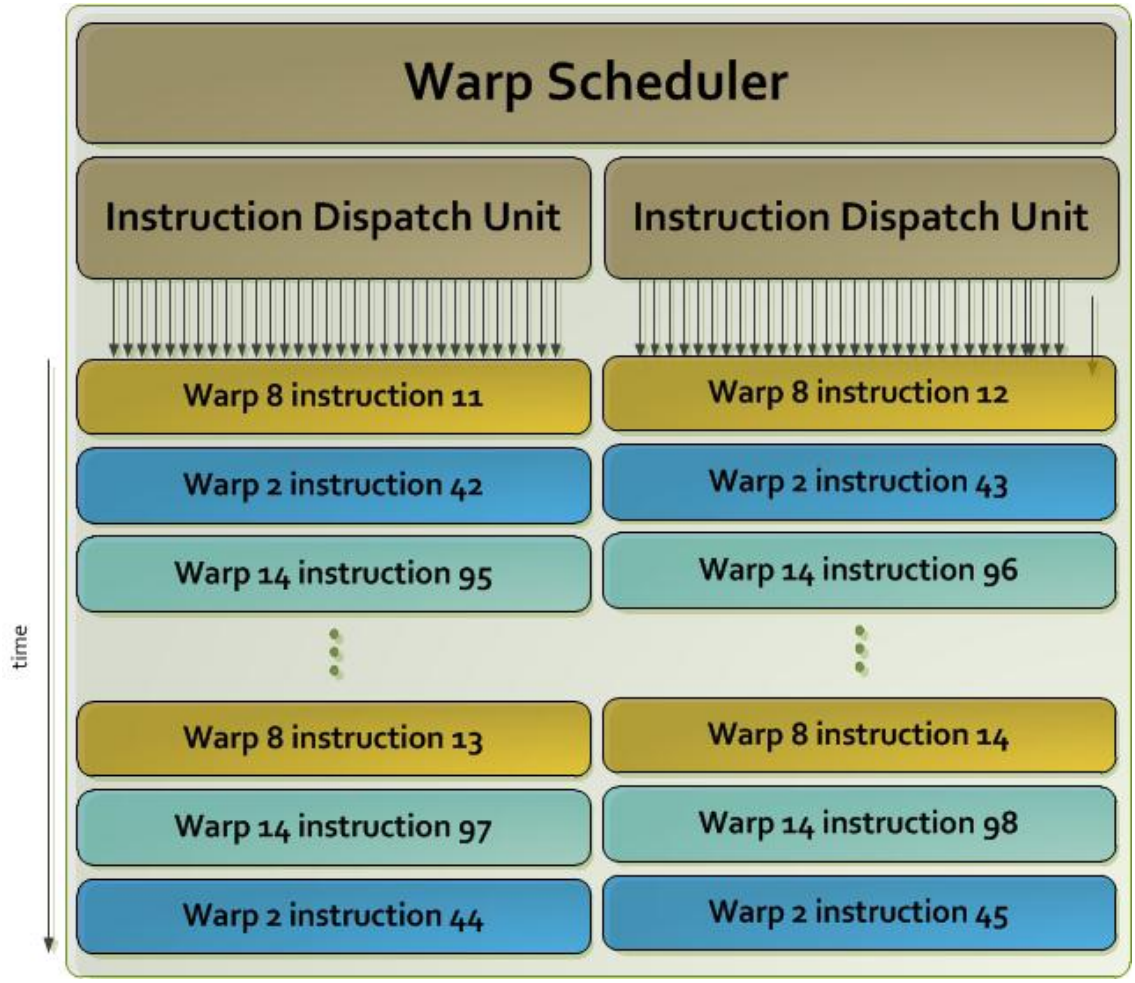
각각의 Kepler GK110 SMX 유닛에는 단일 정밀도 CUDA 코어 192개가 사용되었으며, 각 코어에는 완전 파이프라인 방식의 부동 소수점 및 정수 연산 논리 유닛이 있다. Kepler에는 FMA(Fused Multiply-Add) 연산을 비롯하여 Fermi에 적용되었던 IEEE 754-2008 호환 단일 정밀도 및 배정밀도 연산이 그대로 사용되었다.

배정밀도 연산은 많은 HPC 애플리케이션의 핵심이 되는 기능이기 때문에 GPU가 제공하는 배정밀도 성능을 대폭 향상시키는 것도 Kepler GK110 SMX의 설계 목표 중 하나였다. Kepler GK110의 SMX에는 이전 세대 GPU와 마찬가지로 빠른 근사치 초월 연산 성능을 제공하기 위해 SFU(특수 기능 유닛)도 사용되었다. 단, SFU의 수가 Fermi GF110 SM보다 8배 늘어났다.

GK104 SMX 유닛과 마찬가지로 새로운 GK110 SMX 유닛 내의 코어도 2배속 쉐이더 클럭 대신 주 GPU 클럭을 사용한다. 2배속 쉐이더 클럭은 G80 Tesla 아키텍처 GPU에서 처음 선보인 이래, 모든 Tesla 및 Fermi 아키텍처 GPU에 사용되어 왔다. 실행 유닛이 더 높은 클럭 속도로 실행되기 때문에 칩에서 더 적은 수의 실행 유닛 사본을 사용하여 정해진 목표 처리량을 달성할 수 있다. 이는 본질적으로 면적을 최적화한 것이지만 빨라진 코어의 클러킹 로직은 전력 소모가 심하다. Kepler의 최우선 목표는 와트당 성능을 높이는 것이었다. 면적과 전력의 측면에서 모두 이점이 있는 최적화 방식도 많이 적용했지만, 면적 비용이 다소 가중되더라도 전력을 최적화하는 쪽을 택했다. 그결과 많은 수의 처리 코어를 더 낮은 GPU 클럭으로 실행하여 전력 소모를 줄일 수 있었다.

쿼드 와프 스케줄러(Quad Warp Scheduler)

SMX는 32개의 병렬 스레드로 이루어진 와프(Warp)라는 그룹 단위로 스레드를 스케줄링한다. SMX마다 4개의 스케줄러와 8개의 명령 디스패치 유닛이 있어 와프를 4개씩 동시에 전달하고 실행할 수 있다. Kepler의 쿼드 와프 스케줄러는 와프를 4개 선택하는데, 각 사이클마다 와프당 2개씩 개별 명령을 디스패치할 수 있다. 배정밀도 명령과 다른 명령을 함께 디스패치할 수 없는 Fermi와는 달리 Kepler GK110에서는 로드/저장, 텍스처, 일부 정수 명령 등 레지스터 파일 읽기가 없는 다른 특정 명령이 배정밀도 명령과 쌍을 이룰 수 있다.



각 Kepler SMX에는 4개의 와프 스케줄러가 포함되어 있으며, 각 스케줄러에는 명령 디스패치 유닛이 2개씩 있다. 위에는 단일 와프 스케줄러 유닛이 나와 있다.

SMX 와프 스케줄러 로직의 성능도 최적화하고자 했다. 예를 들어 Kepler와 Fermi의 스케줄러에는 스케줄링 기능을 처리하는 다음과 같은 유사한 하드웨어 유닛이 포함되어 있다.

a. 지연 시간이 긴 연산(텍스처 및 로드)에 대한 레지스터 스코어보딩
b. 와프 간 스케줄링 결정(예: 가능한 후보들 중 다음으로 실행할 최적의 와프 선택)
c. 스레드 블록 레벨 스케줄링(예: GigaThread 엔진)

하지만 Fermi의 스케줄러에는 산술 데이터 경로 자체의 데이터 해저드를 방지하기 위한 복잡한 하드웨어 단계도 포함되어 있다. 멀티포트 레지스터 스코어보드는 유효한 데이터가 아직 준비되지 않은 모든 레지스터를 추적하고, 종속성 검사 블록은 완전히 디코딩된 다수의 와프 명령에 걸쳐 스코어보드에 대해 레지스터 사용량을 분석하여

전달하기에 적합한 명령을 결정한다.

Kepler의 경우 이 정보가 확정적(산술 파이프라인 지연이 가변적이지 않음)이기 때문에 명령을 전달할 준비가 되는 시점을 컴파일러에서 사전에 확인하고 이 정보를 명령 자체에 제공하는 것이 가능했다. 결과적으로 몇 개의 복잡하고 전력 소모가 큰 블록을 사전에 확인된 지연 정보를 추출하는 단순한 하드웨어 블록으로 대체하고 이러한 블록을 이용해 와프 간 스케줄러 단계에서 와프를 선별해낼 수 있게 되었다.

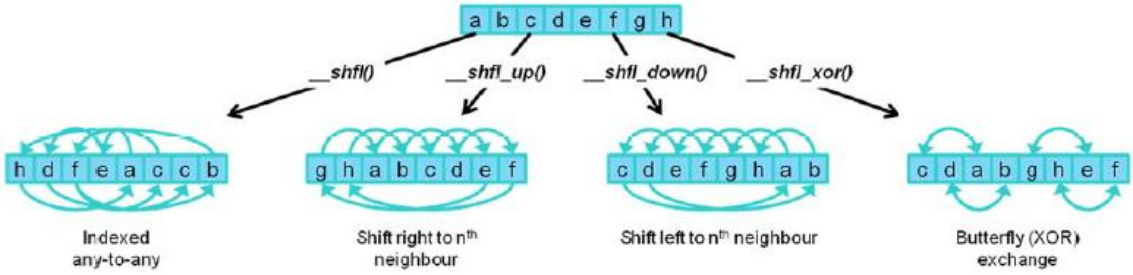
새로운 ISA 인코딩: 스레드당 255개 레지스터

GK110에서는 스레드에서 액세스할 수 있는 레지스터의 수가 4배 증가하여 스레드마다 최대 255개의 레지스터에 액세스할 수 있다. 스레드당 가용 레지스터 수가 늘면서 레지스터 할당률이 높은 코드나 Fermi의 스필링 동작에서 대폭적인 처리 속도 향상을 기대할 수 있게 되었다. 대표적인 예로, CUDA를 사용하여 격자 QCD(양자 크로모 역학) 계산을 수행하는 데 사용되는 QUDA 라이브러리를 들 수 있다. 스레드당 사용할 수 있는 레지스터 수가 늘면서 QUDA fp64 기반 알고리즘의 성능이 5.3배 향상되고 로컬 메모리에 발생하는 스필링도 줄어든다.

셔플 명령

성능을 더욱 향상시키기 위해 Kepler는 와프 내에서 스레드 간에 데이터를 공유할 수 있게 하는 새로운 셔플 명령을 구현한다. 이전에는 와프 내의 스레드 간에 데이터를 공유하기 위해서는 공유 메모리를 통해 데이터를 전달하는 별도의 저장 및 로드 작업이 필요했다. 셔플 명령을 사용하면 스레드가 와프에 포함된 다른 스레드의 값을 거의 모든 순열로 읽을 수 있다. 셔플 기능은 임의로 인덱싱된 참조를 지원하므로, 모든 스레드가 다른 모든 스레드에서 값을 읽을 수 있다. next-thread(고정된 양만큼 위 또는 아래로 오프셋)과 와프의 스레드 간 XOR "버터플라이" 스타일 치환과 같은 유용한 셔플 서브셋도 CUDA instrinsics로 사용할 수 있다.

셔플 기능은 한 단계로 저장 및 로드 작업을 실행할 수 있도록 하여 공유 메모리의 성능을 높여 준다. 또한 와프 레벨에서 교환되는 데이터를 공유 메모리에 저장할 필요가 없기 때문에 스레드 블록당 필요한 공유 메모리 용량도 줄일 수 있다. 와프 내에서 데이터를 공유해야 하는 FFT의 경우 셔플을 사용하는 것만으로도 성능을 6% 향상시킬 수 있다.



이 예제는 Kepler에서 새로운 셔플 명령을 이용하는 몇 가지 방법을 보여 준다.

원자적 연산

원자적 메모리 연산은 동시 스레드가 공유 데이터 구조에 대한 읽기-수정-쓰기 작업을 올바르게 수행할 수 있도록 하는 중요한 병렬 프로그래밍 방식이다. add, min, max, compare-and-swap 등의 원자적 연산은 읽기, 수정, 쓰기 작업이 다른 스레드의 방해를 받지 않고 수행된다는 점에서 원자적이라고 할 수 있다. 원자적 메모리 연산은 잠금을 사용하여 스레드 실행을 직렬화하지 않는 병렬 정렬, 빼기 연산, 병렬 데이터 구조 구축에 널리 사용된다.

Kepler GK110의 전역 메모리 원자적 연산의 처리량은 Fermi 세대에 비해 크게 향상되었다. 일반적인 전역 메모리 주소로의 원자적 연산 처리량은 클럭당 1개 연산으로 9배 향상되었다. 독립적인 전역 주소로의 원자적 연산 처리량도 크게 향상되었고 주소 충돌을 처리하는 로직의 효율성이 개선되었다. 원자적 연산이 전역 로드 작업과 같은 속도로 처리되는 경우가 많다. 덕분에 커널 내부 루프에 사용할 수 있을 만큼 원자적 연산 속도가 빨라져 이전에 일부 알고리즘에서 결과를 합산하는 데 필요했던 감소 통과(Reduction Pass)를 배제할 수 있게 되었다. Kepler GK110은 전역 메모리에서의 64비트 원자적 연산에 대한 기본 지원도 확장해 준다. 따라서 GK110은 atomicAdd, atomicCAS 및 atomicExch(Fermi 및 Kepler GK104에서도 지원) 외에 다음도 지원한다.

- atomicMin
- atomicMax
- atomicAnd
- atomicOr
- atomicXor

기본적으로 지원되는 다른 원자적 연산(예: 64비트 부동 소수점 원자적 연산)도 CAS(Compare-and-Swap) 명령을 사용해 에뮬레이트할 수 있다.

텍스처 향상

GPU의 전용 하드웨어 텍스처 유닛은 이미지 데이터를 샘플링하거나 필터링해야 하는 컴퓨팅 프로그램에 있어서 중요한 리소스다. Kepler에서 텍스처 처리량은 Fermi에 비해 증가했다. 각 SMX 유닛에는 Fermi GF110 SM에 비해 4배 증가한 16개의 텍스트 필터링 유닛이 있다.

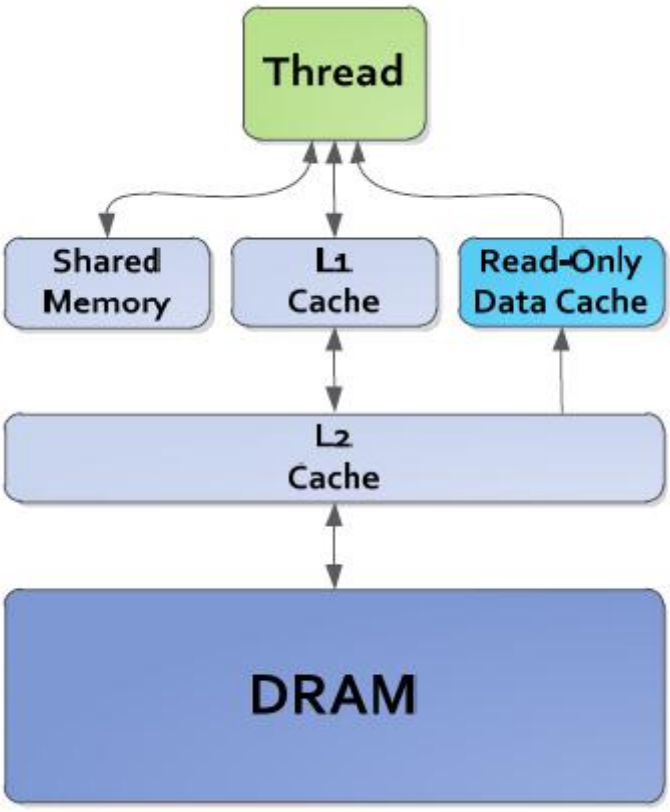
Kepler에서는 텍스처 상태 관리 방식도 바뀌었다. Fermi 세대에서는 GPU가 텍스처를 참조하려면 그리드 실행 전에 고정 크기 바인딩 테이블에서 "슬롯"을 할당받아야 했다. 해당 테이블의 슬롯 수는 궁극적으로 프로그램이 런타임에 읽을 수 있는 고유 텍스처 수를 제한한다. 결국 프로그램이 Fermi에서 동시에 액세스할 수 있는 텍스처는 128로 제한된다.

Kepler에서는 바인딩되지 않은 텍스처를 사용하므로 슬롯을 사용한 추가 단계가 필요 없다. 즉, 텍스처 상태가 메모리에 객체로 저장되고 하드웨어가 필요 시에 이러한 상태 객체를 불러오기 때문에 바인딩 테이블이 불필요하다. 따라서 컴퓨팅 프로그램에서 참조할 수 있는 고유 텍스처 수의 제약이 효과적으로 해소된다. 대신 프로그램에서 언제든지 텍스처를 매핑하고 다른 포인터와 마찬가지로 텍스처 핸들을 전달할 수 있다.

Kepler 메모리 서브시스템 – L1, L2, ECC

Kepler의 메모리 계층 구조는 Fermi와 유사하게 구성된다. Kepler 아키텍처는 SMX 멀티프로세서마다 L1 캐시를 하나씩 사용하여 로드 및 저장 작업을 위한 통합 메모리 요청 경로를 지원한다. 또한 Kepler GK110은 아래 설명과 같이 읽기 전용 데이터에 새로운 추가 캐시를 컴파일러 지향 방식으로 사용할 수 있게 해 준다.

Kepler 메모리 계층 구조



64 KB 구성 가능 공유 메모리 및 L1 캐시

이전 세대 Fermi 아키텍처와 마찬가지로 Kepler GK110 아키텍처에도 각 SMX마다 64KB의 온칩 메모리가 사용된다. 이 메모리는 공유 메모리 48KB와 L1 캐시 16KB, 또는 공유 메모리 16KB와 L1 캐시 48KB로 구성할 수 있다. Kepler에서는 공유 메모리와 L1 캐시를 32KB/32KB으로 분할할 수 있어 보다 유연한 공유 메모리 및 L1 캐시 할당 구성이 가능해졌다. 각 SMX 유닛의 처리량을 높이기 위해 64b 이상의 로드 작업에 사용되는 공유 메모리 대역폭도 Fermi SM의 2배인 코어 클럭당 256B로 증가했다.

48KB 읽기 전용 데이터 캐시

Kepler에는 L1 캐시 외에 기능이 실행되는 동안 읽기 전용으로 인식되는 데이터를 처리하는 데 사용되는 48KB 캐시가 새롭게 도입되었다. Fermi 세대에서는 텍스처 유닛에서만 이러한 캐시에 액세스할 수 있었다. 전문 프로그래머들은 데이터를 텍스처로 매핑함으로써 이 경로를 통해 데이터를 명시적으로 로드할 경우 많은 이점이 있다는 사실을 알고 있지만 이러한 방식은 제약이 많았다.

Kepler에서는 텍스처 처리 성능이 높아지면서 이 캐시의 용량이 대폭 증가했을 뿐만 아니라 SM에서 일반 로드 작업 시에 이 캐시에 직접 액세스할 수 있게 했다. 읽기 전용 경로를 사용하면 로드 및 작업 세트가 차지하는 공간이 공유/L1 캐시 경로에서 배제되기 때문에 이점이 있다. 또한 읽기 전용 데이터 캐시의 높은 태그 대역폭은 여러 가지 시나리오 중에서도 특히 최고 속도의 비정렬 메모리 액세스 패턴을 지원한다.

이 경로의 사용은 컴파일러에 의해 자동으로 관리된다. 프로그래머가 C99 표준 "const __restrict" 키워드를 사용하여 상수로 알려진 변수 또는 데이터 구조에 대한 액세스는 컴파일러에 의해 읽기 전용 데이터 캐시를 통해 로드되도록 태깅된다.

향상된 L2 캐시

Kepler GK110 GPU에는 Fermi 아키텍처에 비해 용량이 2배 증가한 1536KB의 전용 L2 캐시 메모리가 사용된다. L2 캐시는 SMX 유닛 간의 주된 데이터 통합 지점으로, 모든 로드, 저장 및 텍스처 요청을 처리하고 GPU 전반에 걸쳐 효율적인 고속 데이터 공유 기능을 제공한다. Kepler의 L2 캐시는 Fermi보다 최고 2배 높은 클럭당 대역폭을 제공한다. 사전에 데이터 주소를 알 수 없는 피직스 솔버, 레이 트레이싱, 희소 행렬곱셈 등의 알고리즘은 캐시 계층 구조로 얻을 수 있는 이점이 더욱 크다. 여러 SM이 같은 데이터를 읽어야 하는 필터와 컨볼루션 커널에도 이점이 있다.

메모리 보호 지원

Kepler의 레지스터 파일, 공유 메모리, L1 캐시, L2 캐시, DRAM 메모리도 Fermi와 마찬가지로 SECDED(Single-Error Correct Double-Error Detect) ECC 코드로 보호된다. 또한 읽기 전용 데이터 캐시는 패리티 검사를 통한 단일 오류 수정을 지원한다. 패리티 오류가 발생한 경우 캐시 유닛이 오류가 발생한 줄의 유효성을 자동으로 검사하여 L2에서 강제로 올바른 데이터를 읽는다.

DRAM에서 전달되는 ECC 체크비트는 불가피하게 DRAM 대역폭을 일정 부분 소모하기 때문에 특히 메모리 대역폭을 많이 소모하는 애플리케이션에서 ECC 지원

작업과 ECC를 지원하지 않는 작업 간에 성능 차이가 발생하게 된다. Kepler GK110에서는 Fermi의 경험을 바탕으로 ECC 체크비트 가져오기 처리를 최적화하는 몇 가지 기능을 구현했다. 그 결과 ECC를 사용하는 경우와 그렇지 않은 경우의 성능 차이가 평균 66% 감소했다(내부 컴퓨팅 애플리케이션 시험조에서 측정).

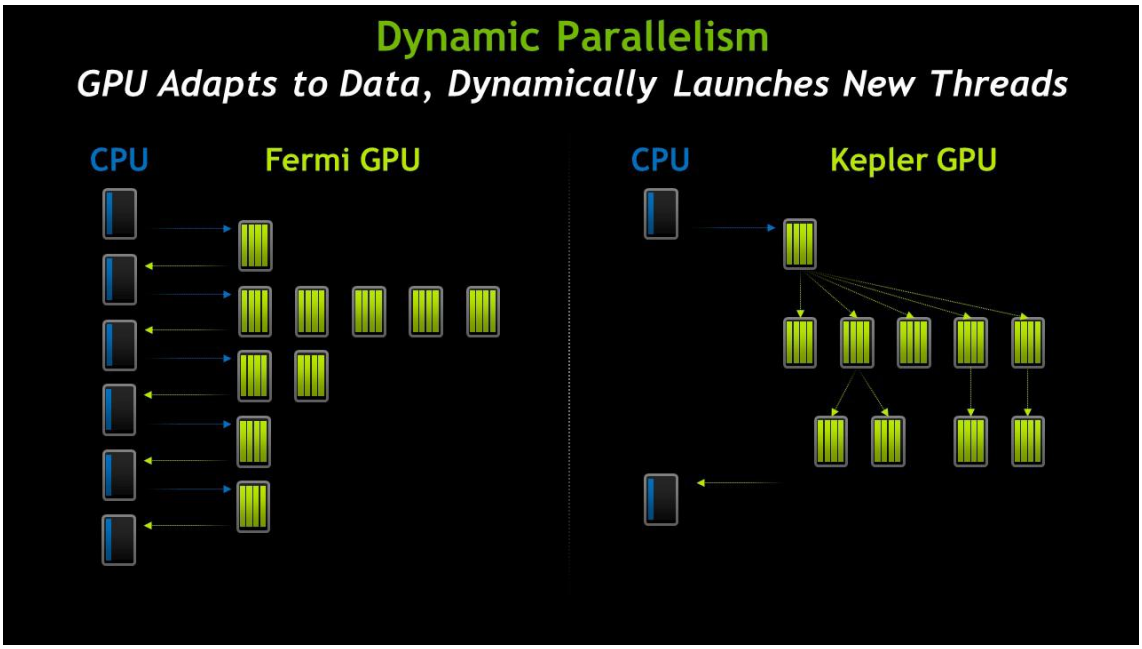
동적 병렬 처리(Dynamic Parallelism)

하이브리드 CPU-GPU 시스템에서 애플리케이션의 대용량 병렬 코드를 효율적인 방식으로 GPU에서 전적으로 실행하면 GPU의 와트당 성능이 높아져 확장성과 성능이 향상된다. 애플리케이션의 이러한 추가 병렬 처리 부분의 성능을 더욱 높이려면 GPU가 보다 다양한 유형의 병렬 워크로드를 지원해야 한다.

동적 병렬 처리는 Kepler GK110에서 새롭게 선보이는 기능으로, CPU와는 별개로 GPU에서 자체적으로 처리할 새로운 작업을 생성하고, 결과를 동기화하고, 가속화된 전용 하드웨어 경로를 통해 해당 작업의 스케줄링을 제어할 수 있게 해 준다.

Fermi는 커널 시작 시에 문제의 스케일과 파라미터가 알려져 있는 경우 대용량 병렬 데이터 구조를 처리하는 성능이 매우 좋다. 모든 작업은 호스트 CPU에서 시작되어 완료될 때까지 실행된 후 CPU로 결과가 다시 반환된다. 그러면 최종 해결 과정에 이 결과를 사용하거나, CPU에서 결과를 분석하여 다시 GPU로 추가 처리를 위한 요청을 보낸다

Kepler GK110에서는 모든 커널이 다른 커널을 실행할 수 있으며, 호스트 CPU와의 상호 작용 없이 추가 작업을 처리하는 데 필요한 스트림, 이벤트를 생성하고 종속성을 관리할 수 있다. 이러한 혁신적인 아키텍처는 개발자가 손쉽게 반복적이고 데이터 종속적인 실행 패턴을 생성 및 최적화할 수 있도록 하고, 더 많은 프로그램이 GPU에서 직접 실행될 수 있게 한다. 그러면 시스템 CPU의 여유 성능을 확보하여 다른 작업을 처리하는 데 이용하거나 성능이 낮은 CPU로 구성된 시스템에서 같은 양의 워크로드를 처리할 수 있다.



동적 병렬 처리는 애플리케이션이 CPU를 사용하는 것이 아니라(왼쪽 이미지) GPU에서 직접 더 많은 병렬 코드를 실행할 수 있게 한다(오른쪽 이미지).

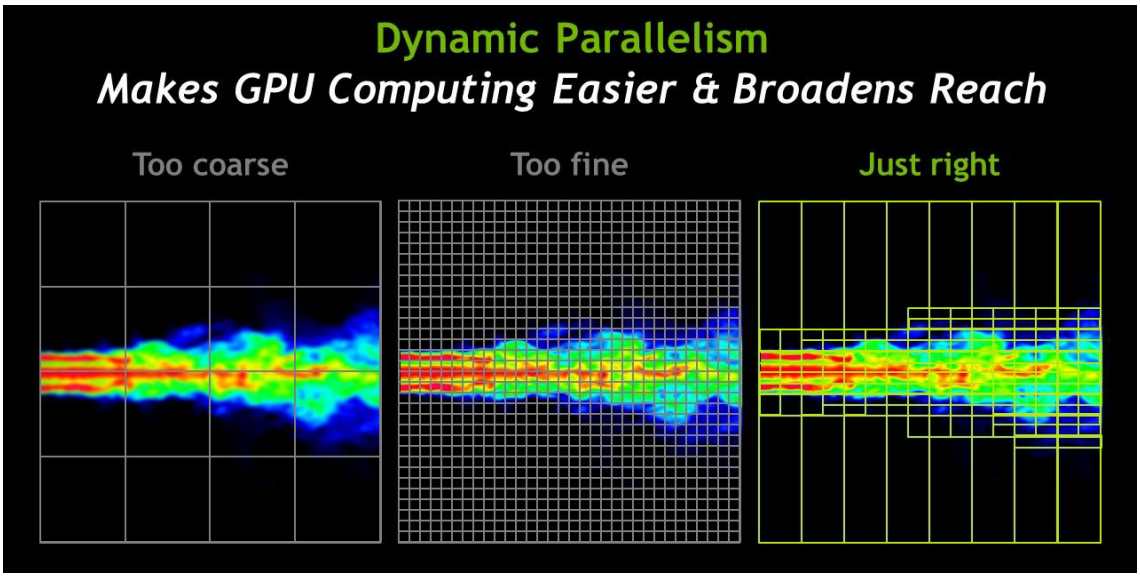
동적 병렬 처리는 병렬 처리 양이 각기 다른 중첩된 루프, 직렬 제어 작업 스레드의 병렬 팀, 애플리케이션의 병렬 처리 부분과의 데이터 지역성을 높이기 위해 GPU로 오프로드된 단순한 직렬 제어 코드 등 보다 다양한 병렬 알고리즘을 GPU에서 구현할 수 있도록 한다.

커널에는 GPU상의 중간 결과를 기준으로 추가 워크로드를 실행하는 기능이 있기 때문에 프로그래머가 작업을 지능적으로 로드 밸런싱하여 처리 성능이 가장 많이 요구되거나 해결에 있어서 가장 중요한 문제 영역에 대부분의 리소스를 집중할 수 있다.

일례로, 수치 시뮬레이션을 위한 동적 설정을 들 수 있다. 일반적으로 그리드 셀은 변화율이 가장 높아 데이터 전반에 걸친 사전 처리를 요하는 영역에 집중된다. 또는 일정하게 성긴 그리드를 사용하여 GPU 리소스의 낭비를 방지하거나, 일정하게 조밀한 그리드로 모든 특징이 캡처되도록 할 수 있지만, 이러한 옵션은 시뮬레이션 기능이 지원되지 않거나 중요하지 않은 영역에서 지나친 컴퓨팅 리소스 소모를 야기할 소지가 있다.

동적 병렬 처리 기능을 사용하면 런타임에 데이터 종속적인 방식으로 그리드 해상도를 동적으로 결정할 수 있다. 시뮬레이션을 성긴 그리드부터 시작해서 중요한 영역으로

"줌 인(Zoom in)"하면서 변화가 거의 없는 영역에서 불필요한 계산 작업이 발생하지 않게 한다. CPU에서 실행된 커널의 시퀀스를 사용하여 이를 구현할 수도 있지만 GPU에서 단일 시뮬레이션 커널의 일부로서 데이터를 분석하고 추가 작업을 실행하도록 하여 그리드 자체를 미세화하는 편이 CPU의 사용과 CPU 및 GPU 간의 데이터 전송을 배제할 수 있어 훨씬 간단하다.



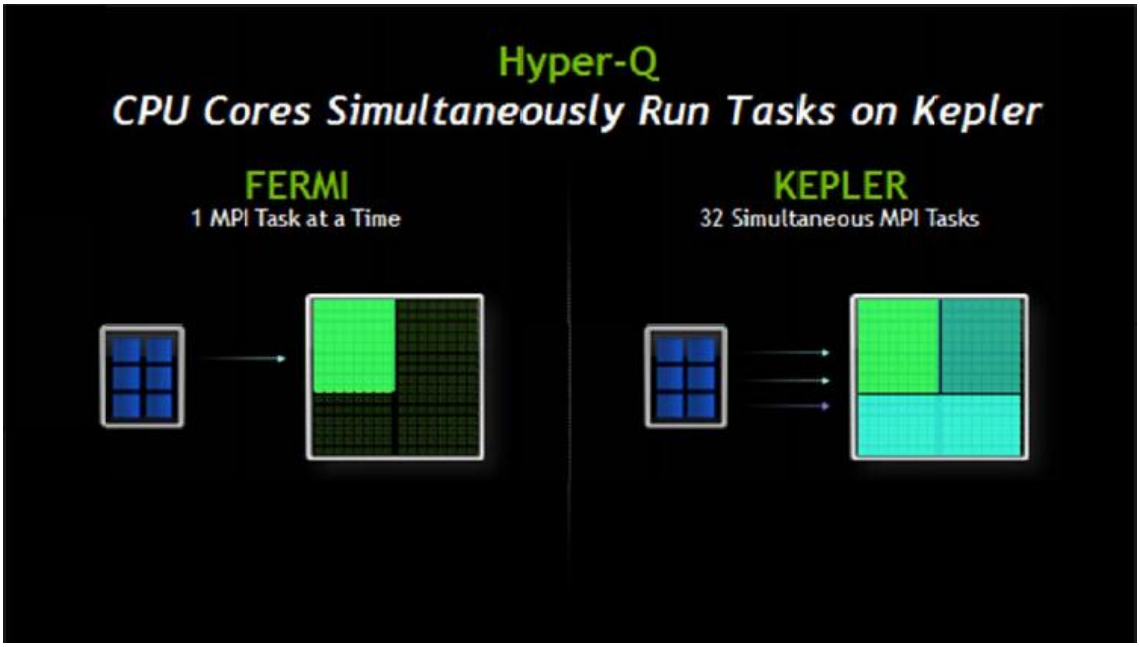
–이미지 제공: Charles Reid

위의 예는 수치 시뮬레이션에 동적으로 사이징되는 그리드를 사용하는 데 따른 이점을 보여 준다. 가장 높은 정밀도 요구 사항을 충족하려면 전체 시뮬레이션 영역에 걸쳐 매우 세밀한 해상도로 고정 해상도 시뮬레이션을 실행해야 하지만, 다중 해상도 그리드는 국소적 변화를 기준으로 각 영역에 적합한 시뮬레이션 해상도를 적용한다.

Hyper-Q

과거에는 최적으로 스케줄링된 여러 스트림의 워크로드를 계속 GPU에 제공하는 것이 문제였다. Fermi 아키텍처는 별도의 스트림에 전달된 16개의 커널을 동시에 실행하도록 지원했지만 결국 모든 스르림이 동일한 하드웨어 작업 대기열로 멀티플렉스되었다. 때문에 잘못된 스트림 내부 종속성이 발생하여 특정 스트림 내의 종속 커널이 다른 스트림의 추가 커널을 실행하기 전에 완료해야 하는 문제가 나타났다. 너비 우선(Breath-first) 실행 순서를 사용하면 문제를 어느 정도 완화할 수 있지만 프로그램의 복잡성이 높아지면서 이를 효율적으로 관리하기가 점점 더 어려워졌다.

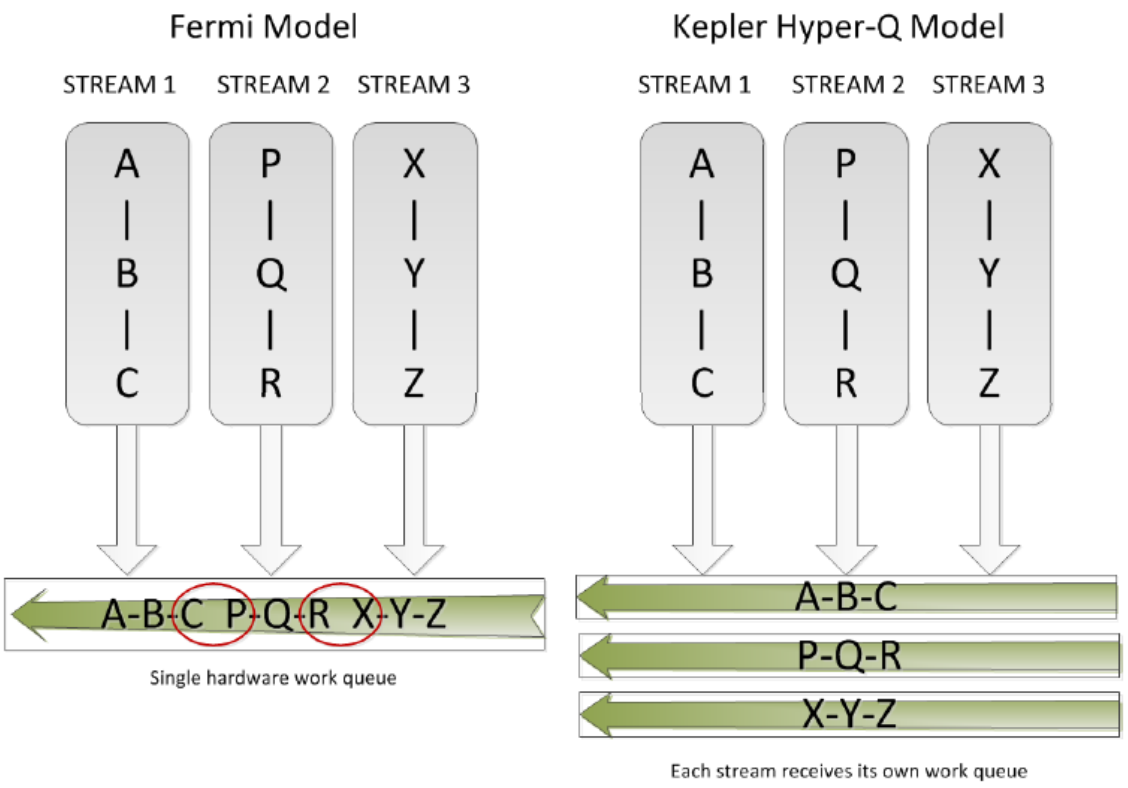
Kepler GK110에서는 새로운 Hyper-Q 기능으로 이러한 기능을 향상시켰다. Hyper-Q는 호스트와 GPU의 CWD(CUDA Work Distributor) 로직 간의 총 연결(작업 대기열) 수를 늘려 주며, 하드웨어에서 관리되는 동시 연결 수를 32개(Fermi의 경우 단일 연결만 가능)까지 지원한다. Hyper-Q는 여러 CUDA 스트림, 여러 MPI(메시지 전달 인터페이스) 프로세스 또는 프로세스 내의 여러 스레드에서 연결할 수 있도록 하는 유연한 솔루션이다. 이전에 여러 작업에 걸쳐 직렬화 오류가 발생해 GPU 활용에 한계가 있었던 애플리케이션도 기존 코드를 변경하지 않고 성능이 32배까지 개선되는 효과를 볼 수 있다.



Hyper-Q는 CPU와 GPU 간의 동시 연결 수를 늘려 준다.

각 CUDA 스트림은 자체 하드웨어 작업 대기열에서 관리되며, 스트림 간 종속성이 최적화되고, 특정 스트림의 연산이 더 이상 다른 스트림을 차단하지 않기 때문에 잘못된 종속성이 발생할 가능성을 없애기 위해 실행 순서를 특별히 맞추지 않고도 여러 스트림을 동시에 실행할 수 있다.

Hyper-Q는 MPI 기반 병렬 컴퓨터 시스템에서 사용할 경우 막대한 이점을 제공한다. 기존의 MPI 기반 알고리즘은 멀티 코어 CPU 시스템에서 실행하도록 작성되는 경우가 많았다. 각 MPI 프로세스에 할당되는 작업량은 코어 수에 따라 조정된다. 이 경우 단일 MPI 프로세스에 GPU를 완전히 점유하기에 불충분한 작업량이 할당될 수 있다. 여러 MPI 프로세스가 GPU 하나를 공유할 수도 있지만, 이러한 프로세스에서 잘못된 종속성으로 인해 병목 현상이 발생할 수도 있다. Hyper-Q는 이러한 잘못된 종속성을 없애 MPI 프로세스 간에 공유되는 GPU의 효율성을 크게 높인다.



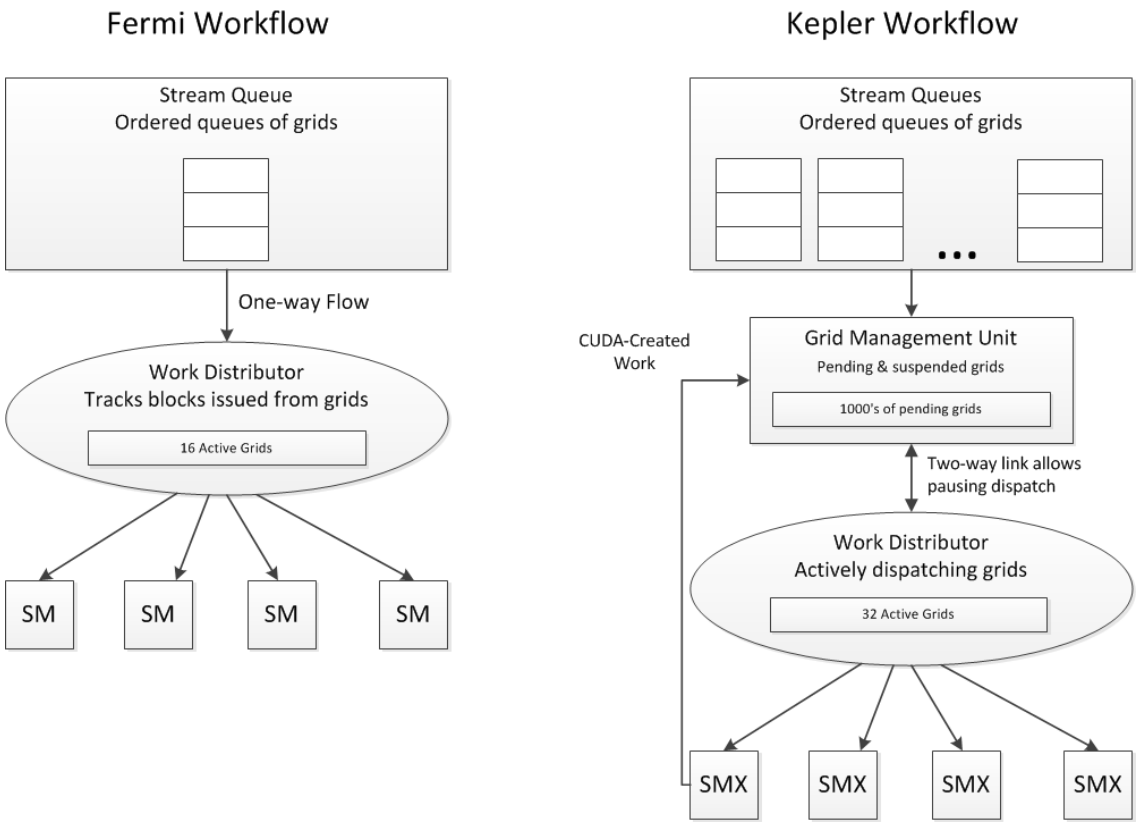
CUDA 스트림에 작동하는 Hyper-Q: 왼쪽의 Fermi 모델에서는 단일 하드웨어 작업 대기열로 인해 발생한 스트림 내부 종속성 때문에 (C,P)와 (R,X)만 동시에 실행할 수 있다. Kepler Hyper-Q 모델은 모든 스트림이 별도의 작업 대기열을 사용하여 동시에 실행될 수 있게 한다.

그리드 관리 유닛 - 효율적인 GPU 활용도 관리

동적 병렬 처리를 통해 CUDA 커널이 GPU에서 직접 작업을 시작하는 기능 등 Kepler GK110의 새로운 기능을 지원하려면 Kepler에서 CPU-GPU 간 워크플로우의 기능성이 Fermi보다 뛰어나야 했다. Fermi에서는 스레드 블록의 그리드가 CPU에 의해 실행되고 항상 완료될 때까지 실행되기 때문에 CWD(CUDA Work Distributor) 유닛을 통해 호스트에서 SM으로 이어지는 단순한 단방향의 작업 흐름이 만들어진다. Kepler GK110은 GPU가 CPU와 CUDA에서 생성된 워크로드를 모두 효율적으로 관리할 수 있도록 하여 CPU-GPU 간 워크플로우를 향상시키도록 설계되었다.

커널이 GPU에서 직접 작업을 실행하도록 하는 Kepler GK110 GPU의 기능을 앞에서 설명했는데, 이러한 새로운 기능을 지원하기 위해 Kepler GK110 아키텍처가 어떻게 변경되었는지를 이해하는 것이 중요하다. Kepler의 경우 Fermi와 마찬가지로 CPU에서 그리드를 실행할 수 있지만, Kepler SMX 유닛 내에서 CUDA에 의해 프로그래밍 방식으로 새 그리드가 생성될 수도 있다. 이처럼 CUDA에서 생성된 그리드와 호스트에서 생성된 그리드를 모두 관리하기 위해 Kepler GK110에는 새로운 GMU(그리드 관리 유닛)가 사용되었다. 이 제어 유닛은 실행을 위해 SMX 유닛으로 보내질 CWD에 전달된 그리드를 관리하고 우선 순위를 지정한다.

Kepler의 CWD는 디스패치할 준비가 된 그리드를 홀드하며, Fermi CWD의 용량보다 2배 많은 32개의 활성 그리드를 디스패치할 수 있다. Kepler CWD는 양방향 링크를 통해 GMU와 통신한다. 이 양방향 링크는 GMU가 새로운 그리드의 디스패치를 일시 중지하고, 일시 중단되어 대기 중인 그리드를 필요할 때까지 홀드할 수 있게 해 준다. GMU에는 Kepler SMX 유닛에 대한 직접 연결도 있는데, 이 연결은 동적 병렬 처리를 통해 GPU에서 추가 작업을 실행하는 그리드가 우선 순위를 지정하고 디스패치할 새로운 작업을 다시 GMU로 보내는 데 사용된다. 추가 워크로드를 디스패치한 커널이 일시 중지되면 종속 작업이 완료될 때까지 GMU가 해당 커널을 비활성 상태로 홀드한다.



새롭게 설계된 Kepler 호스트-GPU 간 워크플로우는 디스패치 중인 그리드를 관리하고, 디스패치를 일시 중지하고, 일시 중단되어 대기 중인 그리드를 홀드할 수 있도록 하는 새로운 그리드 관리 유닛을 보여 준다.

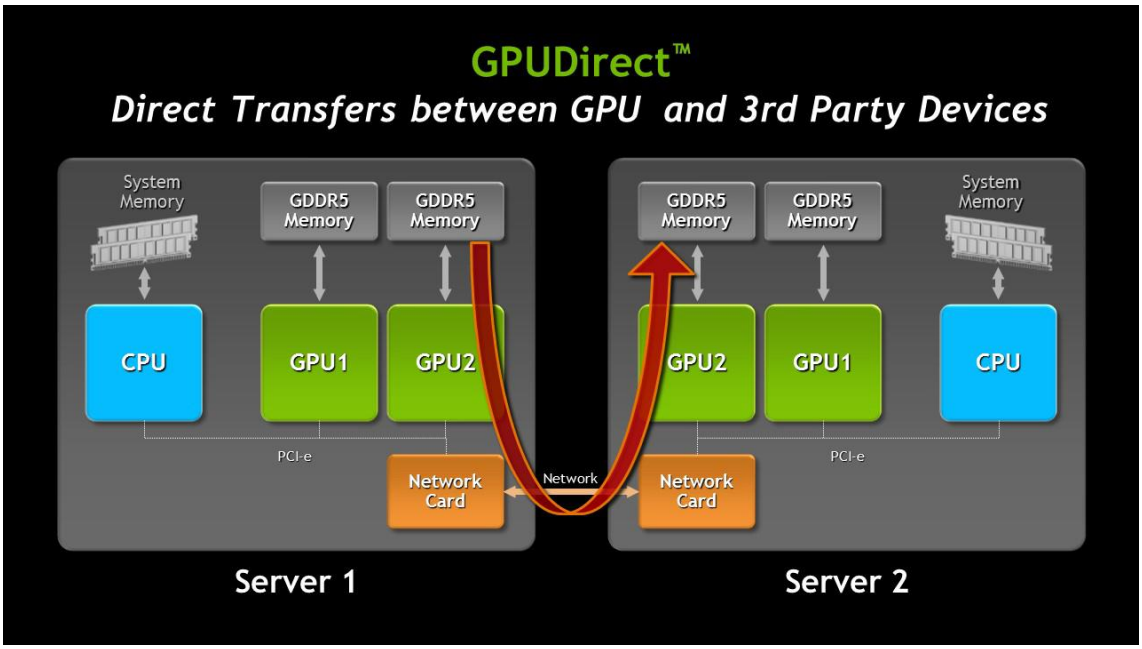
NVIDIA GPUDirect™

대용량의 데이터를 처리할 때는 데이터 처리량을 늘리고 지연을 줄이는 것이 컴퓨팅 성능을 높이는 데 있어서 가장 중요하다. Kepler GK110은 NVIDIA GPUDirect의 RDMA 기능을 지원한다. GPUDirect는 IB 어댑터, NIC, SSD와 같은 타사 디바이스가 GPU 메모리에 직접 액세스할 수 있게 하여 성능을 높이도록 설계되었다. CUDA 5.0을 사용할 때 GPUDirect는 다음과 같은 중요한 기능을 제공한다.

- CPU 쪽 데이터 버퍼링이 없는 NIC와 GPU 간의 DMA(직접 메모리 액세스)
- GPU와 네트워크의 기타 노드 간의 대폭 향상된 MPISend/MPIRecv 효율성
- CPU 대역폭 및 지연 병목 현상 해소
- 다양한 타사 네트워크, 캡처 및 저장 디바이스와 호환

예약 시간 마이그레이션(오일 및 가스 탐사를 위한 탄성파 이미징에 사용) 등의 애플리케이션은 여러 개의 GPU에 대용량 이미징 데이터를 배분한다. 데이터를 처리하려면 수백 개의 GPU가 공동 작업을 실행해야 하고 중간 결과를 주고받는 경우도 많다. GPUDirect는 P2P 및 RDMA 기능을 기반으로 서버 내 또는 서버 간의 이러한 GPU-GPU 간 통신 시나리오를 위해 훨씬 높은 전체 대역폭을 지원한다.

Kepler GK110은 P2P, 비디오용 GPUDirect 기능 등 다른 GPUDirect 기능도 지원한다.



GPUDirect RDMA는 네트워크 어댑터와 같은 타사 디바이스에서 GPU 메모리에 직접 액세스할 수 있게 한다. 즉, 서로 다른 노드의 GPU 간에 직접 전송이 가능한 것이다.

결론

2010년, Fermi의 출시와 함께 NVIDIA는 CPU와 GPU가 함께 작동하면서 컴퓨팅 성능 집약적인 워크로드를 처리하는 하이브리드 컴퓨팅 모델을 기반으로 HPC(고성능 컴퓨팅) 산업의 새로운 시대를 열었다. NVIDIA는 이제 새로운 Kepler GK110 GPU로 HPC 산업의 수준을 한 차원 더 높이려 한다.

Kepler GK110은 컴퓨팅 성능과 처리량을 극대화하면서 뛰어난 전력 효율성이 보장되도록 완전히 새롭게 설계되었다. SMX, 동적 병렬 처리, Hyper-Q 등 하이브리드 컴퓨팅의 속도, 프로그래밍의 용이성, 폭넓은 애플리케이션에 적용되는 범용성을 크게 높이는 혁신적인 기능들이 아키텍처에 많이 적용되었다. Kepler GK110 GPU는 앞으로 워크스테이션에서 슈퍼컴퓨터에 이르기까지 수많은 시스템에 사용되어 HPC 분야의 난제들을 해결해나갈 것이다.

부록 A - CUDA 요약 정보

CUDA는 NVIDIA GPU에서 C, C++, Fortran 및 기타 언어로 작성된 프로그램을 실행할 수 있게 하는 하드웨어/소프트웨어 통합 플랫폼이다. CUDA 프로그램은 여러 병렬 스레드에 걸쳐 실행되는 커널이라는 병렬 함수를 호출한다. 그림 1과 같이 프로그래머 또는 컴파일러는 이러한 스레드를 스레드 블록, 그리고 스레드 블록의 그리드로 구성한다. 스레드 블록의 각 스레드는 커널 인스턴스를 실행한다. 또한 각 스레드에는 스레드 블록 및 그리드 내의 스레드 및 블록 ID, 프로그램 카운터, 레지스터, 스레드별 전용 메모리, 입력 및 출력 결과가 들어 있다.

스레드 블록은 배리어 동기화와 공유 메모리를 통해 서로 간에 공동 작업이 가능한 동시에 실행 중인 스레드의 집합이다. 스레드 블록의 그리드 내에는 블록 ID가 들어 있다. 그리드는 동일한 커널을 실행하고, 전역 메모리에서 입력을 읽고, 결과를 전역 메모리에 쓰고, 종속 커널 호출 간에 동기화하는 스레드 블록의 집합체이다. CUDA 병렬 프로그래밍 모델에서 각 스레드에는 스레드별 전용 메모리 공간이 할당된다. 이 메모리 공간은 레지스터 스필링, 함수 호출 및 C 자동 배열 변수에 사용된다. 각 스레드 블록에는 블록별 공유 메모리 공간이 할당되어 스레드 간 통신, 데이터 공유 및 병렬 알고리즘에서의 결과 공유에 사용된다. 스레드 블록의 그리드는 커널 차원의 전역 동기화 후 전역 메모리 공간의 결과를 공유한다.

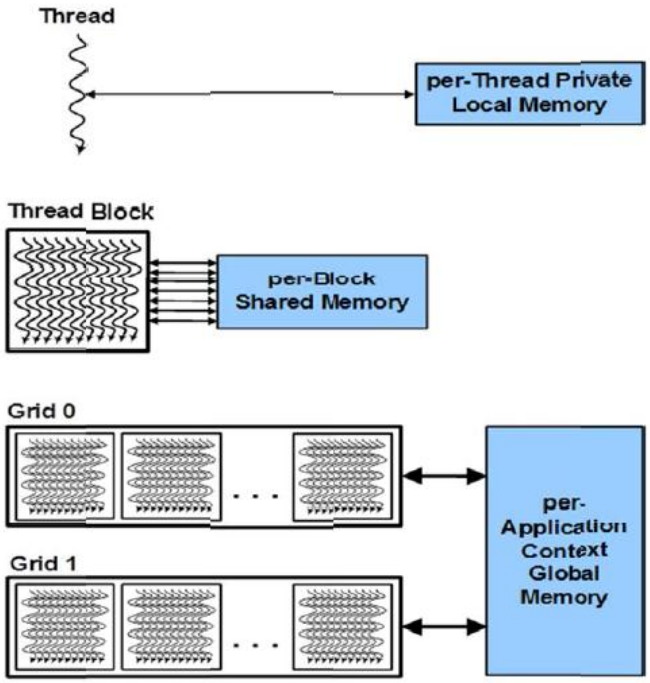


그림 1: 스레드, 블록 및 그리드, 그리고 각각에 해당되는 스레드별 전용 메모리 공간, 블록별 공유 메모리 공간, 애플리케이션별 전역 메모리 공간으로 구성된 CUDA 계층 구조

CUDA 하드웨어 실행

CUDA의 스레드 계층 구조는 GPU의 프로세서 계층 구조에 매핑된다. GPU는 하나 이상의 커널 그리드를 실행하고 스트리밍 멀티프로세서(Fermi의 SM/Kepler의 SMX)는 하나 이상의 스레드 블록을 실행한다. SMX의 CUDA 코어와 기타 실행 유닛은 스레드 명령을 실행한다. SMX는 32개의 스레드로 이루어진 와프라는 그룹 단위로 스레드를 실행한다. 일반적으로 프로그래머는 기능적 정확성을 위해 와프 실행을 무시하고 개별 스칼라 스레드에 집중할 수도 있지만, 와프에 속한 스레드가 모두 동일한 코드 경로를 실행하고 인접한 주소를 사용해 메모리에 액세스하도록 하면 성능을 크게 향상시킬 수 있다.

법적 고지