KISM Smart Media Journal
Vol. 4, No. 4 pp. 56~63 ISSN:2287-1322

작업 처리 단위 변화에 따른 GPU 성능과 메모리 접근 시간의 관계 분석

Analysis of GPU Performance and Memory Efficiency according to Task Processing Units

손동오*, 심규연*, 김철홍**
(Dong Oh Son, Gyu Yeon Sim, Cheol Hong Kim)

요 약

최신 GPU는 프로세서 내부에 포함된 다수의 코어를 활용하여 높은 병렬처리가 가능하다. GPU의 높은 병렬성을 활용하는 기법 중 하나인 GPGPU 구조는 GPU에서 대부분의 CPU의 작업을 처리가 가능하게 해주며, GPU의 높은 병렬성과 하드웨어 자원을 효과적으로 활용할 수 있다. 본 논문에서는 다양한 벤치마크 프로그램을 활용하여 CTA(Cooperative Thread Array) 할당 개수 변화에 따른 메모리 효율성과 성능을 분석하고자 한다. 실험결과, CTA 할당 개수 증가에 따라 다수의 벤치마크 프로그램에서 성능이 향상되었지만, 일부 벤치마크 프로그램에서는 CTA 할당 개수 증가에 따른 성능 향상이 발생하지 않았다. 이러한 이유로는 벤치마크 프로그램에서 생성된 CTA 개수가 적거나 동시에 수행할 수 있는 CTA 개수가 정해져 있기 때문으로 판단된다. 또한, 각 벤치마크 프로그램별로 메모리 채널 정체에 따른 메모리 스톨, 내부연결망 정체에 따른 메모리 스톨, 파이프라인의 메모리 단계에서 발생하는 스톨을 분석하여 성능과의 연관성을 파악하였다. 본 연구의 분석결과는 GPGPU 구조의 병렬성 및 메모리 효율성 향상을 위한 연구에 대한 정보로 활용될 것으로 기대된다.

■ 중심어 : 그래픽 처리장치 병렬컴퓨팅; 메모리; 성능; CTA 스케쥴링 기법

Abstract

Modern GPU can execute mass parallel computation by exploiting many GPU core. GPGPU architecture, which is one of approaches exploiting outstanding computational resources on GPU, executes general-purpose applications as well as graphics applications, effectively. In this paper, we investigate the impact of memory-efficiency and performance according to number of CTAs(Cooperative Thread Array) on a SM(Streaming Multiprocessors), since the analysis of relation between number of CTA on a SM and them provides inspiration for researchers who study the GPU to improve the performance. Our simulation results show that almost benchmarks increasing the number of CTAs on a SM improve the performance. On the other hand, some benchmarks cannot provide performance improvement. This is because the number of CTAs generated from same kernel is a little or the number of CTAs executed simultaneously is not enough. To precisely classify the analysis of performance according to number of CTA on a SM, we also analyze the relations between performance and memory stall, dram stall due to the interconnect congestion, pipeline stall at the memory stage. We expect that our analysis results help the study to improve the parallelism and memory-efficiency on GPGPU architecture.

■ keywords : General Purpose computation on the Graphics Processing Unit; Memory; Performance; Cooperative Thread Array Scheduling Schemes

Ⅰ. 서 론

최근 컴퓨팅 시스템의 성능을 향상시키기 위해서 다양한 방법들이 연구되고 있다. 컴퓨팅 시스템 성능을

* 학생회원, 전남대학교 전자컴퓨터공학부
** 정회원, 전남대학교 전자컴퓨터공학부
본 논문은 중소기업청에서 지원하는 2015년도 산학연협력 기술개발사업(No.C0298874)과 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2015R1D1A3A01019454)

접수일자 : 2015년 11월 27일
수정일자 : 2015년 12월 29일
게재확정일 : 2015년 12월 30일
교신저자 : 김철홍 e-mail : chkim22@jnu.ac.kr

향상시키기 위한 연구로 마이크로프로세서뿐만 아니라 네트워크 구조[1] 변경이나 저장장치[2]에 대한 연구도 진행되어 왔다.

발생되는 명령어를 처리하는 마이크로프로세서는 코어(Core), 레지스터(Register), 프로그램 카운터(PC: Program Counter), 산술 논리 장치(ALU: Arithmetic/Logic Unit), 캐쉬(Cache) 등 다양한 장치를 포함하고 있으며 컴퓨팅 시스템의 핵심이다. 하나의 코어만을 사용하는 싱글코어 프로세서와 비교하여 멀티코어 프로세서는 듀얼코어, 쿼드코어, 헥사코어 등 여러 개의 코어를 활용하여 연산을 수행함으로 매우 높은 성능을 보인다[3-4]. 하지만, 멀티코어 프로세서의 성능을 향상시키기 위해서 코어의 집적개수를 증가시키거나 주파수를 증가시키는 방법은 높은 전력 소비와 온도 문제를 야기 시킨다[5]. 또한, 범용 연산을 처리하기 위해서 개발된 멀티코어 프로세서는 다수의 코어를 사용함에도 불구하고 다수의 범용 명령어를 동시에 처리하기 때문에 병렬성을 효과적으로 활용할 수 없다[6]. 이러한 멀티코어 프로세서의 문제점들을 해결하기 위해서 다양한 연구가 진행되고 있다. 최근 컴퓨팅 시스템의 성능을 향상시키기 위한 방법 중 하나로 CPU에서 처리하는 작업을 GPU로 이관시켜 처리하는 그래픽 처리장치 병렬 컴퓨팅(GPGPU: General Purpose computation on the Graphics Processing Unit)은 많은 연구자들에게 주목 받고 있다[7-11]. 기존의 GPU는 그래픽 작업을 효과적으로 처리하기 위해서 개발된 별도의 프로세서로 그래픽 연산만을 수행하였다. 하지만, GPGPU는 GPU에서 처리하던 그래픽 연산뿐만 아니라 CPU에서 처리하는 범용 연산까지 처리가 가능하다. GPU에 집적된 다수의 코어들은 수천 개의 스레드들을 병렬연산을 통해 효과적으로 수행할 수 있으며, CPU의 작업을 이관 받아 수행하여 컴퓨팅 시스템의 성능을 크게 증가 시킬 수 있다.

컴퓨팅 시스템의 성능을 향상시키기 위해서는 GPGPU의 가장 작은 작업 처리 단위인 스레드를 병렬적으로 수행해야한다. 스레드를 병렬적으로 수행하기 위해서 스레드는 워프, CTA, 커널, 애플리케이션과 같이 계층적으로 집단화 한다. 이러한 스레드의 집합 중 하나인 CTA(Cooperative Thread Array)는 GPU 코어에 할당되어 작업을 처리한다. CTA는 스케줄링 기법을 통해서 각 GPU 코어에 한 개 부터 여러 개가 할당될 수 있으며, 본 논문에서는 각 GPU 코어에 할당될 수 있는 최대 CTA 개수는 8개이다. 본 논문에서는 GPU의 각 코어에 할당되는 CTA의 개수가 시스템에 미치는 영향을 분석하기 위하여 다양한 벤치마크 프로그램을 활용하여 각 코어에 할당하는 최대 CTA의 개수에 변화를 주어 성능에 미치는 영향을 분석하고자 한다. 또한, CTA 할당 개수 변화에 따른 성능과 메모리 접근 시간의 연관성을 분석하기 위해서 다양한 메모리 스톨에 대해서 추가적으로 분석하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 GPGPU 구조와 작업이 할당되는 애플리케이션 계층구조에 대해서 설명하고 3장에서 실험환경과 실험결과 및 분석 내용에 대해서 자세히 기술한다. 마지막 4장에서 본 논문을 마무리한다.

Ⅱ. GPGPU 구조

이 장에서는 실험에서 사용된 GPGPU 구조와 벤치마크 프로그램을 수행할 때 사용되는 애플리케이션의 계층구조에 대해서 설명한다.

기존의 GPU는 CPU에서 발생하는 그래픽 작업을 보다 효과적으로 처리하기 위해서 개발되었다. 하지만 최신 GPU는 기존의 그래픽 연산뿐만 아니라 CPU에서 처리는 범용 연산까지도 수행이 가능하다. GPU는 하나의 프로세서에 수십 개 이상의 코어를 포함하고 있으며, 이를 활용하여 병렬연산을 효과적으로 수행할 수 있다. 다수의 코어를 병렬적으로 수행하여 수백 TFLOPS의 연산을 처리할 수 있으며, 이는 CPU의 연산 처리량보다 높다. 실험에서 사용하는 GPGPU는 NVIDIA사의 Fermi 구조를 대상으로 한다. 해당 구조는 다수의 코어를 프로세서 내부에 포함하고 있다. GPU 내부의 코어는 스트리밍 멀티프로세서(SM: Streaming Multiprocessors)라고 부르며, 본 논문에서는 SM으로 표기한다. SM 내부에는 병렬연산을 수행하는 다수의 스트리밍 프로세서(SP: Streaming Processor), 워프(Warp)들을 할당하는 워프 스케줄러(Warp Scheduler)와 디스패치(Dispatch)가 있으며, 레지스터 파일(Register File), L1 캐쉬(L1 Cache)와 공유 메모리(Shared Memory) 등으로 구성되어 있다.

GPGPU 구조는 SIMD(Single Instruction Multiple Data) 방식을 사용하여 하나의 명령어로 다수의 연산을 수행한다. 따라서 각 SM은 하나의 명령어를 통해 다수의 스레드를 동시에 수행하여 병렬성을 확보한다. 기본적으로 32개의 스레드들을 동시에 수행하며 이들의 집합을 워프(Warp)라고 한다. 또한, 다수의 워프들을 집단화하여 CTA(Cooperative Thread Array)라고 부른다. CTA는 스레드 블록과 같은 의미로 사용되며, 각 CTA는

하나의 커널(Kernel)에 집합된다[12]. 커널은 GPGPU에서 수행되는 가장 큰 단위이며, 애플리케이션은 최소 1개에서 여러 개의 커널로 구성되어 있다. 워프와 CTA 둘 다 스레드들의 집합이지만 작업의 처리는 주로 워프 단위에서 명령어를 수행하게 되고, CTA는 이러한 워프들의 집합으로 SM에 할당되어 CTA 내에 존재하는 워프들을 할당하여 명령어를 수행하게 된다. 따라서 본 논문에서는 SM에 직접적으로 할당되는 CTA에 초점을 맞춰 연구를 진행 하고자한다.

GPGPU에 작업을 할당하기 위해서 애플리케이션의 커널이 GPGPU에 할당된다. 각 커널에 존재하는 CTA와 워프는 CTA 스케줄러와 워프 스케줄러가 사용하여 라운드 로빈 스케줄링(Round-Robin Scheduling) 기법을 이용하여 순차적으로 SM에 할당된다. 각 SM에 할당되는 CTA의 최대 개수는 하드웨어 자원 계산에 의해 정해지게 되며, 본 실험에서 사용된 SM의 최대 CTA 할당 개수는 8개이다. CTA가 SM에 할당되게 되면, 각 SM은 CTA 내부의 스레드들을 통해 연산을 수행하며, 메모리 요청, SM 스톨 등에 의해 연산이 수행되지 않을 경우 대기시간 동안 다른 CTA를 수행하게 된다. 따라서 할당 가능한 CTA 개수가 증가되게 되면, 메모리 요청, SM 스톨에 상관없이 지속적으로 연산을 수행할 수 있고, 성능을 향상시킬 수 있다.

Ⅲ. 실험 분석

1. 실험환경

표 1. 하드웨어 구성변수

실험 인자	값
Number of SM	14
Warp Size(SIMD Width)	32
Number of Threads/SM	1024
Shared Memory/SM	16KB
Constant Cache/SM	8KB, 2-way 64byte lines, Read-only
Texture Cache/SM	12KB, 24-way 128byte lines, Read-only
L1 Data Cache	16KB, 4-way, 128byte lines
Unified L2 Cache	64KB, 8-way, 128byte lines
Clock (Core: Interconnection: DRAM)	575MHz: 575MHz: 750MHz
Number of Memory Controller	6
Number of Memory Chip/Controller	2
Memory Channel Bandwidth	4 bytes
CTA&Warp Scheduler (Scheduling Scheme)	Two-level scheduler (Round-Robin)

본 논문에서는 각 SM에 CTA를 할당하기 위해서 GPU 성능 평가 시뮬레이터인 GPGPU-SIM[13]과 전력 측정 시뮬레이터인 McPAT[14]을 통합한 GPUWATTCH를 사용하여 GPU를 모델링하였다. GPU 모델링 대상으로는 NVIDIA사의 Fermi 구조 중 TeslaC2050을 모델링 하여 실험을 진행하였다[15]. 모델링에 사용된 GPU는 기본적으로 14개의 SM을 포함하고 있으며, 각 SM에 8개의 CTA를 할당하도록 구성되어 있다. 표 1은 실험에서 사용된 GPGPU의 하드웨어 구성변수를 보여준다.

표 2. 벤치마크 프로그램

벤치마크 프로그램	약자	설명
1D Discrete Haar Wavelet Decomposition	DHWD [16]	1차원 신호 이산 Haar Wavelet 분해 연산
Vector Addition	VA [16]	두 벡터의 합 연산
K-Means	KM [17]	데이터 마이닝에서 사용되는 클러스터링 알고리즘
Needleman-Wunsch	NW [17]	DNA 서열 정렬을 위한 글로벌 비선형 최적화 방법
Gaussian Elimination	GE [17]	가우시안 소거법을 이용한 행렬 연산
Graph Algorithm: Breadth-First Search	BFS [13]	그래프 폭 우선 검색 알고리즘
Neural Network	NN [13]	숫자 인식 신경 네트워크
Weather	WP [13]	날씨 예측 모델링

본 논문에서 사용한 벤치마크 프로그램은 NVIDIA S

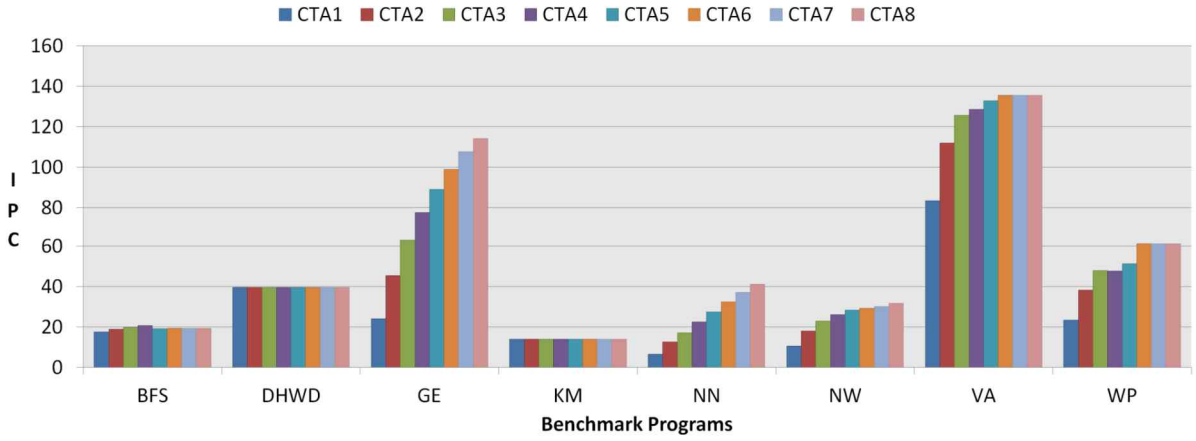

그림 1. CTA 할당 개수 변화에 따른 벤치마크 프로그램별 성능

DK 벤치마크 프로그램[16], Rodinia 벤치마크 프로그램[17], ISPASS 벤치마크 프로그램[13]을 사용하여 다양한 벤치마크 프로그램 수행에 따른 성능을 자세히 분석하였다. 3종류의 벤치마크 프로그램을 사용하였고, 그중에서 8개의 벤치마크 프로그램을 선택하여 실험에 사용하였다. 수행된 벤치마크 프로그램은 표 2에서 확인할 수 있다. 표 2는 각 벤치마크 프로그램의 이름과 약자, 그리고 해당 벤치마크 프로그램에 대한 설명을 나타낸다.

2. 실험결과

이 장에서는 CTA 할당 개수에 따른 GPGPU 구조의 성능과 메모리 스톨에 대해서 자세히 분석한다. 그림 1은 다양한 벤치마크 프로그램을 활용하여 CTA 할당 개수 변화에 따른 성능을 보여주고 있다. 세로축은 성능 값을 나타내며, 본 논문에서는 성능 값에 대한 지표로 IPC(Instructions Per Cycle)를 사용한다. 또한, 해당 IPC는 GPU 시스템의 성능을 나타낸다. 가로축은 본 실험에서 사용된 벤치마크 프로그램의 약자를 나타낸다. 그림을 보면 다수의 벤치마크 프로그램(GE, NN, NW, VA, WP)에서 CTA 할당 개수가 증가함에 따라 성능이 증가함을 확인할 수 있다. 이러한 결과는 예상이 가능한 수치이지만, DHWD, KM 벤치마크 프로그램에서는 CTA 할당 개수가 증가함에도 불구하고 성능의 변화가 없다. 또한, BFS 벤치마크 프로그램은 오히려 CTA 할당 개수가 증가함에도 불구하고 성능의 감소가 발생하였다.

DHWD, KM 벤치마크 프로그램에서 CTA 할당 개수 증가에 따라 성능의 변화가 없는 이유로는 해당 벤치마크 프로그램에서 동시에 수행할 수 있는 CTA 개수가 정해져 있거나, 생성된 CTA 개수가 적은 것으로 예상할 수 있다. 이와 같은 경우 GPU 코어에 실제로 할당되는 CTA 개수가 매우 적기 때문에 각 GPU 코어에 할당 가능한 최대 CTA 개수 증가와 상관없이 동일한 성능을 보여준다.

BFS 벤치마크 프로그램의 경우 CTA 할당 개수 증가에 따라 성능이 감소하였다. 이러한 이유로는 할당 가능한 최대 CTA 개수 증가로 인해 각 CTA에서 요구하는 메모리 요청에 따라 메모리 충돌로 인한 스톨이 발생하거나 파이프라인에 의한 SM 스톨, 하드웨어 자원 부족으로 인한 스톨이 발생한 것으로 예상된다.

그림 2는 정규화에 따른 CTA 할당 개수별 성능을 보여준다. 본 실험에서 사용된 CTA 할당 개수의 기본값인 CTA 8개를 기준으로 정규화하였다. 그림을 보면 대부분의 벤치마크 프로그램은 기본 CTA 할당 개수(CTA8)와 비교하여 낮은 성능을 보여주며, CTA 할당 개수가 증가함에 따라 성능이 꾸준히 증가함을 확인할 수 있다. BFS 벤치마크 프로그램을 제외한 모든 벤치마크 프로그램에서는 기본 CTA 할당 개수가 가장 성능이 높거나 같음을 확인할 수 있다. BFS 벤치마크처럼 특이점을 보이는 경우를 제외하고 대다수의 벤치마크 프로그램은 CTA 할당 개수 증가를 통해 성능 향상을 보인다. 이 결과를 통해 CTA 할당 개수가 증가하면 GPGPU 구조의 성능이 향상됨을 확인할 수 있다.

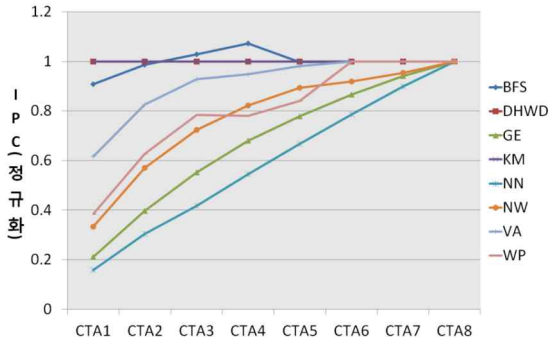

그림 2. CTA 할당 개수별 정규화 된 성능

그림 1에서 CTA 할당 개수 증가에 따른 성능을 분석하였다. 하지만, 정확한 실험결과에 따른 내용이 아닌 예측이므로 이를 증빙하기 위해서 각 벤치마크 프로그램별로 발생하는 메모리 스톨을 분석하여 CTA 할당 개수 변화가 성능에 미치는 영향을 자세히 분석하고자 한다. GPGPU 구조에서 메모리 스톨이 발생하는 요인으로는 메모리 채널 정체에 따른 메모리 스톨, 내부연결망 정체에 따른 메모리 스톨, 메모리 스테이지에서 발생하는 파이프라인 스톨 등이 있다.

각 벤치마크 프로그램별로 발생하는 메모리 스톨을 분석하여 CTA 할당 개수 변화에 따른 성능과의 연관성을 찾고자 한다.

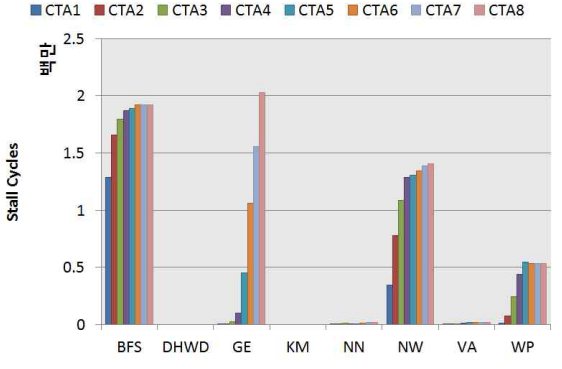

그림 3. 메모리 채널 정체에 따른 메모리 스톨 분석

그림 3은 메모리 채널 정체에 따른 메모리 스톨을 보여준다. 메모리 채널에서 데이터 요청이 많이 발생할 경우 메모리 스톨이 발생한다.

그림을 보면 DHWD, KM, NN, VA 벤치마크 프로그램은 CTA 할당 개수 변화에 따라 발생하는 메모리 스톨의 영향이 미미한 것을 알 수 있다. 이를 통해 해당 벤치마크 프로그램은 명령어 수행에 따라 발생하는 메모리 요청이 적음을 확인할 수 있다. 하지만, 이를 제외한 BFS, GE, NW, WP 벤치마크 프로그램은 CTA 할당 개수에 따라 스톨 사이클이 증가하는 것을 확인할 수 있다.

CTA 할당 개수 증가에 따라 메모리 스톨 사이클이 증가하는 이유로는 CTA 할당 개수가 증가함에 따라 CTA에서 발생하는 메모리 요청이 증가하기 때문이다. 하지만 CTA 할당 개수 증가에 따라 메모리 스톨 사이클이 증가함에도 불구하고 각 벤치마크 프로그램의 성능은 증가함을 확인할 수 있다.

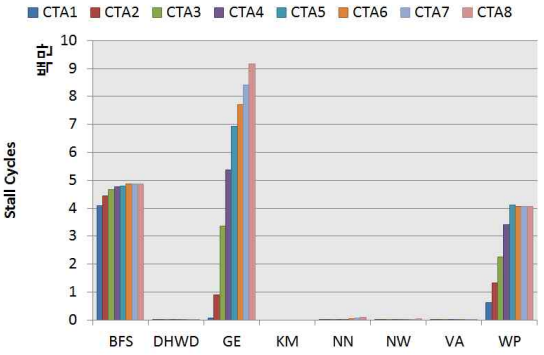

그림 4. 내부연결망 정체에 따른 메모리 스톨 분석

그림 4는 내부연결망 정체에 따른 메모리 스톨에 대한 그래프이다. 내부연결망에 많은 데이터가 요청되면 내부연결망은 정체가 발생하고 요청된 작업을 처리하기까지 메모리 스톨이 발생한다. BFS, GE, WP 벤치마크 프로그램에서 CTA 할당 개수 증가에 따라 메모리 스톨 사이클이 증가함을 확인할 수 있다.

CTA 할당 개수 증가에 따라 메모리 요청이 증가하여 메모리 스톨 또한 증가하지만, 메모리 스톨이 발생할 경우 해당 CTA는 메모리 요청이 끝날 때까지 대기하며, 다른 CTA 연산을 수행한다. 이처럼 특정 CTA의 스톨이 발생할 때 SM에 할당된 다른 CTA를 활성화하여 메모리 요청시간 동안 다른 CTA를 연산함에 따라 성능이 증가한다.

DHWD, NN, NW, VA 벤치마크 프로그램에서도 메모리 스톨이 발생하나 아주 적은 수의 메모리 스톨이 발생한다. 또한, KM 벤치마크 프로그램은 내부연결망 정체에 따른 메모리 스톨이 발생하지 않는다.

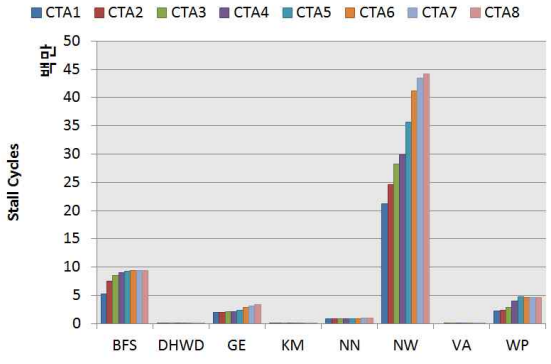

그림 5. 메모리 스테이지에서 발생하는 파이프라인 스톨 분석

마지막으로 그림 5는 파이프라인 중 메모리 스테이지에서 발생하는 스톨을 보여준다. 메모리 스테이지에서 메모리 요청이 발생할 경우 요청된 메모리가 전송될 때까지 파이프라인은 스톨 된다. NW 벤치마크 프로그램에서 가장 높은 스톨 사이클을 보이며, 기존의 결과처럼 BFS, GE, NW, WP 벤치마크 프로그램에서 스톨이 발생한다. DHWD, KM, NN, VA 벤치마크 프로그램에서 발생하는 메모리 스톨은 매우 적다.

실험결과, BFS, GE, NW, WP 벤치마크 프로그램에서 CTA 할당 개수 증가에 따라 메모리 스톨이 크게 증가하였고 BFS 벤치마크 프로그램을 제외한 나머지 벤치마크 프로그램들은 CTA 할당 개수 증가에 따라 성능 또한 증가하였다. 이를 통하여 CTA 할당 개수 증가는 다수의 CTA를 수행하기 위해서 더 많은 메모리 요청을 발생시키고 메모리 스톨을 증가시키지만 더 높은 성능을 보임을 알 수 있다. BFS 벤치마크 프로그램은 CTA 할당 개수 증가에 따라 메모리 스톨이 증가하였고, 성능은 감소하였다. GE, NW, WP 벤치마크 프로그램에서 메모리 스톨이 증가하였다. 또한, GE 벤치마크 프로그램은 메모리 채널 정체에 따른 메모리 스톨과 내부연결망 정체에 따른 메모리 스톨이 크게 증가하였다. 하지만, 메모리 스톨이 크게 증가하였음에도 불구하고, 성능은 증가하였기 때문에 BFS 벤치마크 프로그램의 성능 감소의 원인으로 메모리 스톨이라고 단정할 수는 없다. 앞서 말했듯이 메모리 스톨, 파이프라인에 의한 SM 스톨, 하드웨어 자원 부족으로 인한 스톨 등 복합적인 원인으로 예상된다. 또한, NN, VA 벤치마크 프로그램은 CTA 할당 개수 증가에 따라 메모리 스톨이 크게 증가하지 않았지만, 성능은 증가하였다. 이러한 이유로는 해당 벤치마크 프로그램은 메모리 요청에 따른 메모리 스톨에 영향을 받지 않으며, CTA 할당 개수 증가에 따라 성능이 향상됨을 알 수 있다. 마지막으로 DHWD, KM 벤치마크 프로그램은 CTA 할당 개수 증가에 따라 성능의 변화가 없었고 메모리 스톨의 증가 또한 발생하지 않았다. 이러한 이유로는 벤치마크 프로그램에서 생성된 CTA 개수가 적거나 동시에 수행할 수 있는 CTA 할당 개수가 정해져 있는 것으로 판단된다.

이 장에서는 CTA 할당 개수 증가에 따른 다양한 벤치마크 프로그램의 성능을 분석하였다. 또한, 메모리 스톨에 따른 성능의 연관성을 분석해보았다. 실험결과에서 보는 바와 같이 대부분의 벤치마크 프로그램은 CTA 할당 개수 증가에 따라 하드웨어 자원의 활용률이 높아지기 때문에 성능의 향상을 확인하였다. 또한, CTA 할당 개수 증가에 따라 메모리 스톨이 증가하는 벤치마크 프로그램이라도 메모리 스톨에 의한 성능 감소보다 다수의 CTA를 사용함에 따른 성능 향상의 이점이 더 큼을 알 수 있다.

Ⅳ. 결 론

본 논문에서는 GPGPU 구조의 병렬성을 향상시키기 위해서 다양한 벤치마크 프로그램을 이용하여 CTA 할당 개수 변화에 따른 성능을 분석하고 성능과 메모리 요청과의 상관관계를 분석하였다. 실험결과 다수의 벤치마크 프로그램에서 CTA 할당 개수 증가에 따라 성능이 증가함을 확인할 수 있었다. 하지만, 특정 벤치마크 프로그램에서는 CTA 할당 개수가 증가함에도 불구하고 성능의 변화가 없는 경우가 발생하였고, CTA 할당 개수가 증가할 때 오히려 성능이 감소하는 경우도 발생하였다. 이러한 원인으로는 동시에 수행할 수 있는 CTA 할당 개수가 정해져 있거나 벤치마크 프로그램에서 생성된 CTA 개수가 적은 것으로 예상된다. 또한, CTA 할당 개수 증가에 따라 성능이 감소한 벤치마크 프로그램은 CTA 할당 개수 증가에 따라 메모리 충돌이나 SM 스톨, 하드웨어 자원 부족 등 복합적인 원인으로 인한 성능감소가 발생한 것으로 예상된다. 이러한 결과를 자세히 분석하기 위해서 CTA 할당 개수 증가에 따른 메모리 스톨을 분석하였다. 분석결과, CTA 할당 개수가 증가함에 따라 CTA에서 발생하는 메모리 요청이 증가하여 메모리 스톨이 증가하게 된다. 하지만, 메모리 스톨의 증가에 따른 성능 하락보다 CTA 할당 개수 증가에 따른 성능 향상의 이점이 더 큼을 알 수 있었다. 메모리 스톨이 크게 증가하지 않는 벤치마크 프로그램들은 CTA 할당 개수가 증가함에도 메모리 요청이 크게 증가하지 않음

으로 메모리와 성능과의 연관성은 발견되지 않았다.

참고 문헌

[1] J. Y. Chang, W. J. Kim, K. J. Byun, and N. W. Eum, "Performance Analysis for Multimedia Video Codec on On-Chip Network," KISM Smart Media Journal, Vol. 1, No. 1, pp. 27-35, 2012.

[2] S. B. Heo, J. H. Park, and H. S. Jo, "An performance analysis on SSD caching mechanism in Linux," KISM Smart Media Journal, Vol. 4, No. 2, pp. 62-67, 2015.

[3] V. Agarwal, M.S. Hrishikesh, S.W. Keckler, and D. Burger, "Clock rate versus IPC: the end of the road for conventional microarchitectures," In Proceedings of the 27th International Symposium on Computer Architecture, pp. 248-259, 2000.

[4] K. Olukotun, B.A. Nayfeh, L. Hammond, K. Wilson, and K. Chang, "The Case for a Single-Chip Multiprocessor," In Proceedings of 7th Conference on Architectural Support for Programming Languages and Operating Systems, pp. 2-11, 1996.

[5] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger, "Clock rate versus IPC: the end of the road for conventional microArchitectures," In Proceedings of 27th International Symposium on Computer Architecture, pp. 248-259, 2000.

[6] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," IEEE Computer, Vol. 41, No. 7, pp. 33-38, 2008.

[7] S. Y. Lee, A. Arunkumar, and C. J. Wu, "CAWA: coordinated warp scheduling and cache prioritization for critical warp acceleration of GPGPU workloads," In Proceedings of the 42nd Annual International Symposium on Computer Architecture, pp. 515-527, 2015.

[8] D. Voitsechov, and Y. Etsion, "Single-graph multiple flows: Energy efficient design alternative for GPGPUs," In Proceedings of the 41st Annual International Symposium on Computer Architecture, pp. 205-216, 2014.

[9] M. Lee, S. Song, J. Moon, J. Kim, W. Seo, Y. G. Cho, and S. Ryu, "Improving GPGPU resource utilization through alternative thread block scheduling," In Proceedings of 20th IEEE International Symposium on High Performance Computer Architecture, pp. 260-271, 2014.

[10] H. J. Lee, K. J. Brown, A. K. Sujeeth, T. Rompf, and K. Olukotun, "Locality-Aware Mapping of Nested Parallel Patterns on GPUs," In Proceedings of 47th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 63-74, 2014.

[11] H. J. Choi, and C. H. Kim, "Analysis on Memory Characteristics of Graphics Processing Units for Designing Memory System of General-Purpose Computing on Graphics Processing Units," KISM Smart Media Journal, Vol. 3, No. 1, pp. 33-38, 2014.

[12] I. A. Buck, "Programming CUDA," In Supercomputing 2007 Tutorial Notes, 2007.

[13] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," In Proceedings of 9th International Symposium on Performance Analysis of Systems and Software, pp. 163-174, 2009.

[14] S. Li, J. H Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," In Proceedings of the International Symposium on Microarchitecture, pp. 469-480, 2009.

[15] NVIDIA's Next Generation CUDA Compute Architecture: Fermi, available at www.nvidia.com/content/pdf/fermi_white_papers/nvidia_fermi_compute_ architecture_whitepaper.pdf

[16] CUDA SDK, available at http://developerdownload.nvidia.com/compute/cuda/sdk/website/samples.html

[17] S. Che, M. Boyer, M. Jiayuan, D. Tarjan, J. W. Sheaffer, S. H. Lee, and K.Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," In Proceedings of the International Symposium on Workload Characterization (IISWC), pp. 44-54, 2009.

저 자 소 개

손동오 (학생회원)

2010 : 전남대학교 전자컴퓨터공학부 공학사

2012 : 전남대학교 전자컴퓨터공학과 석사

현재 : 전남대학교 전자컴퓨터공학과 박사과정

관심분야 : 컴퓨터구조, 임베디드시스템

심규연 (학생회원)

2015 : 전남대학교 전자컴퓨터공학부 공학사

현재 : 전남대학교 전자컴퓨터공학과 석사과정

관심분야 : 컴퓨터구조, 임베디드시스템, 메모리구조

김철홍(정회원)

1998 : 서울대학교 컴퓨터공학사

2000 : 서울대학교 컴퓨터공학부 석사

2006 : 서울대학교 전기컴퓨터공학부 박사

2005-2007: 삼성전자 반도체총괄 SYS.LSI사업부 책임 연구원

현재 : 전남대학교 전자컴퓨터공학부 교수

관심분야 : 임베디드시스템, 컴퓨터구조, SoC 설계, 저전력 설계