2013년 4월 전자공학회 논문지 제 50 권 제 4 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 4, April 2013 http://dx.doi.org/10.5573/ieek.2013.50.4.089

논문 2012-50-4-12

GPU 아키텍처의 AES 암호화 성능 예측 분석 모델

(An Analytical Model for Performance Prediction of AES on GPU Architecture)

김 규 운*, 김 현 우**, 김 희 정**, 허 태 영**, 정 상 혁**, 송 용 호***

(Kyuwoon Kim, Hyunwoo Kim, Huijeong Kim, Taeyoung Huh, Sanghyuk Jung, and Yong Ho Song)

요 약

컴퓨터의 그래픽 연산장치인 GPU는 그래픽 데이터의 연산뿐만 아니라 일반시스템 데이터를 처리할 수 있도록 발전되었으며, 3D 그래픽 관련 알고리즘이나 병렬 실행이 가능한 코드에 대해서는 CPU 보다 우수한 성능을 보여주고 있다. CPU 기반으로 제작된 일반적인 알고리즘을 GPU에서 실행하기 위해서는, GPU 시스템의 아키텍처를 이해하고 병렬처리 능력과 새로운 메모리 구조를 고려하여 코드를 재작성하여야 한다. 이를 위해서는 알고리즘을 성능 예측 모델에 적용하여 GPU 시스템에서 예상되는 성능 예측이 필수적이다. 이를 통해 GPU 기반 어플리케이션 개발에서 발생할 수 있는 문제점들을 사전에 예측하고, 성능에 대한 평가 지표를 구성할 수 있다. 본 논문에서는 AES 암호화 알고리즘에 성능예측 모델을 적용하여 작업량이 많은 조건하에서 높은 정확도로 성능 예측을 수행하였다.

Abstract

The graphic processor unit (GPU) has been developed to process not only graphic data but also general system data. It shows a better performance than CPU in algorithm for 3D graphics and parallel program. In order to execute algorithm for CPU on GPU, we should understand about GPU architectures and rewrite program considering parallel processing capability and new memory model of GPU. For this reasons, a performance prediction model for the algorithm and its predicted performance through GPU system are required. These can predict problems in GPU application development or construct a performance evaluation standard for GPU. In this paper, we applied the AES encryption algorithms on our performance model and accomplished performance prediction with high accuracy under a heavy workload.

Keywords : CUDA, GPU, AES, HPC, parallel processing

* 정회원, LG전자 주임연구원
(LG Electronics Research Engineer)
** 학생회원, 한양대학교 전자컴퓨터통신공학과
(Dept. of Electronics and Computer Engineering, Hanyang University)
*** 정회원, 한양대학교 융합전자공학부
(Dept. of Electronic Engineering, Hanyang University)
※ 이 논문은 2011년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임
(한국연구재단-2011-0017147)
접수일자: 2012년12월27일, 수정완료일: 2013년3월18일

Ⅰ. 서 론

컴퓨터에서 그래픽 연산을 담당하는 기존의 GPU는 3차원 그래픽에 사용되는 부동 소수점 데이터를 빠르게 처리하기 위해 동시에 수행 가능한 수 십 또는 수 백 개의 코어로 구성되어 있다. 이러한 기존의 GPU 코어를 그래픽 데이터 처리가 아닌 일반시스템 데이터 처리에 활용하기 위해서는 연산 기능을 변경할 필요가 있다. 정수 데이터 연산을 가능하게 하고, 일반 프로그램

에 쓰이는 다양한 논리연산, 산술연산, 조건문, 분기문 등을 처리하는 명령어를 실행할 수 있게 GPU를 발전시킨 프로세서를 *GPGPU(General-purpose computing on graphics processing units)*, 또는 간단히 GPU라고도 한다. GPGPU는 다체 문제(N-Body), 유체 역학 시뮬레이션 등에서 기존의 CPU 대비 우수한 성능을 보여주고 있다.[1~2] 그러나 병렬화 되지 않은 알고리즘에서는 상대적으로 느린 성능을 보여주는데 이것은 CPU와 GPGPU가 다른 구조를 갖고 있기 때문이다.[3] GPGPU를 통해 프로그램을 가속하기 위해서는 GPGPU 장치의 메모리 구조를 잘 이해하고 병렬 실행 가능성을 예측해서 기존의 알고리즘을 GPGPU에 적합한 프로그램으로 작성해야 한다. 이를 위해서는 성능 예측 모델을 만들고 알고리즘의 특성을 반영하여 시스템에서 예상되는 성능을 예측하도록 해야 한다. 또한 이를 통해 구현 할 알고리즘이 GPGPU에서 빠르게 실행될 수 있는지 미리 예상할 수 있도록 한다.[4]

기존에도 GPGPU의 구조와 특징을 분석하고 이를 수치화 하여 모델링을 시도하는 연구들이 있었다. 이중에는 마이크로 벤치마크를 통해 GPU에서 실행되는 명령어의 소비전력을 측정한 후 이를 전체 GPU 코드에 반영시켜서 시스템에서 요구하는 전력을 분석하거나 이 논문에서 추구하고자 하는 것과 유사하게 병렬실행능력과 메모리별 접근속도를 고려하고 수치화하여 GPU 코드에 적용하여 성능을 예측하고 병목점을 예측하였다.[5~6] GPU는 그 내부가 명확하게 공개되어있지 않기 때문에 마이크로 벤치마크를 통해 성능을 유추하고 이를 기반으로 분석 모델을 제시하고 있다.[7] 이러한 GPU 모델링 연구는 이미 만들어진 GPU코드를 기반으로 실행성능을 예측했다. 하지만, 이번 연구에서는 CPU 기반으로 된 코드에 대해 시스템 파라미터를 분석하여 이를 적용하였다. 이에 따라 정밀하게 튜닝된 GPU 코드에 비해 정확도는 떨어질 수 있으나 변환된 성능의 최저점을 예측하는데 도움이 될 수 있도록 하였다.

이 논문에서는 GPGPU와 시스템을 모델링 하기위해 파라미터를 만들고 이를 암호화 알고리즘에 적용하여 성능을 예측하였다. 암호화 알고리즘은 대칭 블록 암호 알고리즘인 *AES(Advanced Encryption Standard)*를 사용하였으며, 이에 대한 성능 예측과 실제 구현한 성능을 비교하였다.[8][9]

Ⅱ. 본 론

1. GPU 시스템

GPU 시스템의 구성은 그림 1과 같다.[10]

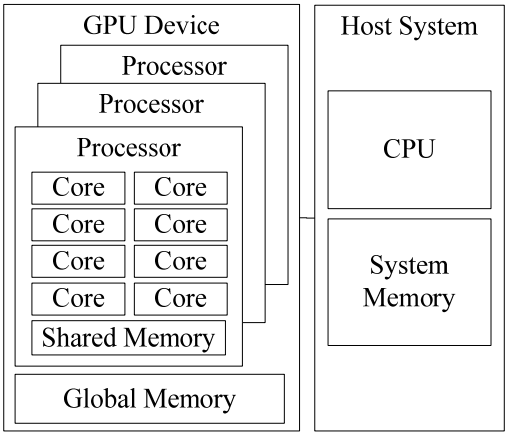

그림 1. GPU 시스템의 구성
Fig. 1. GPU system architecture.

그림 1에서와 같이 GPU는 여러 개의 프로세서와 전역 메모리 이루어져 있고 각 프로세서는 다시 여러 개의 코어와 공유 메모리로 이루어져 있다. 코어의 내부에서는 레지스터를 사용해서 연산을 하고, 데이터는 프로세서 단위로 공유 메모리를 통해 공유할 수 있다. 또한 호스트 시스템과 프로세서 간에 데이터 공유를 위한 전역 메모리가 있다. GPU는 코어의 수와 작동속도, 메모리의 크기 등이 설계 업체마다 다양할 수 있으며 특히 Embedded System에서는 특정 메모리의 존재 유무도 달라질 수 있다. 따라서 시스템 성능 예측을 위해서는 사용하고자 하는 시스템의 정확한 이해가 필요하다.[11~12]

표 1. 계층적 메모리의 구조
Table 1. Hierarchical memory structure.

Type	Location	Cached	Access	Scope	Lifetime
Register	On	no	R/W	thread	Thread
Local	Off	no	R/W	thread	Thread
Shared	On	no	R/W	block	Block
Global	Off	no	R/W	all	Host
Constant	Off	yes	R	all	Host
Texture	Off	yes	R	all	Host

PC에서 주로 사용되는 NVIDIA GPU의 계층적 메모리를 정리하면 표 1과 같다. 각 메모리들은 코어 내부에 존재하거나(on chip) 코어 외부에 존재할 수 있으며(off chip), 메모리에 따라 GPU의 메모리 접근 권한이 다르다. 또한 메모리에 로드 된 데이터의 유지 시간이 다르므로 커널 함수에서 공유할 수 있는 범위가 제한적이다. CPU는 캐시메모리 제어기가 메모리 사용을 효율적으로 관리하고 있지만, GPU 아키텍처에서는 효율적인 데이터 처리를 위해서 공유 메모리를 캐시메모리로 사용하여 직접 관리할 필요가 있다.

2. 성능 예측 모델

성능 예측을 하기 위해서 시스템에 종속적인 부분을 아키텍처 파라미터로 정의하고 소스 코드의 분석을 통해 얻어지는 부분을 코드 파라미터라고 정의하면, 파라미터들은 표 2와 같이 나타낼 수 있다.

호스트 시스템과 GPU 장치의 수치적인 특성, 그리고 그에 따른 실제적인 성능을 반영한 아키텍처 파라미터는 장치간의 메모리 전송속도, 커널 실행에 수반되는 오버헤드, GPU의 클럭 속도와 파이프라인 스테이지라고 정의할 수 있다. 이 파라미터들을 결정하기 위해서는 이론적으로 계산한 값과 테스트 코드를 통해 산출된 결과를 이용한다. 먼저, 메모리 전송속도는 커널 함수의 시작 전에 입력데이터를 호스트 시스템의 메인 메모리에서부터 GPU의 전역 메모리로 전송하는 시간과 커널 함수 종료 후에 결과 GPU의 전역메모리에서부터 호스트시스템의 메모리로 데이터를 가져오는 시간으로 볼 수 있다. 이 메모리 전송 시간은 시스템을 구성하는 시스템 버스, 메모리 컨트롤러, 메인 메모리 등의 성능에 의해 결정되므로 실험을 통해 획득한다. 커널 실행에 수반되는 오버헤드는 GPU에서 커널 코드를 로드하는 시간, GPU의 디바이스 드라이버를 통해 호스트 시스템에서 커널을 실행하는데 걸리는 시간, 커널실행 후에 호스트 시스템으로 복귀하는 시간이 포함되며, 이것은 아무 작업도 하지 않는 이른바 NULL 커널을 실행시켜서 얻어지는 상수 시간으로 결정된다. 이와 같이 산출된 파라미터들과 GPU 코어의 클럭 속도, 파이프라인 스테이지 파라미터를 함께 고려하여 시스템을 모델링한다. 알고리즘에 따라 파이프라인이 중지되는 경우도 있지만, NVIDIA의 GPU는 4스테이지의 파이프라인이 있으므로 클럭 속도가 4배가 되는 효과를 기대할 수 있다.[13]

표 2. 모델링 파라미터
Table 2. Modeling Parameters.

Architectural Parameter	Code Parameter
GPU System - Memory transfer - Kernel launch - Clock speed - Pipeline stages	Source Code - Parallel execution - Instructions - Divergence

소스코드 파라미터는 CPU기반으로 작성된 코드를 GPU 아키텍처의 특성을 고려하여 재해석하는 작업이 필요하다. 우선 전체 코드의 수행시간은 각 코어에 할당된 작업의 개수와 하나의 작업이 코어에서 실행되는 시간으로 볼 수 있다. 이는 커널 함수의 병렬 실행 가능성과 병렬화 된 연산의 조밀도 관점에서 볼 수 있다. 각 코어들은 *SIMT(single instruction multiple thread)*구조로 같은 명령어를 동시에 실행하게 되므로 병렬 실행 가능성은 데이터 의존성에 따라 달라진다. 예를 들어, 벡터의 덧셈과 같이 각 요소들 간에 의존성이 없는 경우에는 병렬 처리가 가능하지만, 피보나치의 수열과 같은 알고리즘의 경우 각 데이터 간 의존성이 생기므로 병렬처리가 불가능하다. 또한, 병렬처리가 가능한 경우라도 모든 연산을 너무 작게 병렬화한다면 연산 후 데이터들을 동기화 하는 오버헤드가 오히려 증가하여 시스템의 성능을 저하시키는 원인이 된다.

커널 함수 명령어의 실행시간은 각 산술/논리 연산에 소요되는 시간과 메모리 접근에 걸리는 시간의 합으로 볼 수 있다. NVIDIA의 GPU는 정수 연산에 4 cycle이 소요되지만 메모리 접근은 400-800 cycle이 소모된다. 상대적으로 오래 소요되는 메모리 접근 시간은 프로세서에서 코어의 수보다 많은 작업이 할당 된 경우 메모리 접근 시간 동안 대기 중인 작업을 처리함으로써 메모리 접근 시간으로 인한 지연을 감출 수 있다. 또한 연속적인 주소에 접근하는 경우 각 메모리에 대한 접근 요청을 하나로 모아서 처리함으로써 더욱 빠르게 처리가 가능하다. 이때 평균 메모리 접근 시간을 600 cycle이라 하고, 최대 16개의 메모리 접근을 한 번에 처리하는 경우 평균 메모리 접근 시간이 (600+16)/16=38.5가

되어 한번 메모리 접근하는 시간이 약 39 cycle 이 되는 효과를 기대할 수 있다.[14]

앞서 설명한 파라미터들 이외에도 GPU 성능 예측 모델링에서 고려해야 할 사항이 있다. 조건문의 경우 각 코어들이 실행하는 스레드가 해당 조건에 맞는 경우도 있고 맞지 않는 경우도 생기게 된다. 이럴 경우 SIMT의 특성상, 한 번에 같은 명령어만 실행 가능하므로 조건이 맞지 않는 코어들은 대기상태가 된다. 이 경우 코어의 이용도는 떨어지게 되는데 이를 divergence (발산)이라 한다. 마지막으로 GPU의 특성상 사용될 수 없는 정적 변수나 함수 포인터 코드의 경우에는 해당 기능을 수행할 수 있는 다른 코드로 변경하여 소스 코드 분석에 사용하도록 한다.

3. 암호화 알고리즘

실험에 사용할 암호화 알고리즘으로는 미국 정부 표준으로 지정된 AES를 사용하기로 한다. 본 논문에서는 AES를 GPU에 적용하기 위해 128 bit의 블록과 128 bit의 키를 갖는 AES를 사용하고, 암호화 모드로는 ECB(Electronic codebook)를 사용하기로 한다. 이로 인해 AES는 모든 블록에 같은 키를 사용하여 암호화하게 되므로, 각 코어들이 하나의 블록을 암호화하도록 함으로써 병렬화가 가능하다. 또한 비교 테스트를 위해 byte 단위로 연산하는 코드와 word (4byte)단위로 연산하는 코드에 대해 분석하도록 한다. AES의 구현은 TBox를 사용해서 연산 횟수를 줄이는 대신 테이블을 사용해서 결과를 계산하도록 하고 있다. 이는 상대적으로 빠른 shared memory의 사용 비중을 높이는 효과를 갖는다.[15]

AES 알고리즘은 AddRoundKey, ByteSub, ShiftRows, MixColumns, SubByte 라는 네 가지 스텝을 반복적으로 수행하여 암호화를 수행하고 있으며 암호화키의 길이에 따라서 반복횟수가 달라 질 수 있다. 이는 암호화된 값과 키 값을 이용하여 역순으로 계산했을 때 복호 될 수 있도록 설계된 특성에 따른 것이다. 각 스텝의 알고리즘은 다음과 같다.

- *AddRoundKey(text, expkey)*
  ```
  for each i in text
    text[i] ^= expkey[i]
  ```
- *ByteSub(text)*
  ```
  for each i in text
    text[i] = sbox[text[i]]
  ```
- *ShiftRow(text)*
  ```
  for each row in text
    text[row] = rotate(text[row], row)
  ```
- *MixColumns*
  ```
  for each col in text
    text[col] = M * text[col]
  ```

MixColumns에서 M는 역변환이 가능한 행렬이다. 실험에서 사용하는 알고리즘은 위의 알고리즘에서 M 행렬과 Sbox를 미리 곱해서 Tbox로 구성할 수 있는데, 행렬 곱셈이 네 개의 Row 단위로 계산되므로 256개의 인덱스를 갖는 네 개의 TBox Array를 생성할 수 있다. 그리고 ShiftRow를 하기 위해서 각 Row 별로 쉬프트 횟수를 달리 하여 변경하면 다음과 같이 알고리즘을 간략화 할 수 있다.

```
e0 = T0[(s0>>24)&0xff]^T1[(s1>>16)&0xff]
     ^T2[(s2>>8)&0xff]^T3[(s3>>0)&0xff]^RK[0]
e1 = T0[(s1>>24)&0xff]^T1[(s2>>16)&0xff]
     ^T2[(s3>>8)&0xff]^T3[(s0>>0)&0xff]^RK[1]
e2 = T0[(s2>>24)&0xff]^T1[(s3>>16)&0xff]
     ^T2[(s0>>8)&0xff]^T3[(s1>>0)&0xff]^RK[2]
e3 = T0[(s3>>24)&0xff]^T1[(s0>>16)&0xff]
     ^T2[(s1>>8)&0xff]^T3[(s2>>0)&0xff]^RK[3]
```

여기서 s0, s1, s2, s3는 text가 32bit 단위로 저장되어 있고, RK는 32bit로 된 RoundKey 이다. 마지막 Round에서는 MixColumns이 생략되므로 M 행렬 대신에 1로 이루어진 행렬에 SBox를 곱하여 다섯 번째 TBox를 만들어 사용한다. GPU에서 TBox를 사용할 때는 전역메모리의 하나인 constant memory에 저장하여 모든 코어에서 접근할 수 있도록 한다. 복호화를 위해서는 TBox의 역함수에 해당하는 다섯 개의 배열이 있지만 이 논문의 실험에서는 복호화를 하지 않으므로 복호화를 위한 TBox는 생성하지 않는다.

Ⅲ. 실 험

실험에 사용된 환경과 GPU의 사양은 표 3, 표 4과 같

표 3. 실험 환경
Table 3. Experimental environment.

CPU	Intel Core2Duo E7300@2.66GHz
RAM	DDR2 SDRAM 4GB
OS	Windows 7 Enterprise 32 bit
GPU	NVIDIA GeForce GTX260
Compiler	Visual Studio 2008 Professional

표 4. NVidia GeForce GTX260 특성
Table 4. Features of NVidia GeForce GTX260.

Compute Capability	1.3
CUDA Version	4.2
Driver Version	301.42
CUDA Core	216(27 * 8)
Processor Clock	1350MHz
Memory Clock	1053MHz
Graphics Memory	GDDR3, 448bit, 896MB

다. 모델링 방법을 검증하기 위해서 모델링을 통해 예측된 성능과 실제로 구현된 프로그램의 성능을 비교한다.

실험환경을 분석하기 전에 GPU시스템의 커널 수행을 추상화하면 다음과 같다.

- Host to Device memory copy
- Kernel launch overhead
- Kernel execution
- Device to Host memory copy

이중에서 메모리 전송과 커널 시작 오버헤드는 아키텍처 파라미터를 통해 분석할 수 있고, 커널 실행은 코드 파라미터 분석을 통해 실행 cycle을 분석하고 이를 아키텍처 파라미터에서 코어의 개수 및 프로세서의 Clock을 통해 분석하게 된다. 먼저, 메모리 전송속도는 이론적인 수치로 (Memory Clock)*(bit width)*(2 for DDR) = 1053MHz*448bit*2=117.9GB/s 의 속도를 얻을 수 있지만, 이것은 GPU와 GPU Memory 간의 전송속도이고, 실제 시스템메모리에서부터 데이터가 전송되는 속도는 1.5GB/s ~ 2.0GB/s 정도로 측정된다. 이것은 시스템 버스를 통해 paged-memory에 접근하고, API를 통해 이를 처리하기 되므로 상대적으로 낮은 성능을 보여주게 된다. 커널 시작 오버헤드는 실제 커널 내부를 비워두고 실행시킨 시간과 메모리 전송시간의 차를 통해 구할 수 있으며, 커널을 한 개 실행시킬 수준의 알고리즘에서는 무시할 정도의 시간이 소요된다.

이제 코드파라미터 분석을 위해 AES의 코드를 분석하면 다음과 같다. 각 코드는 32 bit 단위로 처리하도록 작성되었으며 128 bit를 처리하기 위해서는 4번 수행한다.

```
Load: s0=GETU32((pt+sb)+offset) ^ rk[0];
Round: t0=tbox0[(s0>>24)]^tbox1[(s1>>16)&0xff]
^tbox2[(s2>>8)&0xff]^tbox3[(s3)& 0xff]^rk[4];
Store: PUTU32((ct + sb) + offset, s0);
```

위 암호화 코드에서 Load와 Store에 사용되는 매크로 함수는 shift, and, or연산으로 구성되어있으며, 암호화 코드는 또한 byte 형 배열에서 32 bit 형으로 데이터를 추출 및 변환하기 위한 코드를 포함하고 있다. Round 코드의 경우는 4번씩 10번 수행해야 되며, 실제로 세 가지 형태로 되어있다. 각각 짝수/홀수/마지막 라운드에 대한 코드이며 약간의 차이점들이 있다. 위의 코드에서 사용되는 명령어를 메모리 접근횟수와 연산자의 수로 분류하면 다음과 같다.

```
Load: 8 load, 60 operation
Round: 800 load, 1,600 operation
Store: 4 store, 20 operation
```

여기서 연산자는 2 cycle으로 가정하여 cycle 단위로 치환하고, 메모리 접근은 횟수에 따라 cycle 단위로 치환하면, Load, Store의 메모리 접근은 암호화의 평문, 암호키, 암호문을 읽고 저장하는 것 이므로 순차적인 메모리 접근이 되어 한 번의 메모리 연산에 (600+16)/16 = 39 cycle이 소모될 것이다. Round 코드는 TBox 배열에서 랜덤하게 데이터를 읽어오므로, 메모리 접근이 랜덤하게 발생한다. 하지만 여러 개의 작업을 동시에 수행할 경우 각 작업에서 요구하는 배열의 인덱스가 겹치거나 근방에 있을 확률이 높아지므로, 충

분히 많은 양의 작업을 동시에 수행한다면 결국 순차적인 메모리 접근과 같게 되어 메모리 접근의 속도가 증가할 것이다. 전체 수행시간은 상대적으로 큰 메모리 접근시간에 의해 연산시간이 중첩되므로, TBox로 구현한 128 bit AES 암호화 연산에는 312 + 31,200 + 156 = 31,668 cycle이 소요되는 것을 알 수 있다. 이것은 코드 파리미터를 통해 코드를 분석하고 얻어진 것이고 이것을 아키텍쳐 파라미터의 병렬실행(클럭속도, 파이프라인)과 합치게 되면 다음과 같은 수식을 얻을 수 있다.

$$Perf_{KernelExec}(N) = \frac{\frac{N}{27\times 8}\times 31{,}688 cycle}{1350 MHz \times 4} \quad (1)$$

이 수식은 한번에 31,668 cycle이 소모되는 커널이 N개 있는 경우 8개의 코어로 이루어진 27개의 프로세서가 1,350MHz로 작동하고, 4 Stage의 파이프라인이 작동할 때 기대되는 성능을 예측한 것이다. 이것을 실제로 GPU에서 실행한 시간과 비교한 것은 다음과 같다.

입력데이터를 특정배수로 증가시키면서 조건문 등을 제거하기 위해 전체 작업의 개수는 전체 코어 개수인 216의 배수로 하였다. 또한 한 프로세서 당 할당되는 작업의 개수는 8개에서 최대값인 512개까지 늘리고, 그 후

표 5. AES 커널의 측정된 시간과 예측한 시간의 비교

Table 5. Comparison between the measured time and the predicted time of AES Kernel.

(단위: us)

Blocks	Measured	Predicted	Ratio
216	71.29	5.86	0.08
432	122.73	11.73	0.10
864	135.40	23.46	0.17
1728	113.60	46.92	0.41
3456	127.14	93.83	0.74
6912	178.17	187.66	1.05
13824	396.51	375.32	0.95
27648	847.24	750.65	0.89
55296	1695.57	1,501.30	0.89
110592	3228.99	3,002.60	0.93
221184	6460.45	6,005.19	0.93

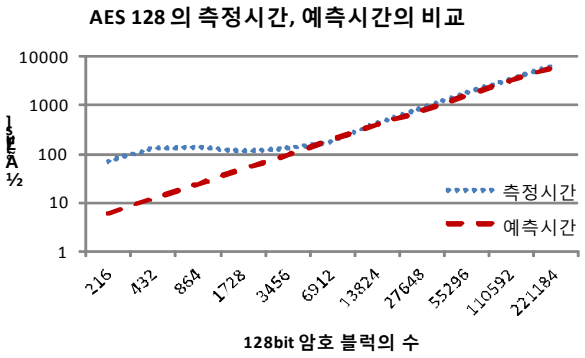

그림 2. AES 128의 암호화 블록 개수에 따른 측정시간, 예측시간의 비교 (단위: us)

Fig. 2. Comparison between the measured time and the predicted time depending on the number of AES 128 blocks.

에는 프로세서 당 할당되는 작업그룹의 개수를 2의 배수로 조절하였다. 예측 실험에서 가정한 수치만큼 알고리즘 내 메모리 접근들이 순차적 메모리 접근이 되기 위해서는, 충분히 많은 개수의 작업이 필요한 것을 알 수 있다. 커널 함수의 정확한 수행시간을 측정하기 위해 동기화 방식으로 커널함수를 호출하고, NVIDIA의 Parallel NSight Visual Studio Edition 기반으로 실행시간을 측정하였다. 실험결과 적은 수의 작업을 실행할 때는 파이프라인 효과와 순차적 메모리 접근에 대한 효과를 기대할 수가 없고 또한 병렬성이 매우 낮아지게 되므로 모델링을 통해 예측한 시간이 부정확하게 되었지만, 128 bit AES 알고리즘에서는 13824개 이상의 블록 입력에 대해서 약 91% 정도의 정확도를 보여주었다. 이것은 AES 알고리즘 자체가 간단하게 이루어져 있고, 병렬성이 충분히 보장되는 상황을 가정했으므로 높은 수준의 정확도를 보여준 것이다. 만일 병렬실행이 어려운 알고리즘이나, 작업 개수가 적은 경우에는 모델링을 다르게 해야 될 것이다.

Ⅳ. 결 론

본 논문에서는 GPU의 병렬 실행 성능을 예측하기 위해 전체 시스템을 아키텍처 파라미터와 코드 파라미터로 분류하고, 각 지표에 대해 모델링을 시도하였다. 범용으로 사용할 수 있게 개발된 GPU는 수백 개의 코어를 내장하고 있어서 계산능력이 뛰어나지만 CPU기

반 시스템과는 다른 구조를 갖는 시스템에서 실행되기 때문에 알고리즘을 새로 작성해야 된다. 이 때, GPU용 커널 개발을 하기 전에 해당 알고리즘의 성능을 미리 예측하는 것은 개발 목표를 쉽게 설정할 수 있게 하고 성능 향상 여부를 가늠할 수 있게 하므로, 이런 모델링 작업은 필수적으로 동반되어야 한다. 이를 위해 본 논문에서는 모델링을 위한 파라미터를 설정하고, 이를 AES 알고리즘에 적용한 후 예측된 성능과 실제 성능을 비교하였다. 알고리즘이 GPU 코어 수에 맞는 수준 이상의 큰 데이터에 대해서는 비교적 높은 수준의 정확도로 예측된 성능을 보여주었고, 이를 확대 적용하면 다른 알고리즘에 대해서도 같은 모델링 방법으로 성능예측을 할 수 있을 것이다. 본 논문에서는 GPU 구조의 핵심인 병렬화를 기반으로 알고리즘의 수행시간을 예측하였다. 이를 어플리케이션 전체로 확장하기 위해서는 GPU시스템에서 제공되는 다양한 API들에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] Lars Nyland, Mark Harris, and Jan PrinsFast, "N-Body Simulation with CUDA", GPU gems 3, pp 677-695, 2007.

[2] Khajeh-Saeed, A, "Computational Fluid Dynamics Simulations Using Many Graphics Processors", Computing in Science & Engineering, Vol 14, Issue 3, pp 10-19, May 2012.

[3] John Nickolls, Ian Buck, Michael Garland and Kevin Skadron , "Scalable Parallel Programming with CUDA," Queue - GPU Computing, Volume 6, Issue 2, pp 40-53, Mar 2008.

[4] 김규운, "GPU 아키텍처의 병렬 어플리케이션 성능 예측을 위한 분석 모델", 한양대학교 대학원, 2011년 2월

[5] Sunpyo Hong, and Hyesoon Kim, "An integrated GPU power and performance model," in Proc. International Symposium on Computer Architecture, Vol. 38, Issue 3, June 2010.

[6] Sunpyo Hong, and Hyesoon Kim, "An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness," in Proc. International Symposium on Computer Architecuter, Vol. 37, Issue 3, June 2009.

[7] Yao Zhang, and John D. Owens, "A Quantitative Performance Analysis Model for GPU Architectures," High Performance Computer Architecture, 2011 IEEE 17th International Symposium on, pp 382-393, Feb 2011.

[8] Manavski. S.A., "CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography," in Proc. IEEE Conf. on Signal Processing and Communications, pp. 65-68, Dubai, United Arab Emirates, Nov 2007.

[9] 염용진, 조용국, "GPU용 연산 라이브러리 CUDA를 이용한 블록암호 고속 구현", 정보보호학회논문지 제18권 제3호, pp 23-32, 2008년 6월

[10] David B. Kirk and Wen-Mei W. Hwu, "*Programming Massively Parallel Processors: A Hands-on Approach,*" Morgan Kaufmann Publishers, 2010.

[11] Leskela, J., Nikula, J. and Salmela, M. "OpenCL embedded profile prototype in mobile device," in Proc. of IEEE workshop on Signal Processing Systems, pp. 279-284, Tampere, Finland, Oct 2009.

[12] John Owens, "GPU architecture overview," in Proc. of ACM SIGGRAPH 2007 course 24, Article No. 2, San Diego, CA, USA, Aug 2007.

[13] Pawan Harish and P. J. Narayanan, "Accelerating Large Graph Algorithms on the GPU Using CUDA" Lecture Notes in Computer Science, High Performace Computing, Vol 4873, pp. 197-208, 2007.

[14] "*CUDA C Programming Guide*", Cuda Toolkit v5.0, NVIDIA, 2012.

[15] Joan Daemen and Vincent Rijmen, "*The Design of Rijndael*", Springer, 2002.

저 자 소 개

김 규 운(정회원)
2009년 한양대학교 미디어통신 공학 졸업(학사).
2011년 한양대학교 전자컴퓨터 통신공학과 졸업(석사).
2011년～현재 LG전자 주임연구원.
<주관심분야 : 임베디드 시스템, 멀티미 디어, GPGPU(CUDA) 등>

김 희 정(학생회원)
2012년 한양대학교 전자통신공학/ 컴퓨터 졸업(학사).
2012년～현재 한양대학교 전자컴 퓨터공학과 석사과정.
<주관심분야 : 컴퓨터구조, 임베 디드 시스템, 낸드 플래시 메모리 등>

정 상 혁(학생회원)
2008년 한양대학교 미디어통신공 학/컴퓨터 졸업(학사).
2010년 한양대학교 전자컴퓨터통 신공학과 졸업(석사).
2010년～현재 한양대학교 전자컴 퓨터공학과 박사과정.
<주관심분야 : 컴퓨터구조, 임베디드 시스템, 비 휘발성 메모리 등>

김 현 우(학생회원)
2011년 동의대학교 전자공학 졸업 (학사).
2012년～현재 한양대학교 전자컴 퓨터공학과 석사과정.
<주관심분야 : 멀티코어 시스템, 멀티미디어 코 덱, 임베디드 시스템 등>

허 태 영(학생회원)
2012년 한국해양대학교 전파공학과 졸업(학사),
2012년～현재 한양대학교 전자 컴퓨터공학과 석사과정.
<주관심분야 : 컴퓨터구조, 임베 디드 시스템, 낸드 플래시 메모리 등>

송 용 호(정회원)
1989년, 1991년 서울대학교 컴퓨 터공학과 졸업 (학사, 석사).
2002년 University of Southern California Electrical Engineering 졸업 (박사).
1991년～1996년 삼성전자 전임연구원.
1996년～2002년 University of Southern California 연구조교.
2002년～2003년 (주)조선 IT 연구소장.
2003년～현재 한양대학교 융합전자공학부 교수.
IEEE International Parallel and Distributed Processing Symposium 프로그램 구성위원.
<주관심분야 : 시스템 구조, SoC, NoC, 멀티미디 어, 비휘발성 메모리 등의 임베디드 시스템 구조 등>