GPGPU를 이용한 큰 정수 곱셈 고속화

최환석[O1] 박진석[2] 이창건[†1]
[1]서울대학교 컴퓨터공학부
[2]연세대학교 컴퓨터과학과
hschoi@rubis.snu.ac.kr, jins1@msl.yonsei.ac.kr, cglee@snu.ac.kr

Fast Big Integer Multiplication using GPGPU

Hwan-Suk Choi[O1] Jin-Suk Park[2] Chang-Gun Lee[†1]
[1]Department of Computer Science and Engineering, Seoul National University
[2]Department of Computer Science, Yonsei University

요 약

최근 GPU(Graphics Processing Unit)의 높은 병렬처리 성능을 활용하여 일반 응용프로그램의 수행 성능을 높이고자 하는 노력이 전개되어왔다. 그 대표적인 것이 NVIDIA사의 CUDA(Compute Unified Device Architecture) 프로그래밍 모델이다. 그러나 CUDA에서는 컴퓨터 과학 응용 분야에서 중요하게 쓰이는 큰 정수의 곱셈이 기본적으로 제공되지 않는다. GPGPU를 활용하여 큰 정수 곱셈의 고속화를 위한 병렬 알고리즘을 개발하고 검증한 결과, 1024bit의 큰 정수 100,000개의 곱셈에서 약19배의 성능 향상을 보였다.

1. 서 론

큰 정수(Big Integer 혹은 Large Integer)의 곱셈은 컴퓨터 과학 응용 분야에서 널리 쓰이는 중요한 연산들 중의 하나이다. 큰 정수란, 컴퓨터가 한 번에 처리할 수 있는 단위 크기 정수보다 더 큰 정수의 표현, 예를 들어 32비트 컴퓨터에서 2^{32}을 넘는 정수를 표현하는 것인데, 이에 대한 연산은 여러 응용에서 필요하다. 그 예로는 GPS 데이터를 받아 위치를 계산하는 것과 공개키(Public Key) 암호화에 사용되는 것 등이 있다. 또한 데이터베이스 연산 처리를 위해 다양하게 활용될 수 있는데, 최근 SNS, 사물통신 등으로 얻어지는 문자, 영상, 위치정보 등의 크고 많은 데이터 즉, 빅데이터를 표현하기 위해 큰 정수 표현은 필수 불가결 요소이며, 이를 빠르고 효율적으로 처리하기 위한 기술의 필요성이 증대되었다. 이에 관련하여 병렬 컴퓨팅이 중요한 기술로 부상하게 되었다.

또한 프로세서 동작 주파수의 향상이 물리적 한계에 부딪히면서, 프로세서는 코어의 개수 증가 형태로 발전되고 있다. 최근에는 2~16개 정도의 멀티코어를 가진 CPU에 비해 그보다 동작 주파수는 낮지만 수십~수천 개의 매니코어를 가진 GPU를 그래픽 연산 처리 이외의 용도에도 이용할 수 있게 한 GPGPU(General Purpose GPU) 기술이 등장하면서 CPU의 기능을 보조할 수 있게 되었다. 이에 따라 GPGPU의 높은 병렬처리 성능을 활용하여 일반 응용프로그램의 수행 성능을 높이고자 하는 노력이 전개되어 왔다.[1] 즉, 병렬 프로그래밍의 중요성이 커지고 있는데, 데이터베이스 연산 처리에도 이를 적용할 수 있다. 그림 1과 같이 CPU(Host)가 큰 데이터를 처리해야 하는 작업을 작게 분할해서 GPGPU(Device)에 할당함으로써 동시에 병렬적으로 처리하여 속도 향상을 도모하는 것이다.

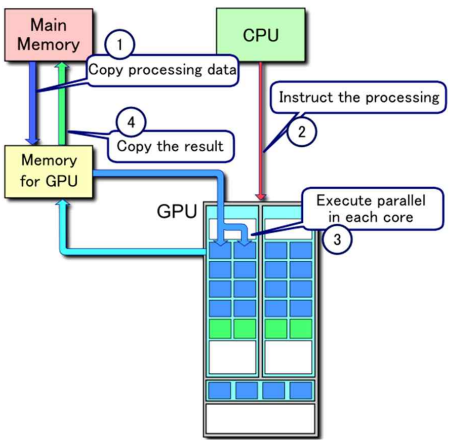

그림 1. GPGPU의 처리 흐름

대표적인 GPGPU의 두가지 흐름은 NVIDIA가 주도하는 CUDA[2]와 Khronos Group에서 주도하는 OpenCL[3]이 있다. 이들이 제공하는 프레임워크를 사용하면 일반적인 C 프로그램을 작성하는 것과 크게 다를 바 없는 방식으로 고성능 병렬처리 프로그램을 손쉽게 작성할 수 있다.[4][5] 그러나 이러한 프레임워크에서 기본적으로 큰 정수에 대한 연산이 정의되어있지 않기 때문에, 라이브

---

† 교신저자임

* 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.R0126-15-1105, 무인차를 위한 멀티코어 및 GPGPU 기반의 실시간 시스템 SW 개발).
* 이 연구를 위해 연구 장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다.

러리로서 개발할 필요가 있다.

본 논문에서는 NVIDIA사의 CUDA C프로그래밍 모델을 활용하여 큰 정수 곱셈 처리를 위한 병렬 알고리즘을 개발하고 검증하였다.

2. 큰 정수 곱셈의 병렬화 알고리즘

큰 정수는 컴퓨터가 컴퓨터 아키텍쳐상 표현 및 처리할 수 있는 최대 비트 수를 넘어가기 때문에 일반적으로 정수의 연속 즉, 배열(혹은 벡터)로써 표현한다. 배열의 크기에 따라 큰 정수의 비트수를 얼마든지 확장할 수 있다. 예를 들어, 32비트 연산이 가능한 GPGPU 프로세서에서의 단위 정수의 크기는 32비트이며, 크기 100인 배열로써 단위 정수들을 나열하면 3200비트의 큰 정수를 표현할 수 있다.

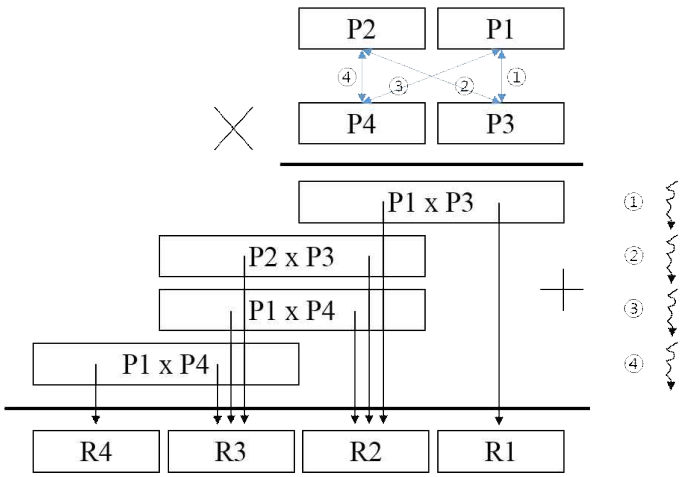

그림 2. 블록화한 병렬 곱셈 기법

본 논문에서 제시하는 큰 정수의 곱셈은 큰 정수를 블록화 하여 각각을 곱하는 작업을 스레드에 분배하며(그림 2의 ①~④), 이들이 계산한 부분적(Partial Product)을 합하는 작업까지 병렬로 처리하도록 한다. 여기에서 중요한 것은 1)길이 n비트 정수끼리 곱한 부분적은 최대길이가 2n이며, 2)부분적을 병렬적으로 합산할 때 생기는 올림수(Carry)에 대한 확인이 필요하다는 것이다.

1)의 문제 때문에 블록의 크기는 그 프로세서가 연산할 수 있는 최대 크기의 절반으로 잡아야한다. 예를 들어, 64비트 크기의 정수를 표현할 수 있는 프로세서에서 블록의 크기는 32비트로 정하여 부분합이 손실 없이 표현되도록 한다.

2)의 문제는 부분적을 합할 때 올림수가 발생하여 단위 크기로는 다 표현할 수 없는 즉, 오버플로우가 발생하게 되는 경우이다. 오버플로우는 스레드가 합을 계산하고 난 후의 결과 값이 이전 결과보다 작게 나타날 때 검출할 수 있으며, 이 때 올림수를 상위 비트에 반영해야한다. 이것은 올림수를 상위 블록(그림2에서 R1의 상위 블록은 R2이다)에 더해줌으로써 해결할 수 있다.

표1는 이를 구현한 알고리즘의 Pseudocode이다. 실제로는 GPGPU상에서 이를 이용하기 위해서 위 코드를 CUDA C로 구현하였으며, CPU가 CUDA C 커널 함수를 호출함으로써 GPU에게 작업을 담당시키게 된다. 또한 GPU가 CPU의 메모리를 직접 접근할 수는 없기 때문에 그림 1에서 보았던 것처럼, 데이터들을 GPU의 메모리로 전송하며 결과 값을 CPU의 메모리로 다시 가져오는 작업이 필요하다.[6]

표 1. 병렬 곱셈 알고리즘 Pseucode

```
Struct {block[32], length} BigInt
Input: multiplicand, multiplier
Output: BigInt result
1: for(i from 0 to multiplicand.length)
2: for(j from 0 to multiplier.length)
3: partial_product := multiplicand * multiplier
4: temp_block[0] := MSB/2 to 0 bit of the
   parital_product
5: temp_block[1] := MSB to MSB/2 bit of the
   parital_product
6: k := i + j
7: result.block[k] := result.block[k] + temp_block[0]
8: if(result.block[k] < temp_block[0]
10: do result.block[k+1] := result.block[k+1] + 1
11: k := k + 1
12: while(result.block[k] != 0)
13: k := i + j + 1
14: result.block[k] := result.block[k] + temp_block[1]
15: 8~13 again
16: end for
17: end for
```

3. 실험 결과

실험 환경은 표 2와 같다.

표 2. 실험 환경

CPU	Model	Intel(R) Core i3-550
	# of Cores	2
	Clock rate	3.29GHz
Mem	Size	4GB
GPU	Model	NVidia GeForce GTX 780
	# of Cores	2304
	Mem Size	3GB
S/W	OS	Ubuntu linux 12.04
	Compiler	GCC 4.4
	CUDA ver.	5.5

실험은 100,000개의 1024bit 16진수 정수 한 쌍의 곱셈을 각각 CPU와 GPGPU에서 수행하였다. CPU에서 순차적으로 수행한 시간과 GPGPU에서 병렬적으로 수행한 시간으로써 성능비를 측정하였다. 곱셈의 결과 검증은 기존에 CPU에서 큰 정수 곱셈이 가능한 Java 프로그램의 결과와 비교하였고, GPGPU에서 생성된 결과가 완전히 일치하는 것을 확인하였다. GPGPU내에서 Thread의 수에 따라 사용하는 core의 수가 달라지고 속도 향상에 미치는 영향이 있으므로 Thread 수를 달리하면서 여러 개의 실험 데이터를 생성하였는데, 이에 대한 실험 결과

는 그림 3과 같다. 그림3의 x축은 Thread의 수이며 왼쪽 y축은 수행시간(단위 msec)이다. 왼쪽 막대그래프(빨강)는 CPU로 처리했을 때의 계산시간, 오른쪽 막대그래프(파랑)는 GPU로 처리했을 때의 계산시간이며 꺾은선 그래프는 성능비를 나타낸다. 그림 3의 결과로서 100,000개의 1024bit 정수에 대한 곱셈 수행이 GPU가 CPU에 비해 최대 약 19배까지 빠르게 되었음을 확인할 수 있다. 코어 수에 비례한 성능향상을 보이지 않았는데, 이것은 코어의 동작 주파수 차이, 병렬화가 불가능한 부분들, 데이터 전송에 따른 오버헤드 그리고 GPU 아키텍처 특성 때문으로 보여진다.

그림 3. CPU와 GPU에서의 100,000 데이터 곱셈 시간

4. 결론 및 향후 과제

컴퓨터 과학 응용 분야에서 중요한 큰 정수 연산의 고속화를 위하여 GPGPU 기반 병렬 처리 알고리즘을 개발하였다.

실험 결과, 이 알고리즘으로서 CPU보다 최대 19배 빠르게 계산하였다. 이것을 더욱 발전시키고 응용하여 향후 실제 데이터베이스 응용에 적용하도록 할 수 있다. 기본적인 연산에서의 이러한 성능향상을 토대로 다양한 데이터베이스 응용에서의 실질적인 성능 향상에 큰 기여를 할 수 있을 것으로 기대된다.

GPU 아키텍처 특성상 성능 향상의 한계가 있었고, 이 때문에 코어의 개수에 비례한 성능 향상을 볼 수 없었다. 특히 CUDA의 구조상 메모리의 크기와 Thread의 개수에 영향을 많이 받게 되는데 이런 측면에서의 최적화가 필요하다.[7]

참고 문헌

[1] 정무경, 박성모, 엄낙웅, 2009, “병렬 프로세서 기술 및 동향”, 전자통신동향분석 제24권 제6호, ETRI

[2] CUDA, Compute Unified Device Architecture, Available: http://www.nvidia.com/object/cuda_home_new.html

[3] OpenCL, Open Computing Language, Available: https://www.khronos.org/opencl/

[4] Benedict R Gaster, Lee Howes, David R. Kaeli, Perhaad Mistry & Dana Schaa, 2014, “Heterogeneous Computing with OpenCL”, 2nd Edition, Morgan Kaufmann

[5] NVIDIA, “CUDA C Programming Guide”, Available: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

[6] 정영훈, 2011, “CUDA 병렬 프로그래밍 : 고성능 GPGPU를 이용한 NVIDIA 병렬 컴퓨팅 아키텍처 CUDA”, 프리렉

[7] 임은지, 이화영, 2013, “CUDA 프로그래밍 튜토리얼”, 정보과학회지 31(4), 한국정보과학회