TON(To Offload or Not): GPGPU 응용의 더 나은 수행 성능을 위한 오프로딩 결정 모델

마정현[0], 박세진, 박찬익

포항공과대학교 컴퓨터공학과 (경북 포항시 남구 효자동 산 31번지)
{doitnow0415, baksejin, cipark}@postech.ac.kr

TON (To Offload or Not): An Offloading Decision Model for Better Performance in GPGPU Applications

Jeonghyeon Ma[0], Sejin Park, Chanik Park
Department of Computer Science and Engineering, POSTECH, Pohang, South Korea

요 약

본 논문에서는 기존 오프로딩 결정 모델을 개선한 GPGPU 오프로딩 결정 모델을 제안한다. 개선된 오프로딩 결정 모델은 GPGPU 응용의 오프로딩 수행 시 발생할 수 있는 추가적인 오버헤드를 고려하고 있다. 즉, 네트워크 대역폭과 데이터 크기에 따른 네트워크 전송 시간 뿐만 아니라 로컬과 원격지간의 메시지 전달 횟수에 따른 네트워크 지연시간의 오버헤드를 포함하고 있고, 또한 원격수행을 지원하기 위한 추가적인 CPU 수행의 오버헤드 역시 포함하고 있다.

행렬 곱 32 x 32 워크로드의 경우 기존 오프로딩 결정 모델에서 실제수행 시간과 약 4배의 차이를 보였지만, 개선된 오프로딩 결정 모델에선 실제 수행 시간과의 차이가 약 1.3배로 개선되었다. 이러한 실제 수행과의 시간차이를 줄이는 것은 더욱 정밀한 오프로딩 결정을 가능하게 한다.

1. 서 론

현재 Graphic Processing Unit (GPU)은 디스플레이를 위한 자원으로 사용되는 것뿐만 아니라 기존에 CPU에 의해 처리되던 compute-intensive 워크로드들의 수행 성능 향상을 위해 다양한 분야에서 사용되고 있다. 이러한 추세는 모바일 응용에도 마찬가지로 적용되는데, 현재 GPGPU 응용들을 지원하기 위한 모바일 기기의 GPU 하드웨어 성능은 데스크탑 기반의 GPU에 비해 현저히 떨어지는 것이 사실이다. 기존에 이러한 모바일폰 성능의 한계점을 해결하기 위해 오프로딩 기술이 도입되었다.

모바일 기기의 GPGPU 오프로딩 기술은 원격의 GPGPU 자원을 사용하는 방식으로, 주로 수행 성능을 향상시키기 위한 목적과 에너지 소비를 절감시키기 위한 목적 두 가지로 나뉜다.[1] 일반적으로 오프로딩 기술을 통한 수행 성능의 향상은 연산에 사용될 데이터를 네트워크를 통해 전송하였을 때 소요되는 시간과 원격 수행 시간의 합이 로컬의 수행시간보다 작을 때 가능하다. 위 계산을 통해, 우리는 모바일 기기에서 수행되는 응용의 최적의 성능을 위해 응용을 원격지에서 수행할 것인지 로컬에서 수행할 것인지 정할 수 있다.

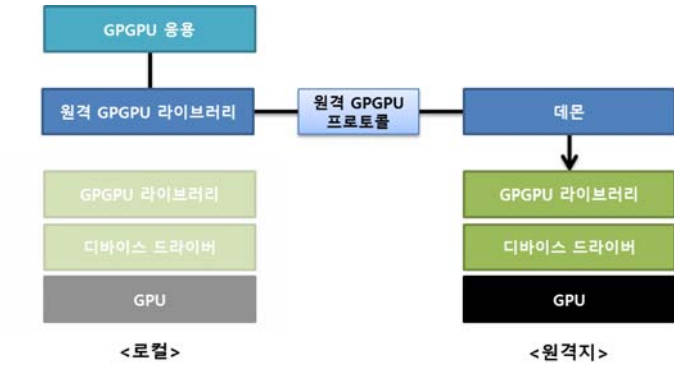
그러나 위의 계산식을 대입하여 GPGPU 응용의 원격지 수행을 결정하는 모델에 적용했을 때, 최적의 성능을 위한 응용의 수행 목적지가 원격지로 나타냈음에도 불구하고, 실제 오프로딩 프레임워크를 통해 수행된 결과는 로컬 수행의 성능이 최적인 경우가 발생한다. 이는 실제 GPGPU 응용의 오프로딩 수행을 위한 추가적인 오버헤드 때문에 발생한다.

이 논문에서는 GPGPU 응용의 오프로딩 가능한 프레임워크에서 최적의 성능을 위해 원격지와 로컬 수행을 결정하는 모델을 설계할 때 고려해야 할 추가적인 요소들에 대해 분석하고, 더 나은 GPGPU 오프로딩 수행 결정 모델을 제시한다.

2. 기초지식 및 관련연구

2.1. GPGPU 오프로딩 프레임워크

라이브러리 수준에서 GPGPU 응용에 대한 오프로딩 수행을 지원하는 기존의 프레임워크로 rCUDA[2], GViM[3], vCUDA[4], VOCL[5] 이 있다.



(그림 1) 라이브러리 수준에서 수행되는 기본 오프로딩 프레임워크

오프로딩 프레임워크는 VM 간의 수행이든 물리적인 로컬-원격지 간의 수행이든 상관없이 (그림 1)의 구조를 따른다. 응용에서 GPGPU 라이브러리를 호출하면 해당

라이브러리는 원격 GPGPU 라이브러리에 의해 인터포즈되어 원격지로 전달되고, 원격지에선 전달받은 라이브러리를 수행하여 수행 결과를 로컬로 전송하는 형식이다.

2.2. 오프로딩 결정 모델

응용의 성능 보장을 위한 최선의 선택을 위해서 오프로딩은 무조건적인 원격 수행이 아닌 선택적인 수행이 이루어져야 한다. 즉, 성능에 영향을 미칠 수 있는 다양한 요소를 고려하여 로컬에서 수행 할 것인지, 원격지에서 수행 할 것인지를 결정해야 한다.

기본적으로 다음의 오프로딩 결정 모델 (수식 1)을 만족하면 오프로딩을 수행한다.[1]

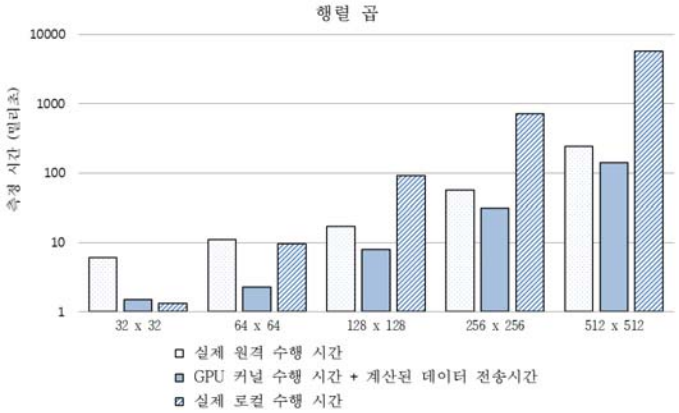
$$D_t + R_t < L_t \quad \text{(수식 1)}$$

(수식 1)에서 D_t는 데이터 전송시간을 의미한다. 데이터 전송시간은 전송되는 데이터 크기를 네트워크 대역폭으로 나눈 값이다. R_t는 워크로드의 원격지 수행 시간이고, L_t는 R_t에서 수행된 동일한 워크로드의 로컬 수행 시간이다.

3. 오프로딩 워크로드 오버헤드 요소 분석

본 연구에서 사용한 실험환경은 로컬PC의 경우 CPU: intel atom D510 1.67GHz, RAM: 2GB, GPU: Nvidia Geforce 210을 사용하였고, 원격지 PC의 경우, CPU: intel i7-3770k 3.50GHz, RAM: 16GB, GPU: ATI Radeon 7970HD를 사용하였다. 로컬과 원격지 간의 네트워크 연결은 Ethernet 100Mbps LAN을 사용하였다.

(그림 2)는 다양한 크기의 행렬 곱에 대해 실제 원격에서의 수행 시간, 실제 로컬에서의 수행 시간, 그리고 오프로딩 결정 모델을 통해 계산된 시간을 나타낸다.



(그림 2) 오프로딩 모델을 통한 계산 결과 값과 실제 로컬 및 원격지 결과 값의 시간 측정

실제 원격수행 시간과 오프로딩 결정 모델을 통해 계산된 데이터 전송시간을 비교해보면, 512 x 512와 같이 큰 크기의 응용에 대해선 시간차이가 많지 않지만, 32 x 32와 같이 작은 크기의 응용에 대해선 상대적으로 많은 차이가 발생함을 알 수 있다. 이는 전송될 데이터 크기가 작고 연산의 복잡도가 낮은 경우에 데이터 전송 크기뿐만 아니라 추가적인 오프로딩 오버헤드에 대한 영향을 받고 있음을 나타낸다. 오프로딩 결정 모델을 통한 계산 결과와 실제 원격 수행 성능의 차이로 인해, 위 경우에서도 잘못된 결정을 하게 된다. 64 x 64의 경우 로컬 수행 성능이 원격 수행에 비해 더 좋은 결과를 보이지만 오프로딩 결정 모델에 따라 응용을 원격으로 수행하게 된다.

3.1. 네트워크 지연 오버헤드

GPGPU 응용의 원격 수행 시 원격-로컬 간의 메시지 전달은 연산에 사용할 데이터를 보내는 작업뿐만 아니라 연산을 위한 초기화 과정에 필요하다. 초기화 과정에 필요한 메시지 전달은 데이터 크기가 약 수 바이트에서 수십 바이트로 매우 작고, 네트워크는 기본적으로 수십~수백 Mbps급을 사용하고 있기 때문에, 네트워크 대역폭에 영향보다는 네트워크 지연을 야기하는 메시지 전달 횟수에 영향을 많이 받는다.

간단한 행렬 곱 예제의 경우 GPGPU 연산을 실행하기 전 초기화를 위해 로컬과 원격지 간에 약 60회의 메시지 전달을 수행한다. (표 1)은 전달되는 메시지 크기의 평균인 40바이트의 메시지 전달을 60회 반복하였을 때의 수행 시간과, 2400바이트의 메시지를 한 번에 전달하였을 때의 수행 시간 차이를 나타낸다. 두 실험이 같은 네트워크 대역폭에서 수행되었다는 점과 실제 수행 시간의 차이를 통해 원격 수행을 위한 초기화 과정에는 네트워크 대역폭의 영향 보다는 다수의 메시지 전달로 인한 영향이 큰 것을 알 수 있다.

(표 1) 전체 크기 2400바이트 메시지의 분할 전송과 단일 전송의 수행시간 비교 (단위: 밀리초)

	분할 전송(40 x 60)	단일 전송(2400 x 1)
수행시간	1.98	0.04

따라서, GPGPU 응용을 위한 오프로딩 결정 모델에는 크기가 작은 데이터의 전달에 대한 오버헤드를 나타낼 수 있는 요소가 포함되어야 한다.

3.2. 오프로딩을 위한 CPU 추가 연산 오버헤드

GPGPU 오프로딩을 위한 프레임워크는 원격 수행을 지원하기 위해 로컬에서 원격지로 데이터를 전송하는데, 원격으로 데이터 전송을 수행하기 위해서는 중복된 데이터의 복사가 발생한다. 또한 GPGPU 커널 소스 데이터가 전송되거나 GPGPU 커널연산에 사용되는 데이터가 전송되는 경우 다양한 크기의 데이터 전송을 위해 데이터 크기를 확인하고 원격지에 메모리를 동적으로 할당하게 된다. 즉, GPGPU API 함수가 호출될 때마다 데이터 전송을 위한 중복 데이터 복사가 발생하고 때에 따라 메모리 할당도 이루어진다. 위 과정들은 오프로딩을 지원하기 위해 추가적으로 CPU에서 수행되어야 하는 작업들로 일반적인 오프로딩 결정 모델에 포함되어있지 않다. 간단한 행렬 곱을 이용한 실험결과 약 1ms의 추가적인 오버헤드가 발생하였고, 복잡한 워크로드 일수록 더욱 큰 오

버헤드가 발생한다.

따라서, 오프로딩 결정 모델에는 GPGPU 응용의 오프로딩을 위한 추가적인 CPU 연산의 오버헤드를 나타낼 수 있는 요소가 포함되어야 한다.

4. GPGPU 오프로딩 결정 모델

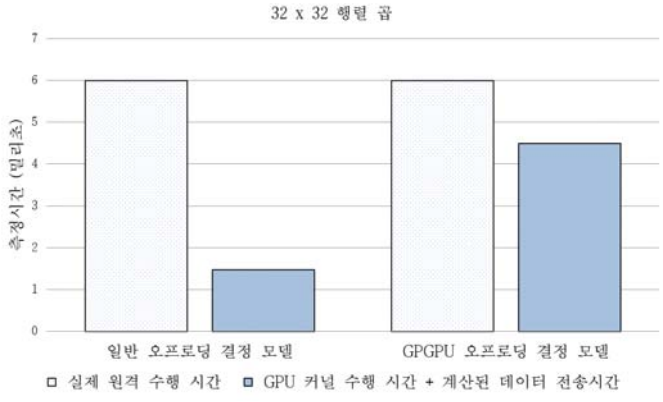
앞선 분석으로부터 GPGPU 오프로딩 결정 모델에 추가되어야 할 요소로 크게 네트워크 지연시간과 오프로딩을 위한 CPU 연산 오버헤드가 있다. 따라서 GPGPU 오프로딩 결정 모델은 다음과 같은 (수식 2)로 나타낼 수 있다. 기존 오프로딩 결정 모델과 마찬가지로 수식을 만족하면 오프로딩을 수행한다.

$$Large\,D_t + SmallD_t + CPUOverhead_t + R_t < L_t \quad \text{(수식 2)}$$

$LargeD_t$는 GPGPU 연산에 사용될 데이터 전송에 소요되는 시간이다. 즉, 네트워크 대역폭에 따라 큰 영향을 받는 시간이다. SmallD_t의 경우 GPGPU 연산을 위한 초기화 과정의 메시지 전달 데이터로 네트워크 대역폭 보다는 메시지 전달 횟수에 따라 더 큰 영향을 받는 시간이다. $CPUOverhead_t$의 경우 오프로딩을 위한 추가적인 CPU 연산 오버헤드 시간을 나타낸다. R_t와 L_t의 경우 이전 오프로딩 결정 모델의 정의와 같다.

5. 실험

(그림 3)은 GPGPU 오프로딩 결정 모델의 실제 적용 효과를 알아보기 위해 행렬 곱 32 x 32 워크로드에 대해 실제 원격수행 시간 결과 값과 오프로딩 결정 모델을 통



(그림 3) 일반 오프로딩 결정 모델과 GPGPU 오프로딩 결정모델의 비교

해 얻은 값에 대한 비교결과이다.

(그림 3)에서 볼 수 있듯이 새로운 GPGPU 오프로딩 결정 모델을 이용하면 약 3ms의 추가 오버헤드를 계산식을 통해 얻을 수 있다. 또한 이를 통해, 실제 수행 시간이 짧은 경우에도 GPGPU 오프로딩 결정 모델에 따라 계산된 결과 값과 실제 원격 수행시간의 차이가 상대적으로 적음을 알 수 있다. 따라서, GPGPU 응용의 로컬과 원격 수행을 선택할 때, 기존 오프로딩 결정 모델에 비해 정확한 판단을 할 수 있다.

6. 결 론

본 논문에서는 GPGPU 응용을 위해 기존 오프로딩 결정 모델을 개선한 GPGPU 오프로딩 결정 모델을 제안하였고, 이 모델을 GPGPU 응용인 행렬 곱 워크로드에 적용해 봄으로써, 기존 오프로딩 결정 모델에 비해 제안된 오프로딩 결정 모델이 얼마나 더 적합한지 알아보았다. 실험결과 수행 시간이 짧은 워크로드에도 개선된 오프로딩 결정 모델은 기존 모델보다 실제 시간에 근접한 정확도를 나타내었고, 이는 GPGPU 응용 수행 시 더욱 정밀한 오프로딩 결정을 할 수 있음을 의미한다.

Acknowledgement

본 연구는 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원과(No. 2011-0016972) 2012년도 지식경제부 산업원천기술개발 사업의 지원을 받아 수행 중인 디바이스 무설정 방식 이종사간 상호연동 기술 개발 (10041801) 과제의 일환으로 수행하였음.

참고문헌

[1] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, Bharat Bhargava; “A Survey of Computation Offloading for Mobile Systems”, In the Jounal of Mobile Networks and Applications, vol 18, pp.129-140, February 2013.

[2] Jose Duato, Antonio J. Pena, Federico Sila, Rafael Mayo and Enrique S. Quintana-Orti; “rCUDA: Reducing the number of GPU-based accelerators in high performance clusters.”, In Proc. of International Conference on High Performance Computing and Simulation (HPCS), pp.224-231, June 2010.

[3] Vishakha Gupta, Ada Gavrilovska, Karsten Schwan, Harshvardhan Kharche, Niraj Tolia, Vanish Talwar, Parthasarathy Ranganathan, GViM: GPU-accelerated virtual machines ,Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing, pp.17-24, 2009

[4] Lin Shi; Hao Chen; Jianhua Sun; "vCUDA: GPU accelerated high performance computing in virtual machines," Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on , vol., no., pp.1-11, 23-29, May 2009

[5] Shucai Xiao, Balaji, P., Qian Zhu, Thankur, R., Coghlan, S., Heshan Lin, Gaojin Wen, Jue Hong, Wu-chun Feng; “VOCL: An optimized environment for transparent virtualization of graphics processing units”, In proc. on Innovative Parallel Computing, pp.1-12 2012