KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.8 pp.245~252 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.8.245

Efficient Workload Distribution of Photomosaic Using OpenCL into a Heterogeneous Computing Environment

Heegon Kim[†] · Jaewon Sa[††] · Dongwhee Choi[†††] · Haelyeon Kim[†††] · Sungju Lee[††††] · Yongwha Chung[†††††] · Daihee Park[††††]

ABSTRACT

Recently, parallel processing methods with accelerator have been introduced into a high performance computing and a mobile computing. The photomosaic application can be parallelized by using inherent data parallelism and accelerator. In this paper, we propose a way to distribute the workload of the photomosaic application into a CPU and GPU heterogeneous computing environment. That is, the photomosaic application is parallelized using both CPU and GPU resource with the asynchronous mode of OpenCL, and then the optimal workload distribution rate is estimated by measuring the execution time with CPU-only and GPU-only distribution rates. The proposed approach is simple but very effective, and can be applied to parallelize other applications on a CPU and GPU heterogeneous computing environment. Based on the experimental results, we confirm that the performance is improved by 141% into a heterogeneous computing environment with the optimal workload distribution compared with using GPU-only method.

Keywords : Heterogeneous Computing, OpenCL, Photomosaic

이기종 컴퓨팅 환경에서 OpenCL을 사용한 포토모자이크 응용의 효율적인 작업부하 분배

김 희 곤[†] · 사 재 원[††] · 최 동 휘[†††] · 김 혜 련[†††] · 이 성 주[††††] · 정 용 화[†††††] · 박 대 희[††††]

요 약

최근 고성능 컴퓨팅과 모바일 컴퓨팅에서 성능가속기를 사용하는 병렬처리 방법들이 소개되어왔다. 포토모자이크 응용은 내재된 데이터 병렬성을 활용하고 성능가속기를 사용하여 병렬처리가 가능하다. 본 논문에서는 CPU와 GPU로 구성된 이기종 컴퓨팅 환경에서 포토모자이크 수행 시 작업부하 분배 방법을 제안한다. 즉, 포토모자이크 응용을 비동기 방식으로 병렬화하여 CPU와 GPU 자원을 동시에 활용하고, 각 처리기에 할당할 최적의 작업부하량을 예측하기 위해 CPU-only와 GPU-only 작업 분배 환경에서 수행시간을 측정한다. 제안 방법은 간단하지만 매우 효과적이고, CPU와 GPU로 구성된 이기종 컴퓨팅 환경에서 다른 응용을 병렬화하 데에도 적용될 수 있다. 실험 결과, 이기종 컴퓨팅 환경에서 최적의 작업 분배량으로 수행한 경우, GPU-only의 방법과 비교하여 141%의 성능이 개선되었음을 확인한다.

키워드 : 이기종 컴퓨팅, OpenCL, 포토모자이크

1. 서 론

최근 고성능 컴퓨팅과 모바일 컴퓨팅 분야에서 성능가속기인 GPU(Graphics Processing Unit), DSP(Digital Signal Processing), FPGA(Field Programmable Gate Array) 등을 사용하여 성능을 개선하는 연구가 활발히 진행되고 있다[1]. 예를 들어, 대표적인 성능가속기인 GPU는 범용 병렬프로그래밍 프레임워크인 CUDA[2] 등을 활용하여 많은 응용에 적용되고 있다[3-6]. 그러나 CUDA의 경우 NVIDIA 계열의 제품에서만 동작하는 단점이 있고, CPU와 같은 마이크로프로세서에서는 지원을 하지 않는다. 이러한 이식성(portability)의 문제점을 해결한 것이 최근 발표된 OpenCL 표준[7]이다. 많은 OpenCL 응용들은 성능가속기를 대상으로 병렬처리를 하고 있고, 성능가속기 중에서도 GPU를 사용하여 여러 응

※ This research was supported by BK21 Plus Program.
† 준 회 원 : 고려대학교 컴퓨터정보학과 박사과정
†† 준 회 원 : 고려대학교 컴퓨터정보학과 석사과정
††† 준 회 원 : 고려대학교 컴퓨터정보학과 석사
†††† 정 회 원 : 고려대학교 컴퓨터정보학과 교수
††††† 종신회원 : 고려대학교 컴퓨터정보학과 교수
Manuscript Received : April 28, 2015
First Revision : June 22, 2015
Accepted : June 26, 2015
* Corresponding Author : Yongwha Chung(ychungy@korea.ac.kr)

용들을 병렬처리한 연구가 발표되었다[8-9]. 또한, 성능을 보다 향상시키기 위해 OpenCL의 이식성을 이용하여 여러 개의 GPU를 사용한 연구도 발표되었다[10]. 그러나 앞서 말한 기존 연구들의 경우 GPU만을 사용하여 병렬처리하였고 CPU 자원은 고려하지 않는 단점이 있다.

OpenCL은 이기종 컴퓨팅 환경에서 병렬처리를 할 수 있는 프레임워크이기 때문에, CPU와 GPU를 동시에 활용하여 병렬처리할 수 있다. 본 논문에서는 이러한 OpenCL의 특징을 이용하여, 포토모자이크(Photomosaic)[11] 응용의 작업부하를 CPU와 GPU에 분배하여 수행시간을 단축하는 방법을 제안한다. 즉, CPU와 GPU를 동시에 활용하기 위하여 주어진 포토모자이크 응용을 OpenCL의 비동기 방식으로 병렬화한다. 그리고 CPU와 GPU 각각에서 처리기 자체의 차이와 OpenCL 구현의 차이를 확인하기 위하여 비동기 방식으로 구현된 OpenCL 코드를 CPU-only(CPU : GPU=100% : 0%)와 GPU-only(CPU : GPU=0% : 100%) 작업 분배 환경에서 측정함으로써 각 처리기에 할당할 작업부하량을 예측한다. 특히 포토모자이크 응용은 원하는 정확도를 파라미터화 할 수 있는데, 이 파라미터값에 따라 동일한 수행 환경에서도 CPU와 GPU의 상대적인 수행시간이 차이가 날 수 있다. 실험 결과, 어떠한 경우에도 제안 방법을 활용하면 CPU나 GPU만을 이용하는 경우나 CPU와 GPU에 균등하게 작업부하를 배분하는 경우보다 수행시간을 단축할 수 있음을 확인하였다.

본 논문의 구성은 다음과 같다. 2절에서는 이기종 컴퓨팅을 가능하게 하는 OpenCL을 소개하고, 병렬처리 성능을 측정하기 위한 응용인 포토모자이크를 설명한다. 3절에서는 제안하는 부하 분산 방법을 설명하고, 4절에서는 실험을 통하여 제안된 방법의 성능을 확인한다. 마지막 5절에서는 결론으로 논문을 마친다.

2. 배 경

2.1 OpenCL(Open Computing Language)

OpenCL은 개방형 범용 병렬 컴퓨팅 프레임워크이다. 많은 제조사와 소프트웨어 전문가의 견해를 반영하여 Khronos 그룹에서 제정된 OpenCL은 여러 제조사들의 협력을 통해서 등장하였다[7]. OpenCL 구현은 CPU, GPU를 포함하여 DSP와 FPGA에 기반한 플랫폼에서도 수행되기 때문에, 하나의 프로그램을 작성하여 여러 종류의 하드웨어를 사용할 수 있는 장점이 있다. 이것은 응용프로그램 수행 중에 OpenCL 커널 프로그램이 빌드되면서 디바이스에 맞는 이진코드를 생성하기 때문이다. 또한, C99를 기반으로 한 언어인 OpenCL C로 커널 프로그램을 작성하기 때문에 프로그램을 작성하는데 있어서 C 언어와 유사한 특징도 가지고 있다.

Fig. 1은 OpenCL의 플랫폼 모델이다. OpenCL 플랫폼은 한 개의 호스트와 한 개 이상의 디바이스들로 구성되고, OpenCL이 동작하기 위해서는 호스트 메모리에서 커널 메모리로 연산에 필요한 데이터가 복사되어야 한다. 마찬가지로 커널 프로그램 수행이 종료되면 디바이스에서 호스트로 연산된 데이터가 복사되어야 한다. 만약 디바이스를 여러 개 사용하는 멀티디바이스로 프로그램을 작성하게 된다면, 호스트에서 커널로, 커널에서 호스트로 데이터가 복사되는 부분의 데이터 종속성을 정확히 판단하고 프로그램을 작성하여야 한다.

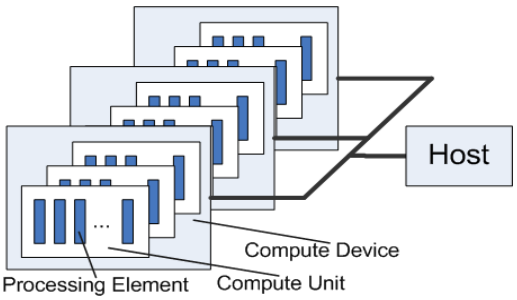


Fig. 1. Platform Model of OpenCL[8]

2.2 포토모자이크(Photomosaic)

포토모자이크[11]는 Photograph와 Mosaic의 합성어로, 하나의 큰 영상을 여러 개의 작은 영상으로 분할하여 각각의 작은 영상으로 대체하는 것이다. 이러한 작은 영상을 "타일"이라고 하고, 일반적으로 크기가 같은 여러 개의 사각형 타일을 이용한다. 또한, 타일의 개수가 많고, 크기가 작을수록 포토모자이크의 영상은 원 영상과 유사해진다.

Fig. 2의 축소된 원본영상과 포토모자이크로 변환된 영상은 유사해보이지만, 포토모자이크의 영상을 확대하여 확인하면 포토모자이크 영상의 한 영역이 여러 개의 작은 타일로 나누어져서 변환된 것을 알 수 있다. 본 논문에서 사용한 포토모자이크 응용은 분할된 원 영상을 유사한 타일로 대체할 때, 반복문을 5회 중첩하여 수행하는 특성을 가지고 있다.

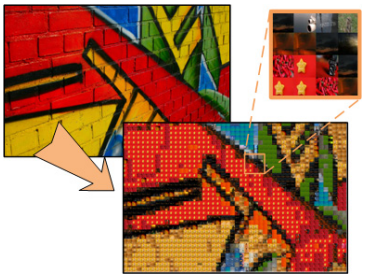


Fig. 2. Result Image of Photomosaic

특히 포토모자이크 응용은 원하는 정확도를 파라미터화(즉, 타일 개수) 할 수 있고, 많은 데이터 병렬성을 포함하여 GPU를 이용한 연구 결과가 발표되고 있다[12-13]. 그러나 이러한 GPU 구현은 CPU를 호스트로만 활용하여, 다중 코어를 장착하여 성능이 지속적으로 개선되는 CPU 자원을 최대한 활용하지 못하는 문제점이 있다. 본 논문에서는 이러한 문제점을 개선하기 위하여 CPU와 GPU의 자원을 동

시에 활용하여 전체 포토모자이크 응용의 수행시간을 단축하는 방법을 제안한다.

3. 제안 방법

3.1 포토모자이크 병렬처리 방법

Fig. 3은 CPU와 GPU에 각각 나누어서 할당된 하나의 입력영상과 연산 후에 합쳐진 결과영상을 보여준다. 본 논문에서는 포토모자이크 응용을 병렬처리하기 위해서 Fig. 3과 같이 주어진 영상을 윗부분과 아랫부분으로 나누어 CPU와 GPU에 각각 할당하고, 각각의 타일 계산은 서로 간의 데이터 종속성이 없다. 따라서 타일의 높이에 원 영상의 높이를 나누어서 CPU와 GPU에 할당할 수 있는 타일의 개수를 정할 수 있다. 예를 들어, 영상의 크기가 20×20이고 타일의 크기가 4×4라고 가정할 때, 원 영상에서 높이를 기준으로 4×4 타일이 5개 들어갈 수 있다. 따라서 원 영상에서 CPU와 GPU에 가능한 분배 방법은 4가지(영상의 높이를 기준으로 CPU : GPU=4 : 16, 8 : 12, 12 : 8, 16 : 4)이다.

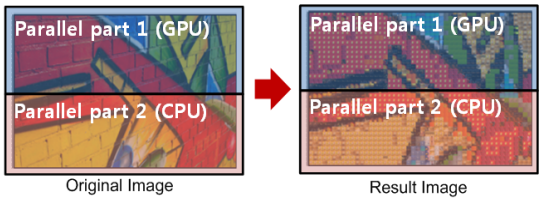


Fig. 3. Parallel Photomosaic Using both CPU and GPU

분할된 영상 중 한 부분은 GPU에서 수행할 수 있고, 나머지 부분은 CPU에서 수행할 수 있다. 두 부분으로 나누어진 원 영상을 CPU와 GPU에서 효율적으로 처리하기 위해서는 CPU와 GPU에 작업량을 적절히 분배해야 할 필요가 있다.

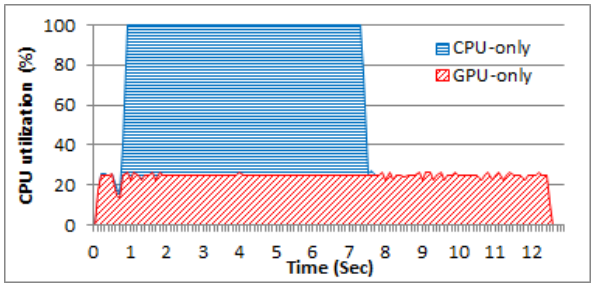


Fig. 4. CPU Utilization with either CPU-only or GPU-only

본 논문에서는 CPU와 GPU를 동시에 이용하기 위해서 OpenCL의 이식성을 이용한다. Fig. 4는 포토모자이크 응용을 GPU만을 사용하여 수행한 GPU-only(CPU : GPU=0% : 100%) 배분에서의 CPU 사용률과 CPU만을 사용하여 연산을 수행한 CPU-only(CPU : GPU=100% : 0%) 배분에서의 CPU 사용률을 각각 보여준다. 기존의 방법[12-13]처럼 OpenCL과 GPU만을 사용하여 포토모자이크 응용을 병렬처리하면 Fig. 4의 GPU-only로 표시된 CPU 활용도를 갖는다. 즉 이와 같이 병렬처리를 수행할 경우, CPU 자원은 호스트로서의 역할만 수행하면서 GPU의 포토모자이크 연산이 종료될 때까지 유휴상태로 대기하게 된다.

따라서 본 논문에서는 CPU와 GPU 수행을 비동기 방식으로 설정하고 유휴상태인 CPU 자원도 포토모자이크 연산에 같이 활용하여 수행시간을 줄이고자 한다. 즉, OpenCL의 비동기 방식을 이용하여 병렬처리를 수행함으로써 CPU와 GPU 자원을 동시에 활용하고, 따라서 기존 방법에서 CPU 자원이 유휴상태로 대기하는 문제를 해결할 수 있다. Fig. 5는 커널 수행을 비동기 방식으로 설정하고 OpenCL을 이용하여 포토모자이크 응용이 CPU와 GPU에서 동시에 수행되는 모습을 보여준다. Fig. 5를 보면, 커널 프로그램은 각각의 디바이스에서 수행되고, GPU를 비동기 방식으로 설정하였기 때문에 호스트 프로그램은 GPU의 연산이 끝날 때까지 대기하지 않고 바로 CPU 커널 프로그램을 동작시킨다.

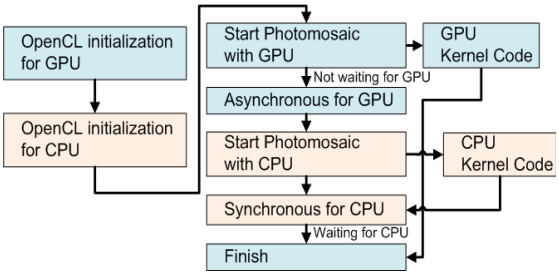


Fig. 5. Parallel Photomosaic Using OpenCL-based Asynchronous Execution with both CPU and GPU

3.2 병렬 포토모자이크 수행시간 분석

CPU와 GPU로 구성된 이기종 컴퓨팅 환경에서 작업부하가 최적으로 분배되었다면 CPU의 연산과 GPU의 연산이 동시에 끝나게 된다. 동시에 연산이 끝난다는 것은 CPU와 GPU 모두 유휴시간이 없었다는 것을 의미하기 때문이다. 그러나 다중코어의 복잡한 구조를 갖는 각각의 처리기에서 해당 응용프로그램을 수행할 때 소요되는 수행시간을 해석적으로 모델링하는 것은 매우 어려운 일이다. 그렇기 때문에, 효율적인 작업 분배량을 정하기 위해서 포토모자이크 응용의 CPU-only와 GPU-only 수행시간을 먼저 비교하고, 이를 통하여 포토모자이크 응용과 각 처리기의 특성을 확인하고자 한다. 수행시간 특성을 분석하기 위한 실험은 Intel i5-2500 CPU(4-코어), NVIDIA GeForce GTX 560(336-코어)의 플랫폼에서 진행하였고, 실험에 사용한 이미지로는 3072×2048의 크기를 갖는 이미지를 사용하였다. 타일은 64×64의 크기를 갖는 이미지를 사용하였다. 본 논문에서는 **작업부하의 최소 분배량**을 타일의 높이로 정의하였기 때문에, 3072(원 영상의 가로 크기)×64(타일의 세로 크기)인 약 0.2 메가픽셀로 구간간격을 나누어 실험하였고, 이 경우 분배량은 총 31개로 나누어 측정할 수 있었다.

Fig. 6, Fig. 7, Fig. 8은 타일의 개수를 1000개, 100개, 10개로 변경하여 포토모자이크를 OpenCL로 수행하였을 경우

의 수행시간이다. 수행시간의 대략적인 경향을 파악하기 위하여 이미지를 11개의 경우로 나누어 GPU와 CPU에 분배하였고, 각 분배량에 따른 **초기화 시간**, **CPU 수행시간**, **GPU 수행시간**을 보여준다. CPU 또는 GPU를 OpenCL에서 사용하기 위해서는 초기화가 반드시 필요하고, CPU나 GPU의 수행시간이 길어지면 초기화 시간의 상대적인 비율이 작아지지만, 수행시간이 짧게 걸리는 경우 초기화 시간의 상대적인 비율이 커지게 된다.

먼저 타일의 개수가 1000개일 때의 수행시간인 Fig. 6을 보면, CPU에 3.6~4.1 메가픽셀, GPU에 1.9~2.4 메가픽셀로 분배한 구간에서 CPU 수행시간과 GPU 수행시간이 유사함을 볼 수 있다. 초기화 시간의 경우, CPU나 GPU의 수행시간이 크기 때문에 초기화 시간의 상대적인 비율이 매우 작다. GPU 수행시간의 경우, GPU에 배분된 작업부하의 구간별로 유사(예를 들어, GPU에 1.3 메가픽셀을 할당한 경우와 0.2 메가픽셀을 할당한 경우의 수행시간이 유사)한 특징을 볼 수 있다. 이러한 이유는, 336개의 코어를 가지고 있는 GeForce GTX 560 그래픽 카드가 데이터 종속성을 고려하

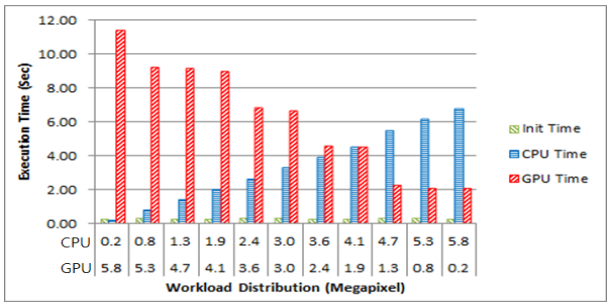


Fig. 6. Execution Time when the Number of Tiles is 1000

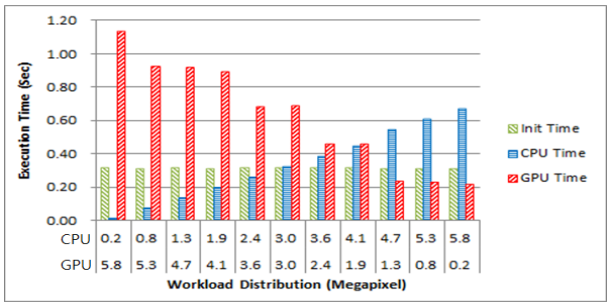


Fig. 7. Execution Time when the Number of Tiles is 100

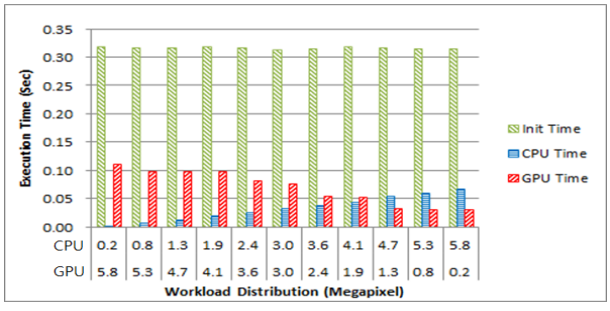


Fig. 8. Execution Time when the Number of Tiles is 10

여 한 번에 처리할 수 있는 픽셀 수는 약 1.3 메가픽셀인 환경에서, 포토모자이크 응용에서 한 개 코어는 한 개의 타일 크기(64×64)를 연산하고, GPU의 경우에는 모든 코어가 연산을 끝낼 때까지 나머지 코어들이 대기하기 때문이다. 따라서 GPU에서 작업 분배량이 다르지만 GPU 수행시간이 같은 구간이 발생하게 된다.

Fig. 7은 포토모자이크의 정확도를 일부 희생하면서 연산량을 감소시킨 경우를 실험하기 위하여 타일의 개수를 100개로 줄였을 경우의 수행시간을 보여준다. OpenCL 초기화 시간의 상대적인 비율이 Fig. 6보다 커진 것을 확인할 수 있고, GPU 수행시간이 GPU에 배분된 작업부하의 구간별로 유사함을 확인하였다.

Fig. 8은 타일의 수를 10개로 하였을 경우의 수행시간을 보여준다. 타일의 개수가 10개일 경우, 총 수행시간에서 OpenCL의 초기화 시간이 상대적으로 매우 높은 비율을 차지하고 있는 것을 확인할 수 있다.

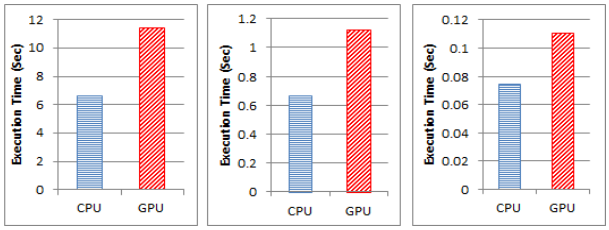
Fig. 6, Fig. 7, Fig. 8을 종합하면 초기화 시간이 일정한 값을 유지하고 있고, 이 초기화 시간은 작업량 분배와 관련이 없는 것을 확인할 수 있다. 특히 타일의 개수가 10개인 Fig. 8을 보면, 초기화 시간이 총 수행시간에서 매우 많은 부분을 차지하는 것을 볼 수 있다. 이 경우, CPU 또는 GPU의 수행시간에 초기화 시간을 더하여 얻은 총 수행시간으로 작업 분배량을 결정한다면 정확한 최적의 작업 분배량을 얻을 수 없게 된다. 이는 작업량 분배에 영향을 미치지 않는 초기화 시간이 총 수행시간의 대부분을 차지하기 때문이다. 따라서 최적의 작업 분배량을 찾기 위해서는 수행시간에서 초기화 시간을 제외하고 분석해야 한다.

3.3 최적의 작업 분배량 결정 방법

CPU와 GPU로 구성된 이기종 컴퓨팅 환경에서 처리기별 최적의 작업 분배량은 처리기의 특성에 따라 달라진다. 따라서 최적의 작업 분배량은 주어진 처리기에서 전수조사를 통해 확인할 수 있으나, 모든 경우의 수행시간을 측정해야 하는 어려움이 있다. 본 논문에서는 주어진 처리기의 수행시간 특성을 한 차례 측정을 통하여 파악하고 최적의 분배량을 결정하는 방법을 제안한다.

OpenCL의 이식성을 활용하여, 주어진 포토모자이크 응용을 비동기 방식으로 구현하고 이를 CPU-only와 GPU-only 작업 분배 환경에서 측정함으로써 각 처리기에 배분할 작업부하량을 대략적으로 예측한다. 이로써 어떠한 이기종 컴퓨팅 환경에서도 동일한 OpenCL 코드를 CPU-only와 GPU-only 작업 분배 환경에서 수행시키면 초기 작업 분배량을 결정할 수 있으며, 보다 정확한 분배량을 도출하기 위하여 필요하다면 유사한 작업 분배량으로 수행시간을 추가 측정하는 현실적인 방법을 취한다.

처리기의 특성을 분석하기 위해서 3.2에서 언급한대로 OpenCL의 초기화 시간을 제외한 CPU와 GPU의 수행시간을 비교한다. Fig. 9는 타일 개수가 각 1000개, 100개, 10개일 때, OpenCL의 초기화 시간을 제외한 CPU-only와 GPU-only 작업 분배 환경에서의 수행시간을 보여준다.



(a) 1000 (b) 100 (c) 10

Fig. 9. Execution Time(except initialization time) Using either CPU-only or GPU-only with Various Tile Sizes

최종 작업 분배량은 Fig. 9와 같이 얻어진 CPU-only와 GPU-only 작업 분배 환경에서의 수행시간 비율을 기준으로 결정할 수 있다. 보다 정확한 분배량을 도출할 필요가 있으면, 초기 작업 분배량을 기준으로 작업부하 최소 분배량(본 논문에서는 3.2에서 언급한대로 0.2 메가픽셀로 설정)을 GPU에서 감하고 CPU에 가하는 경우의 수행시간과, CPU에서 감하고 GPU에 가하는 경우의 수행시간을 추가적으로 측정한다. 이렇게 측정된 총 3번의 수행시간 중 가장 빠른 수행시간을 갖는 경우를 최종 작업 분배량으로 결정한다. 즉, 최적의 작업 분배량을 얻기 위해서는 초기화 시간을 제외한 CPU-only와 GPU-only 작업 분배 환경에서의 수행시간 비율을 기준으로 초기 작업 분배량을 결정하여 수행시간을 측정한다. 결정된 초기 작업 분배량에서 나눌 수 있는 최소 분배량만큼 GPU에 분배량을 더 할당하여 수행시간을 측정하고, GPU에 덜 할당하여 수행시간을 측정한다. 이렇게 측정된 3번의 수행시간 중에서 최소의 수행시간을 가지는 분배량이 최적의 작업 분배량으로 결정된다. 포토모자이크 응용의 경우, 동일한 이기종 컴퓨팅 환경에서 다른 이미지 크기의 최종 작업 분배량을 결정하기 위해서는 최대 3번의 수행시간 측정이 필요하지만, 동일한 이미지 크기의 다른 이미지를 처리하는 경우 기존에 결정한 최종 작업 분배량을 그대로 활용하면 된다.

4. 실 험

4.1 실험 환경

3절에서 언급한 예측 방법을 검증하기 위한 실측 실험은 Intel i5-2500 CPU(4-코어), NVIDIA GeForce GTX 560 (336-코어)의 플랫폼에서 진행하였고, 실험에 사용한 이미지로는 3072×2048의 크기를 갖는 이미지를 사용하고, 타일은 64×64의 크기를 갖는 이미지를 사용하였다. 본 논문에서는 작업부하의 최소 분배량을 타일의 높이로 정의하였기 때문에, 3072(원 영상의 가로 크기)×64(타일의 세로 크기)인 약 0.2 메가픽셀로 구간간격을 나누어 실험하였다. 원 영상에서 세로를 기준으로 32개의 타일이 존재하고, 32개의 타일을 CPU와 GPU에 나누어 분배를 할 경우, 분배 가능한 방법은 총 31가지이다. 최적의 작업 분배량을 검증하기 위하여 31가지의 수행시간을 모두 측정하여 제안 방법을 검증하였다. 또한, 연산량에 따른 성능 향상을 알아보기 위하여 탐색할 타일 후보의 개수를 1000개, 100개, 10개로 변화시켜 실험하였다. 타일의 개수를 줄이면 탐색할 타일의 수가 적어지기 때문에 수행시간에서 이득을 얻을 수 있지만, 탐색하는 타일의 후보가 적어지기 때문에 분할된 원 영상이 적절한 타일로 교체가 될 확률이 낮아지게 되는 단점이 있다.

4.2 최적의 작업 분배 예측/실측 검증

OpenCL에서 초기화 시간을 제외한 CPU와 GPU의 수행시간을 비교해보면, 타일 개수가 변경되어도 CPU 수행시간과 GPU 수행시간은 거의 유사한 것을 확인할 수 있다. 그렇기 때문에, 이를 기반으로 하여 최적의 수행시간을 갖는 분배량을 예측할 수 있다.

타일 개수가 1000개와 100개일 경우에 CPU와 GPU의 수행시간의 비율은 약 37% : 63%이다. 따라서 총 6 메가픽셀을 갖는 3072×2048 이미지의 경우, CPU에 약 3.8 메가픽셀, GPU에 약 2.2 메가픽셀로 분배할 경우 최적의 분배량이라

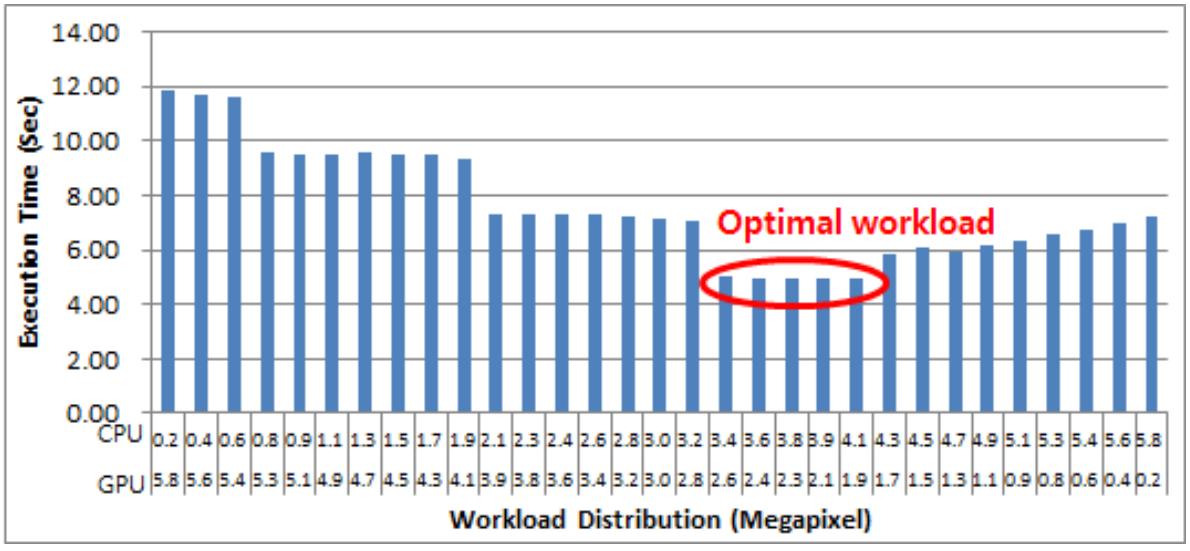


Fig. 10. Execution Time with Different Workload Distribution Using both CPU and GPU when the Number of Tiles is 1000

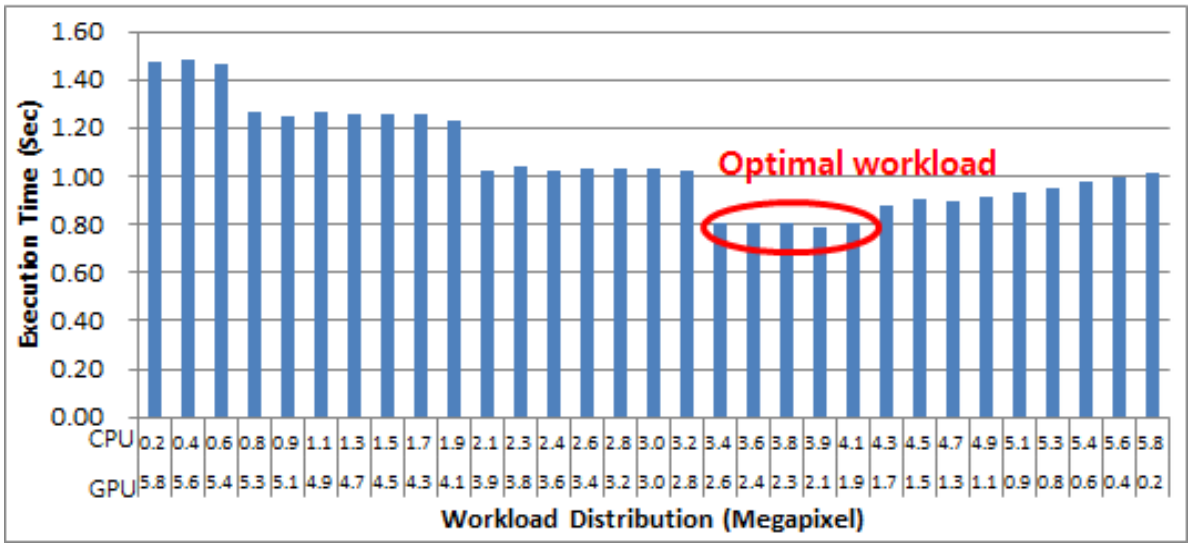


Fig. 11. Execution Time with Different Workload Distribution Using both CPU and GPU when the Number of Tiles is 100

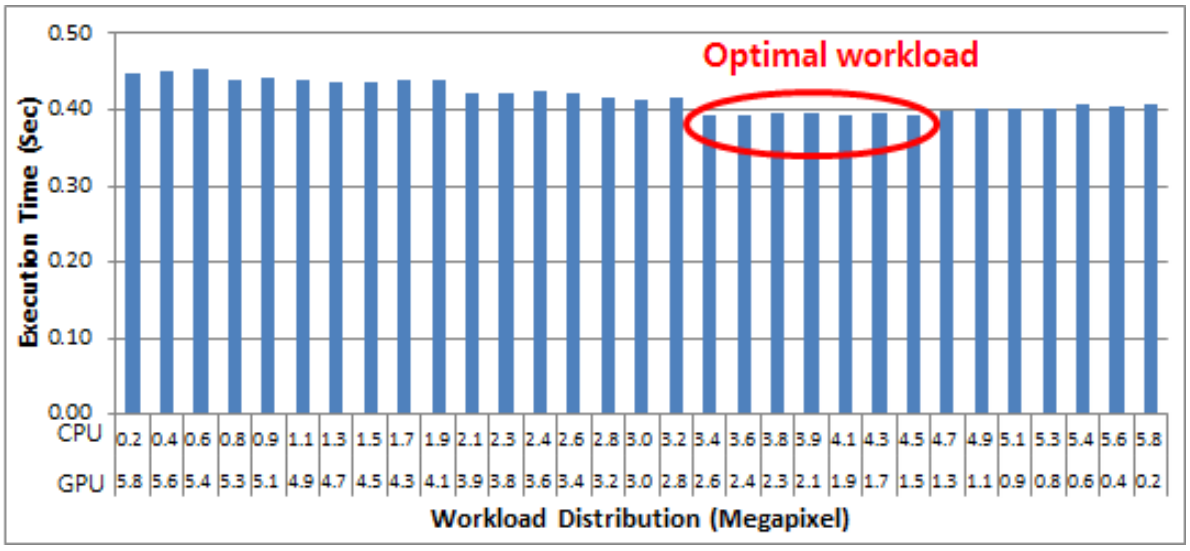


Fig. 12. Execution Time with Different Workload Distribution Using both CPU and GPU when the Number of Tiles is 10

고 예측할 수 있다. 만약 타일 개수가 10개일 경우에는 CPU와 GPU의 수행시간의 비율이 약 40% : 60%이고, CPU와 GPU에 각각 약 3.6 메가픽셀, 약 2.4 메가픽셀로 분배할 수 있다.

Fig. 10은 타일 개수가 1000개일 경우의 분배량에 따른 총 수행시간을 보여준다. 실험 결과, CPU에 약 3.4 메가픽셀, GPU에 약 2.6 메가픽셀을 분배하였을 경우의 구간부터 CPU에 약 4.1 메가픽셀, GPU에 약 1.9 메가픽셀을 분배하였을 경우의 구간까지에서 총 수행시간이 가장 적은 것으로 확인되었다.

Fig. 11은 타일 개수 100개일 경우의 분배량에 따른 총 수행시간을 보여준다. 마찬가지로 CPU에 약 3.4 메가픽셀, GPU에 약 2.6 메가픽셀을 분배하였을 경우부터 CPU에 약 4.1 메가픽셀, GPU에 약 1.9 메가픽셀을 분배하였을 경우의 구간까지에서 수행시간이 가장 적은 것으로 확인되었다.

Fig. 12는 타일 개수 10개일 경우의 분배량에 따른 총 수행시간을 보여준다. 타일 개수 1000개와 100개의 경우와 달리 작업 분배량에 따른 차이가 크지 않으나, CPU에 약 3.4 메가픽셀, GPU에 약 2.6 메가픽셀을 분배하였을 경우부터 CPU에 약 4.5 메가픽셀, GPU에 약 1.5 메가픽셀을 분배하였을 경우의 구간까지에서 수행시간이 가장 적은 것으로 확인되었다.

4.3 실험 결과

3절에서 언급한 부하 분산 예측(CPU-only와 GPU-only 작업 분배 환경에서 측정한 결과를 이용한 초기 작업 분배량)과 실측 검증에서 얻은 결과를 정리하면 Table 1과 같다. 타일 개수 1000개의 경우에서는 실제 측정하여 구한 최적의 분배량은 CPU에서 3.4~4.1 메가픽셀, GPU에서는 1.9~2.6 메가픽셀이고, 제안 방법으로 구한 분배량은 CPU에 3.8 메

가픽셀, GPU에 2.2 메가픽셀이다. 타일 개수 100개의 경우, 개별 수행시간은 1000개 타일의 경우와 다르지만 예측과 최적 분배량은 1000개 타일의 경우와 동일함을 확인하였다(사실 예측과 최적 분배량도 1000개 타일의 경우와 상이할 수 있으나, 0.2 메가픽셀 단위로 작업부하를 설정하면서 발생하는 현상임). 반면 타일의 개수가 10개일 경우에는, 실제 측정하여 구한 최적의 분배량은 CPU에서 3.4~4.5 메가픽셀, GPU에서는 1.5~2.6 메가픽셀이고, 제안 방법으로 구한 분배량은 CPU에 3.6 메가픽셀, GPU에 2.4 메가픽셀이다.

Table 1. Optimal Workload Distribution with Various Tile Sizes

(Megapixel)

		Predicted Rate	Actual Measurement
The number of tiles : 1000	CPU	3.8	3.4~4.1
	GPU	2.2	1.9~2.6
The number of tiles : 100	CPU	3.8	3.4~4.1
	GPU	2.2	1.9~2.6
The number of tiles : 10	CPU	3.6	3.4~4.5
	GPU	2.4	1.5~2.6

종합하면, 연산량이 다른 3가지 모든 경우에서 실제 측정한 최적의 분배량 구간 안에 예측한 작업 분배량이 존재함을 확인하였다. 즉, 작업부하의 최소 분배량 기준으로 모든 경우의 수행시간을 실제 측정 후 최적의 분배량을 얻는 경우는 총 31번의 수행시간 측정이 필요하다. 그러나 제안 방법을 사용하면, CPU-only와 GPU-only 작업 분배 환경에서 측정한 결과를 이용한 초기 작업 분배량이 실측한 최적 분배량 구간 내에 있음을 확인할 수 있었다.

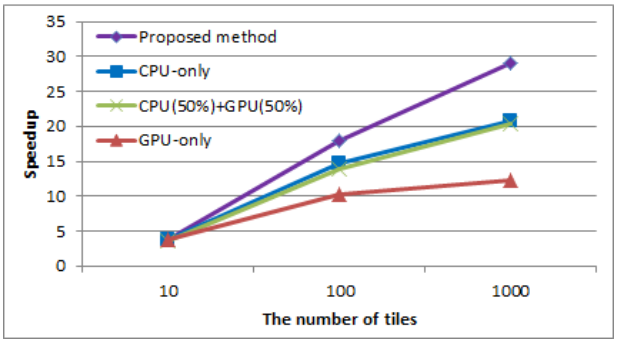


Fig. 13. Speedup with Various Tile Sizes

Fig. 13은 타일 개수가 1000개, 100개, 10개일 경우의 CPU-only, GPU-only, CPU(50%)+GPU(50%)(즉, 작업부하를 CPU와 GPU에 균등배분), 제안 방법의 Speedup을 보여준다. 타일 개수의 모든 경우에서 제안 방법으로 이미지를 분배하였을 경우 CPU-only, GPU-only, CPU(50%)+GPU (50%)보다 성능이 좋아지는 것을 확인할 수 있었다. 타일 개수가 많아질수록 제안 방법의 Speedup이 더 높게 나왔는데, 이것은 타일 개수가 많아짐에 따라서 OpenCL의 초기화 시간이 차지하는 비율이 적어지기 때문이다. 즉, 정확도를 높이기 위해 타일 개수를 1000개로 설정할 경우, CPU-only, GPU-only, CPU(50%)+GPU(50%), 제안 방법의 Speedup은 각각 20, 12, 20, 29이었다(OpenCL의 초기화 시간이 차지하는 비율이 적어지더라도 단순히 작업부하를 CPU와 GPU에 균등배분하면 CPU-only와 GPU-only보다 좋은 성능을 보장하지 않음). 특이한 점은 주어진 실험 환경에서는 GPU보다 CPU만을 이용할 때 더 좋은 성능을 제공하고, CPU에서 병렬처리할 경우 캐시 히트율의 개선으로 Super-speedup을 얻을 수 있었다는 점이다.

5. 결 론

본 논문에서는 이기종 컴퓨팅 환경에서 포토모자이크 응용의 병렬처리를 위하여 OpenCL을 이용한 CPU-GPU 간 작업 분배량을 결정하는 방법을 제안하였다. 또한, 연산량(즉, 타일 개수)의 변화가 CPU와 GPU 간의 작업 분배량에 미치는 영향을 분석하였다.

실험 결과, 타일 개수와 무관하게 제안된 방법으로 계산된 초기 분배량이 실제 측정하여 얻은 최적 분배량 구간 내에 있음을 확인하였다. 또한, 연산량이 늘어날수록 CPU-only, GPU-only, CPU(50%)+GPU(50%) 방법 대비 제안 방법의 Speedup 차이가 증가함을 확인하였다. 즉, 타일 개수가 1000개일 경우에 제안 방법을 사용하면 Speedup은 29를 얻게 되는데, 이것은 CPU-only, GPU-only, CPU(50%)+ GPU (50%)로 병렬처리한 결과보다 각각 45%, 141%, 45% 향상된 성능임을 확인하였다.

References

[1] C. Chun, S. Hong, J. Bae, and M. Lee, "High Performance Computing Using GPU's: with Application to Financial Derivatives Modeling," *The Journal of Korean Institute of Next Generation Computing*, Vol.5, No.1, pp.30-40, 2009.

[2] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," *ACM Queue*, Vol.6, No.2, pp.40-53, 2008.

[3] I. Kim, K. Chang, C. Lee, D. Oh, and W. Ro, "An Efficient H.264 Encoding Process using Multiple GPUs," in *Proceedings of the IEEK Summer Conference*, Jeju, pp.739-740, 2012.

[4] J. Kang, D. Lee, I. Kang, and H. Yu, "A Study on a Declines in Performance by Memory Copy in CUDA," in *Proceedings of the 40th conference of the KIPS*, Jeju, pp.135-138, 2013.

[5] J. Kim, S. Ko, and N. Park, "Implementation of High-Throughput AES Algorithm using CUDA," in *Proceedings of the 2014 Spring Conference of the KIPS*, Suwon, pp.119-120, 2014.

[6] Y. Jeong, N. Tran, and M. Lee, "Parallelization and

Optimization of Boyer-Moore Algorithm on GPU," in *Proceedings of the KCC*, Busan, pp.54-56, 2014.
[7] J. Stone, D Gohara, and G Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *Computing in Science and Engineering*, Vol.12, No.3, pp.66-73, 2010.
[8] Y. Yuan, Z. He, Z. Gong, and W. Qiu, "Acceleration of AES Encryption with OpenCL," in *Proceedings of The 9th Asia JCIS*, Wuhan, pp.64-70, 2014.
[9] M. Bach, V. Lindenstruth, O. Philipsen, and C. Pinke, "Lattice QCD based on OpenCL," *Computer Physics Communications*, Vol.184, No.9, pp.2042-2052, 2013.
[10] V. Demchik and N. Kolomoyets, "QCDGPU: open-source package for Monte Carlo lattice simulations on OpenCL-compatible multi-GPU systems," in *Proceedings of The 3rd HPC-UA*, Kyiv, pp.92-99, 2013.
[11] R. Silvers and M. Hawley, "Photomosaics," New York: Holt Paperbacks, 1997.
[12] D. Kang and K. Yoon, "Photomosaic using a programmable GPU," *Journal of the Korea Computer Graphics Society*, Vol.15, No.1, pp.17-25, 2008.
[13] J. Yang, C. Joo, and K. Oh, "Photo Mosaics using Quad-tree structure on GPU," *Journal of the Korea Computer Graphics Society*, Vol.17, No.1, pp.25-31, 2011.

김 희 곤
e-mail : khg86@korea.ac.kr
2011년 고려대학교 컴퓨터정보학과(학사)
2013년 고려대학교 컴퓨터정보학과(석사)
2013년~현 재 고려대학교 컴퓨터정보학과 박사과정
관심분야 : 병렬처리, 영상처리

사 재 원
e-mail : sjwon92@korea.ac.kr
2015년 고려대학교 컴퓨터정보학과(학사)
2015년~현 재 고려대학교 컴퓨터정보학과 석사과정
관심분야 : 병렬처리, 영상처리

최 동 휘
e-mail : dongwhee@korea.ac.kr
2013년 고려대학교 컴퓨터정보학과(학사)
2015년 고려대학교 컴퓨터정보학과(석사)
2015년~현 재 LG MC사업부 SW연구원
관심분야 : 감시시스템, 병렬처리, 영상처리

김 혜 련
e-mail : mahakhl@korea.ac.kr
2012년 고려대학교 컴퓨터정보학과(학사)
2014년 고려대학교 컴퓨터정보학과(석사)
2014년~현 재 네이버 SW개발부 연구원
관심분야 : 감시시스템, 병렬처리, 영상처리

이 성 주
e-mail : peacfeel@korea.ac.kr
2006년 고려대학교 전산학과(학사)
2008년 고려대학교 전산학과(석사)
2013년 고려대학교 전산학과(박사)
2013년~2015년 고려대학교 컴퓨터정보학과 연구교수
2015년~현 재 고려대학교 컴퓨터정보학과 교수
관심분야 : 멀티코어, 에너지 효율성, 정보보호

정 용 화
e-mail : ychungy@korea.ac.kr
1984년 한양대학교 전자통신공학과(학사)
1986년 한양대학교 전자통신공학과(석사)
1997년 미국 Univ. of Southern California (박사)
1986년~2003년 한국전자통신연구원 생체인식기술연구팀장
2003년~현 재 고려대학교 컴퓨터정보학과 교수
관심분야 : 병렬처리, 영상처리, 축산 IT

박 대 희
e-mail : dhpark@korea.ac.kr
1982년 고려대학교 수학과(학사)
1984년 고려대학교 수학과(석사)
1989년 플로리다 주립대학 전산학과(석사)
1992년 플로리다 주립대학 전산학과(박사)
1993년~현 재 고려대학교 컴퓨터정보학과 교수
관심분야 : 데이터마이닝, 인공지능, 축산 IT