이기종 환경을 위한 OpenCL 기반의 정렬 벡터라이징 설계 및 구현

황윤철*

요약

지난 몇 년간 산업계에는 여러 코어를 가진 CPU가 나오는 혁신적인 일이 발생했고, 이로 인해 생산되는 전자기기의 성능은 우리가 생각했던 것보다 더욱 고성능화 되었다. 그리고 최근에 등장하고 있는 연산 가능한 전자기기들은 대부분 제어용 프로세스와 복수의 연산용 프로세서로 구성된 이기종 환경으로 구성되어 있다. 따라서 이기종 환경을 보다 효율적으로 사용하여 처리 능력을 향상시키기 위해서는 그에 맞는 병렬처리 프로그램이 필요하다. 그래서 본 논문에서는 기존의 순차적 정렬 프로그램을 병렬 처리 프로그램으로 변환하여 처리 효율을 향상시킨 병렬처리 프로토타입을 정의하고 구현하였다. 구현에는 병렬 처리 프로그램의 산업계 표준인 OpenCL 프레임워크를 사용하였다. 그리고 제안한 기법을 순차 처리 방법과 비교 평가한 결과 처리 효율이 30%~50%가 향상되었음을 입증하였다.

Design and Implementation of the Sort Vectorizing based on OpenCL for Heterogeneous Environment

Yoon-Cheol Hwang*

ABSTRACT

Over the past few years, the industry has come out a CPU with multiple cores, and innovative happens, resulting in the production of electronic equipment performance more than we thought was high performance. and, Recently electronic devices capable of operation are mostly in a heterogeneous environment that consists of control processes and multiple processors for the operation. In order to use more efficiently a heterogeneous environment and to improve the processing capability, Program is needed to meet thereto parallel processing. Thus, in this paper, defined and implemented parallel processing prototype which is improved processing efficiency by converting existing sequential sorting program into parallel processing program. The OpenCL framework, industry standard of parallel processing program, was used for implementation. Moreover, a result of comparing the sequential method and proposed method proved that processing efficiency was improved 30% to 50%.

Key Words : Heterogeneous, OpenCL, Vectorizing, Sort, Parallel Processing

* 배재대학교 주시경대학 교양교육부 (✉ dolpin98@nate.com)
· 제1저자(First Author) : 황윤철 · 교신저자(Correspondent Author) : 황윤철
· 접수일(2013년 10월 28일), 수정일(1차 : 2013년 11월 29일), 게재확정일(2013년 12월 12일)

Ⅰ. 서 론

지난 몇 년간 산업계에는 여러 코어를 가진 CPU가 나오는 혁신적인 일이 발생했고, 이로 인해 생산되는 전자기기의 성능은 우리가 생각했던 것보다 더욱 고성능화 되었다. 특히 GPU는 단순히 특화된 그래픽 프로세스가 아니라 고성능 계산 엔진으로 변모하게 되었다[1]. 그리고 표준적인 컴퓨터 구성요소를 CPU와 GPU를 조합한 이종 플랫폼이라고 정의하게 되었다. 또한 CPU는 2004년 이후 동작 주파수 향상이 아니라 연산코어 증가라는 형태로 진화했다. 그런데 싱글 코어 CPU에서 만든 기존의 애플리케이션을 멀티코어 CPU에서 동작시킨다고 처리속도가 빨라지지 않았다. 그래서 멀티코어 프로세서의 성능을 충분히 사용하는 소프트웨어의 개발이 절실하게 필요하게 되었고, 이로 인하여 병렬 소프트웨어가 다시금 주목받게 되었다[2].

병렬 컴퓨팅(parallel computing) 또는 병렬 연산은 동시에 많은 계산을 하는 연산의 한 방법이다. 크고 복잡한 문제를 작게 나눠 동시에 병렬적으로 해결하는 데에 주로 사용되며, 병렬 컴퓨팅에는 비트 수준, 명령어 수준, 데이터, 작업 병렬 처리 방식 등과 같이 여러 방법과 종류가 존재하였다[3]. 그러나 기존의 병렬 소프트웨어는 각 벤더별로 독자적으로 이루어졌기 때문에 이종 플랫폼들과 동일 벤더의 서로 다른 세대의 플랫폼들 사이에 호환성이 없어 병렬처리에는 큰 효과를 볼 수 없었다. 따라서 이종 플랫폼이나 동일 벤더의 서로 다른 세대의 플랫폼들에서도 동작할 수 있는 소프트웨어 개발이 절실히 요구되었다. 이런 요구를 충족시켜 주기 위해 개발된 것이 OpenCL 프레임워크이다[4].

따라서 본 논문에서는 다양한 이종 플랫폼들에서 사용이 가능한 OpenCL 플랫폼을 이용하여 기존의 싱글 코어에서 처리되는 소트 프로그램을 병렬로 처리하기 위한 프로토타입을 정의하고 구현하여 병렬처리의 처리 효율이 우수함을 입증한다. 이 프로토타입은 OpenCL 프레임워크의 기본 컴포넌트들을 이용하여 이종 플랫폼들에서도 모두 동작할 수 있도록 구현하고 기존의 순차적인 처리기법보다 처리 성능이 얼마나 향상되었는지를 실제 처리시간을 측정하여 입증한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문과 관련 있는 OpenCL에 대하여 설명하고, 3장에서는 OpenCL 플랫폼을 사용하여 구현한 시스템의 주요 프로토타입을 정의하고, 구현환경과 시스템의 전체적인 동작을 설명한다. 4장에서는 구현된 시스템의 성능을 평가하고, 마지막 5장에서는 3장과 4장의 결과를 토대로 이 논문의 결론을 맺는다.

Ⅱ. OpenCL

OpenCL은 2008년 Khronos 그룹 안에서 제정된 산업계 표준으로 소프트웨어 제조사, 컴퓨터 시스템 설계 회사, 마이크로프로세서 제작사들이 협력해서 만든 표준 프레임워크로써[5] OpenCL(Open Computing Laguage)을 최대한 간결하게 표현하면 '이기종 병렬 연산기 환경을 위한 병렬처리 프로그래밍 프레임워크'이다[6]. 즉, OpenCL 명세 표준화 작업의 목표는 CPU, GPU, Cell/B.E., DSP 등 여러 종류의 프로세서에 의해 구성된 병렬 연산기 환경에서 공통적으로 이용 가능한 컴퓨팅 환경을 구축하는 것이다. OpenCL은 커널 코드를 작성하기 위한 C99 기반의 언어인 OpenCL C와 플랫폼을 정의하고 제어하기 위한 API를 포함하고 있으며, 작업 기반(task-based) 및 데이터 기반(data-based)의 병렬 컴퓨팅을 제공한다.

지금까지의 연산 프로세서 혹은 연산코어를 다수 지니고 있는 시스템에서 병렬처리를 하려면 이용할 시스템에 맞는 개발환경을 선택했다[7]. 그러나 OpenCL은 프로세서나 OS, 메모리 구성과는 관계없

이 모든 병렬 컴퓨팅 시스템을 지원한다는 점이 장점으로 작용한다. 또한 OpenCL C언어는 이름에서 추측하는 것처럼 C언어의 문법과 거의 유사하다. 따라서 새로운 기능이 많지 않으므로 쉽게 배울 수 있으며, OpenCL 지원 플랫폼이라면 API를 공통으로 사용할 수 있기 때문에 플랫폼을 옮길 때마다 전용 API를 배울 필요가 없다는 것이 OpenCL의 가진 또 다른 장점이다.

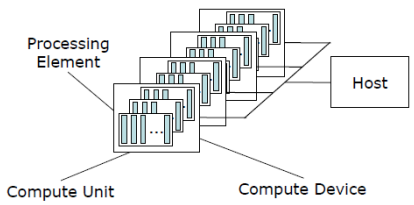

그림 1. OpenCL View
Fig 1. OpenCL View

OpenCL 플랫폼 모델은 하나의 호스트와 한 개 이상의 OpenCL 디바이스로 구성되며, 각 디바이스은 <그림 1>에서와 같이 한 개 이상의 계산 유닛을 가지고 있고 각 계산유닛은 한 개 이상의 PE(Processing Element)를 가지고 있다. OpenCL의 실행은 디바이스에서 실행되는 커널과 호스트에서 실행되는 호스트 프로그램 두 가지로 나뉜다. 커널의 실행은 호스트가 정의한 문맥(context)의 내용에 따라서 진행된다. OpenCL은 커널을 실행하기 전에 호스트가 문맥이라는걸 미리 정의해 주어야 한다. 문맥에는 사용할 디바이스와 실행시킬 커널, 그 커널을 사용하는 프로그램, 커널과 호스트와 디바이스가 사용할 메모리 등을 리소스 형태로 가지고 있다. 리소스 형태의 프로그램은 프로그램 객체라고 부르며 바이너리 소스, executable(exe파일), build option, log, kernel 개수 등등의 정보들을 합한 것을 뜻한다. 메모리도 마찬가지로 메모리 객체라고 부르며 global memory에 있는 메모리의 핸들과 reference count를 뜻한다. 문맥과 동시에 명령 큐라는것도 만들어진다. 이름그대로 명령들이 순서대로 쌓이고 처리되는 공간이다. 호스트는 이곳에 쌓인 명령들을 알맞은 PE로 보내주면 PE들은 받은 명령들을 처리한다[8].

III. OpenCL 기반의 정렬 벡터라이징 구현

본 장에서는 OpenCL 프레임워크를 사용하여 순차적인 프로그램을 벡터라이징하여 병렬 프로그램으로 변환하는 시스템의 주요기능을 중심으로 한 프로토타입을 정의하고 그 동작과정을 설명한다. 그리고 실제로 구현한 환경과 구현 세부사항에 대해 설명한다.

3.1 정렬 벡터라이징 프로토타입

OpenCL에서는 디바이스에서 실행하는 프로그램과 호스트에서 동작하는 프로그램을 명확하게 구분한다. 특히 디바이스에서 실행되는 프로그램을 커널 프로그램이라 부른다. 따라서 본 논문에서는 호스트 프로그램은 OpenCL 런타임 API를 이용해 C로 개발하였고 커널 프로그램은 OpenCL C 언어로 개발하였다.

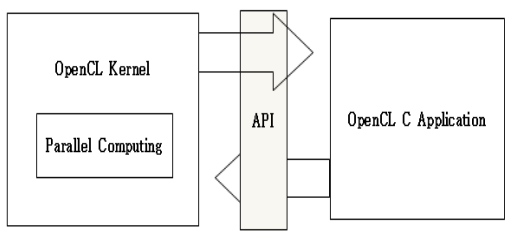

그림 2. 최상위레벨 프로토타입
Fig 2. TopLevel Prototype

<그림 2>는 본 논문에서 제안하는 병렬 프로그램의 최상위레벨의 프로토타입을 표현한 것이다. 어플리케

이션에서는 OpenCL API를 통해서 OpenCL Kernel에 접근하여 병렬 처리를 하여 계산을 한다. 그리고 계산 결과를 다시 OpenCL API를 통해서 어플리케이션에서 받아서 출력하도록 구현하였다.

여기서는 본 논문 구현의 최상위레벨에서 수행되는 중요부분에 대한 프로토타입을 다음과 같이 기술한다. 모든 단계는 CPU 모드에서 실행되며 커널 실행만 OpenCL 디바이스에서 실행된다.

1) 파일 입력 : 콘솔로부터 입력 파일을 읽어온다.

2) 메모리 객체를 디바이스 글로벌 메모리로 전송 : OpenCL API를 통하여 입력파일, 출력파일, 기타 다른 메모리 버퍼들로 이루어진 모든 버퍼 메모리 객체를 디바이스의 글로벌 메모리로 전송한다.

3) 커널 실행: 커널은 OpenCL 디바이스에서 실행하며 여러 서브 연산들이 단일 커널에서 실행되며 글로벌 메모리에 읽고 쓰는 지연시간을 최소화하기 위해 공유 메모리를 사용한다. 공유 메모리는 하나의 메모리 버퍼처럼 사용되며 수행의 중간상태를 저장하기 위해 사용하며 연산이 끝나면 공유 메모리에 있는 내용이 글로벌 메모리로 복사된다.

4) 디바이스의 글로벌 메모리로부터 메모리 객체로 정보 전송 : 커널이 연산을 마치면 디바이스의 글로벌 메모리에는 연산 결과가 저장된다. 이 연산 결과를 OpenCL API를 통하여 OpenCL 디바이스에서 호스트의 RAM이나 다른 저장 장치로 전송한다.

본 논문의 구현에서는 스칼라의 집합으로부터 벡터를 생성하기 위해 int4라는 데이터 타입을 사용하였고, 이럴 경우 4개의 요소를 셀 수 있는 정수형 벡터를 사용할 수 있다. 그러면 입력 파일을 각 요소들이 하나의 행으로 이루어진 int4의 큰 배열로 만들어 병렬처리할 수 있다.

3.2 정렬 벡터라이징 구현

OpenCL 기반의 정렬 벡터라이징은 Intel(R) Core(TM) i5-2400 CPU와 AMD Radeon HD 7640G 디바이스에서 Intel SDK for OpenCL* Application 2013을 이용하여 구현하였다[9][10]. OpenCL로 질의한 두 개의 사용 디바이스 정보는 다음과 같다.

```
Device Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz
CL_DEVICE_TYPE: CL_DEVICE_TYPE_CPU
CL_DEVICE_MAX_COMPUTE_UNITS: 2
CL_DEVICE_MAX_WORK_ITEM_SIZES: 1024 / 1024 / 1024
CL_DEVICE_MAX_WORK_GROUP_SIZE: 1024
CL_DEVICE_MAX_CLOCK_FREQUENCY: 3100 MHz
CL_DEVICE_IMAGE_SUPPORT: 0
CL_DEVICE_GLOBAL_MEM_SIZE: 1024 MByte
CL_DEVICE_LOCAL_MEM_SIZE: 32 KByte
CL_DEVICE_MAX_MEM_ALLOC_SIZE: 512 MByte
CL_DEVICE_QUEUE_PROPERTIES: CL_QUEUE_PROFILING_ENABLE

AMD A8-4500M APU with Radeon(tm) HD Graphics 1.90 GHz
CL_DEVICE_TYPE: CL_DEVICE_TYPE_GPU
CL_DEVICE_MAX_COMPUTE_UNITS: 2
CL_DEVICE_MAX_WORK_ITEM_SIZES: 512 / 512 / 64
CL_DEVICE_MAX_WORK_GROUP_SIZE: 512
CL_DEVICE_MAX_CLOCK_FREQUENCY: 1900 MHz
CL_DEVICE_IMAGE_SUPPORT: 2
CL_DEVICE_GLOBAL_MEM_SIZE: 255 MByte
CL_DEVICE_LOCAL_MEM_SIZE: 16 KByte
CL_DEVICE_QUEUE_PROPERTIES: CL_QUEUE_PROFILING_ENABLE
```

여기서 중요하게 살펴봐야할 명세는 이용 가능한 연산 디바이스 수를 나타내는 CL_DEVICE_MAX_COMPUTE_UNITS과 하나의 변수에 할당할 수 있는 최대 메모리 공간을 나타내는 CL_DEVICE_MAX_MEM_ALLOC_SIZE, 그리고 커널에서 프로그램이 실

행되는 시간과 데이터 전송시간을 측정할 수 있는지 여부를 나타내는 CL_QUEUE_PROFILING_ENABLE 플래그이다. <그림 3>과 <그림4>는 본 논문에서 OpenCL기반으로 버블정렬(Buble Sort)과 선택 정렬을 구현한 커널 프로그램이다.

```
__kernel void BubbleSort(__global int *a,int gonum){
    int i,j;
    int tmp;
    for(i=0;i<gonum-1;i++){
        for(j=0;j<gonum-i-1;j++){
            if(a[j]>a[j+1]){
                tmp=a[j];
                a[j]=a[j+1];
                a[j+1]=tmp;
            }
        }
    }
}
```

그림 3 OpenCL 기반의 버블 정렬
Fig 3. Bubble Sort based on OpenCL

```
__kernel void ParallelSelection(__global int *arr,int n)
{

    int i, j, min;
    int temp;
    for(i = 0; i < n-1; i++)
    {
        min = i;
        for(j=i+1; j < n; j++)
            if(*(arr+j) < *(arr+min)) min = j
            temp=*(arr+i);
        *(arr+i)=*(arr+min);
        *(arr+min)=temp;

    }
}
```

그림 4. OpenCL 기반의 선택 정렬
Fig 4. Selection Sort based on OpenCL

Ⅳ. 성능평가

앞 장에서 구현한 OpenCL기반의 정렬 벡터라이징을 윈도우 콘솔기반과 OpenCL 기반의 데이터 입출력 시간을 가지고 성능 평가를 3회 실시하였다. 그리고 실제 병렬 처리가 이루어지는지를 확인하기 위해 Time() 함수를 사용하여 사용자가 프로그램을 실행시켰을 때의 동작시간을 측정하였다. <그림 5>은 성능 평가를 하기 위해 작성된 호스트 프로그램이다.

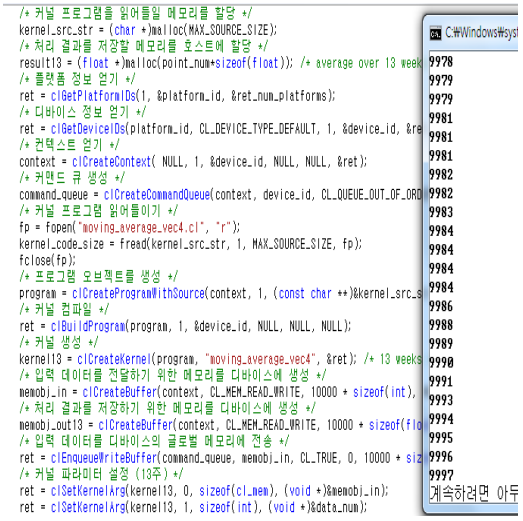

그림 5 OpenCL기반의 호스트 프로그램
Fig 5. Host Program based on OpenCL

배열에 랜덤함수를 이용하여 10000개, 30000개, 50000개의 랜덤 데이터를 입력한 후 선택정렬과 버블정렬에 의해서 오름차순으로 정렬되는 시간을 비교 평가하면 <그림 6>과 같다. 또한 일반 정렬 기법과 정렬 벡터라이징을 사용했을때의 수행 시간을 비교해 보면 <표 1>과 같다.

표 1. 데이터수에 따른 성능 비교
Table 1. Performance Comparison by data number

데이터 수	버블		선택	
	일반	CL	일반	CL
1000	0.000	0.002	0.015	0.001
2000	0.016	0.008	0.016	0.006
4000	0.078	0.032	0.031	0.022
8000	0.265	0.126	0.110	0.090
16000	1.155	0.504	0.437	0.483
32000	4.493	2.016	1.841	1.903
64000	17.706	8.064	7.082	7.713
128000	70.886	32.258	28.080	30.452

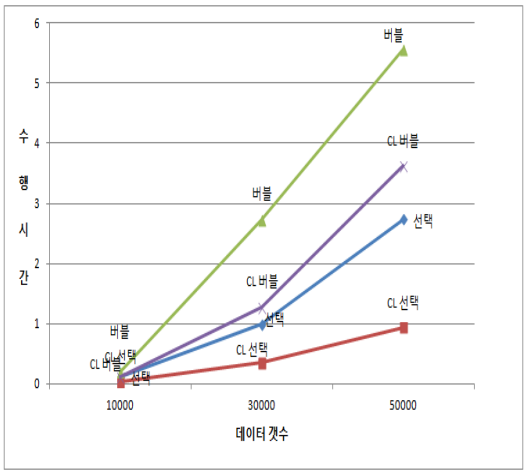

그림 6. 성능 비교 평가
Fig 6. Compare Performance Test

OpencCL을 이용한 병렬 처리 기법을 순차 처리 기법과 비교 평가한 결과 데이터가 적었을 경우 두 기법의 처리효율차가 적었다. 그러나 처리해야 할 데이터가 많아질수록 두 기법의 처리 효율 차이가 커짐을 확인할 수 있었다. 대략적으로 OpencCL을 이용한 병렬 처리 기법이 순차 처리 기법보다 처리 효율이 30%~50%정도 뛰어남을 성능평가를 통해 알 수 있었다.

Ⅴ. 결론

현대에 등장하고 있는 컴퓨터와 스마트 폰 등과 같은 전자기기들은 대부분 제어용 프로세스와 복수의 연산용 프로세서로 구성된 이기종 환경으로 구성되어 있다. 이런 환경에서 기존에 개발된 프로그램을 실행한다고 해서 처리 성능이 향상되지는 않는다. 그 이유는 프로그램이 수행해야 하는 일을 순차적으로 처리하기 때문이다. 따라서 이기종 환경을 보다 효율적으로 사용하여 처리 능력을 향상시키기 위해서는 그에 맞는 병렬처리 프로그램이 필요하다. 그리고 이런 병렬 처리 프로그램은 병렬 연산기를 구성하는 프로세서에 의존하지 않고 모든 벤더들의 제품에서 동작하도록 작성되어야 한다.

그래서 본 논문에서는 병렬처리 프로그램이 갖추어야 할 요건을 충족하고 기존의 순차적 프로그램을 병렬 처리 프로그램으로 변환하기 위해 산업계 표준으로 제정된 OpenCL 프레임워크를 사용하였다. 그리고 이를 기반으로 기존의 순차적인 정렬 프로그램을 병렬처리하기 위한 프로토타입을 정의하고 구현하였다. 구현된 프로토타입은 코드를 먼저 컴파일하여 데이터와 인자를 생성한 후 실행을 위해 커널에 전송하여 처리하는 단계로 동작하게 하였다. 그리고 구현된 병렬 처리 기법을 순차 처리 기법과 비교 평가하여 처리 효율이 30%~50%가 향상되었음을 확인하였다.

본 논문에서 구현된 병렬 처리 방법이 현재 프로세서를 생성하는 모든 벤더에서 똑같이 동작하고 최고의 성능을 얻을수 있는 것은 아니다. 따라서 각 벤더의 플랫폼 환경에서 구현된 프로토타입이 최고의 성능을 발휘할 수 있도록 최적의 하드웨어 선정과 프로그램 개발에 대한 연구가 지속적으로 이루어져야 한다.

참고문헌

[1] Munshi, Aaftab. OpenCL. Parallel Computing on the GPU and CPU, SIGGRAPH, 2008.

[2] Stone, John E.; GOHARA, David; SHI, Guochun. OpenCL: A parallel programming standard for heterogeneous computing systems. Computing in science &engineering, 2010.

[3]Komatsu, Kazuhiko, et al. Evaluating performance and portability of OpenCL programs. In: The fifth international workshop on automatic performance tuning. 2010.

[4] Aaftab Munshi, Benedict R. Gaster, Timothy G. Mattson, James Fung, Dan Ginsburg, OpenCL Programming Guide, Addison-Wesley, USA,

2012.
[5] Khronous Group, OpenCL 2.0 Specification http://www.khronos.org/opencl
[6] KHRONOS OPENCL WORKING GROUP, et al. The Opencl Specification. A. Munshi, Ed, 2008.
[7] NVIDIA, CUDA. API reference manual. 2012.
[8] Fixstars, OpenCL Programming, Hanbit Media, 2012.
[9] Http://software.intel.com/en-us/article/vcsource-tools-opencl-sdk
[10]Http://support.amd.com/us/gpudownload/Pages/index.aspx

저자소개

황윤철(Yoon-Cheol Hwang)

1996년 한남대학교 대학원 전자계산공학과(공학석사)
2008년 충북대학교 대학원 전자계산학과(이학박사)
2008년~현재 배재대학교 교양교육부 외래교수
※관심분야 : 분산 및 병렬처리, 네트워크 및 보안