**첨부 2: 학생 포트폴리오 예시**

**GPGPU 환경에서 효율적인 메모리 접근 패턴을 위한 자료구조 설계**

**과목명: [CSE4152-0X] 고급소프트웨어실습I**

**담당교수: 서강대학교 컴퓨터공학과 임 인 성**

**제출자: 20141494 강동욱**

**개발기간: 2017.12.12 - 2017.12.22**

**최 종 보 고 서**

**프로젝트 제목: GPGPU 환경에서 효율적인 메모리 접근 패턴을 위한 자료구조 설계**

**제출일: 2017. 12. 22**

**제출자: 20141494 강동욱**

**I. 개발 목표**

CPU clock speed 향상에 한계가 다가오면서, multi-core를 이용한 computing 환경으로의 전환이 다가왔고, 자연스럽게 그래픽 프로세서로만 사용하던 many-core architecture를 가지는 GPU를 general purpose computing에 사용하게 되었다.

GPGPU(General-Purpose computing on Graphics Processing Units) computing은 GPU를 사용하여 CPU가 전통적으로 취급했던 응용 프로그램들의 계산을 수행하는 기술이다. 이를 가능하게 한 것은 프로그램 가능한 "단"과 고정도 연산을 그래픽 파이프라인에 연결하는 것으로, 이를 통하여 소프트웨어 개발자들이 그래픽이 아닌 데이터에 스트림 프로세싱을 사용할 수 있게 된다.

이번 프로젝트 에서는 GPGPU 중 CUDA C 언어를 사용한 임의의 크기를 가지는 두 가지 구조의 행렬 원소들의 반복연산을 시행하면서, 메모리 접근 패턴의 여러 요소들에 따른 수행시간 변화를 관찰한다. global memory와 constant memory의 사용, AOS와 SOA로 접근, Block size 변경 등을 수행한다.

**II. 개발 범위 및 내용**

**가. 개발 범위**

**cuda2\_constant.cu**

**나. 개발 내용**

호스트 메모리와 디바이스 메모리의 데이터 교환 방식에 따른 GPU에서 행렬의 곱셈을 구하였을 때 실행시간을 구한다. 또한 데이터 접근방식에 따른 연산시간, Block size 변경에 따른 연산시간 변화를 관찰한다.

**III. 추진 일정 및 개발 방법**

1. **추진 일정**

**2017.12.12-2017.12.12 프로그램작성**

**2017.12.22- 2017.12.22 프로그램 수정 및 보고서작성**

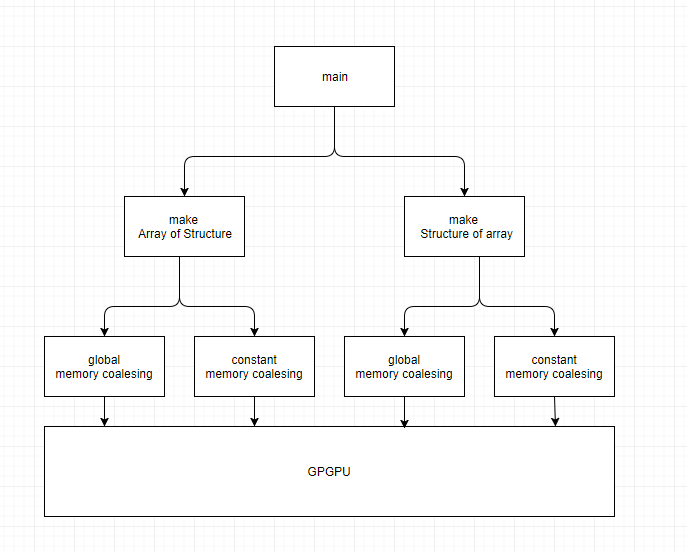
**나. 개발 방법**

**IV. 연구 결과**

- 최종 연구 개발 결과를 자유롭게 기술할 것.

- 다음과 같은 내용을 포함해야함. 그 외의 내용은 자유롭게 기술할 것.

1. **합성 및 분석 내용**:



**전체 소프트웨어 구성도**

**각 부분의 역할 및 사용 기술 및 툴 :**

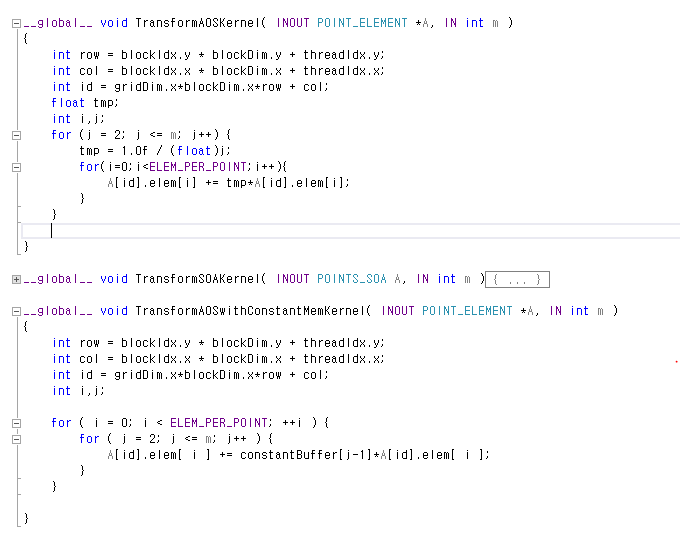
먼저 main함수에서 임의의 데이터를 가지는 AOS 데이터와 SOA 데이터를 생성하는 함수를 호출하여 생성한다. 또한 constant memory coalescing에서 사용할 constant data를 생성한다. 생성된 각각의 데이터에 대해 Global memory coalescing 방식을 사용한 GPGPU연산과 Constant memory coalescing을 사용한 GPGPU 연산을 시행하고, 그 결과를 출력한다.

**사용자 인터페이스 규약**

코드 상단의 #define 부분을 수정하여 연산에 사용될 element의 개수, element per point 등을 수정하여 사용할 수 있다.

**알고리즘 구성 요소에 관한 자세한 분석내용**

AOS 데이터와 SOA 데이터를 생성은 각각 데이터를 가지는 structure들의 array와 array들의 structure를 생성한다. Global memory coalescing과 constant memory coalescing은 계산상에서 각각 아래와 같은 차이를 보인다.



**Constant memory에서는 기존에 계산한 contantBuffuer를 사용한다.**

2. **제작 내용**: 개발 결과. (개발한 함수, 변수에 대한 자세한 내용 작성할 것 )

먼저 구성도상의 데이터를 만들어 주는 두 함수 void generate\_point\_data ( OUT POINT\_ELEMENT \*p\_AOS, OUT POINTS\_SOA \*p\_SOA, IN int n )

생성할 데이터의 포인터 p\_AOS,p\_SOA와 생성할 데이터의 개수 n을 변수로 가진다.

Contant data를 만들어주는 void generate\_constant\_data ( IN int m ) 생성할 데이터의 개수를 m으로 받는다.

각각 구성도 상의 4가지 GPGPU계산 과정을 실행시키는 함수들

void transform\_points\_AOS

void transform\_points\_SOA

void transform\_points\_AOS\_with\_constant

void transform\_points\_SOA\_with\_constant

과 실제 GPGPU연산에서 분산처리할 kenel함수들

void TransformAOSKernel

void TransformSOAKernel

void TransformAOSwithConstantMemKernel

void TransformSOAwithConstantMemKernel

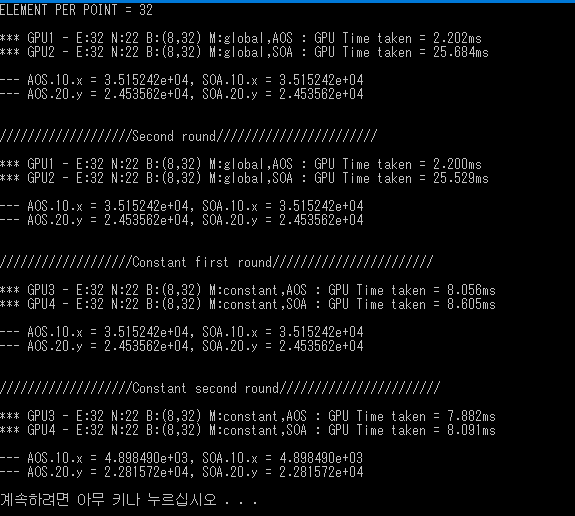
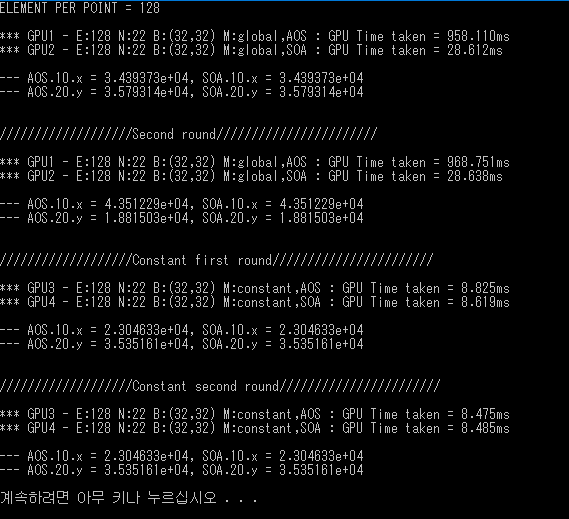
을 개발하였다.

각각의 함수는 AOS, SOA 데이터의 포인터, 데이터의 개수/포인트당 데이터의 개수,cutoff를 parameter로 받는다.

자세한 구현방법은 코드를 첨부하였다.

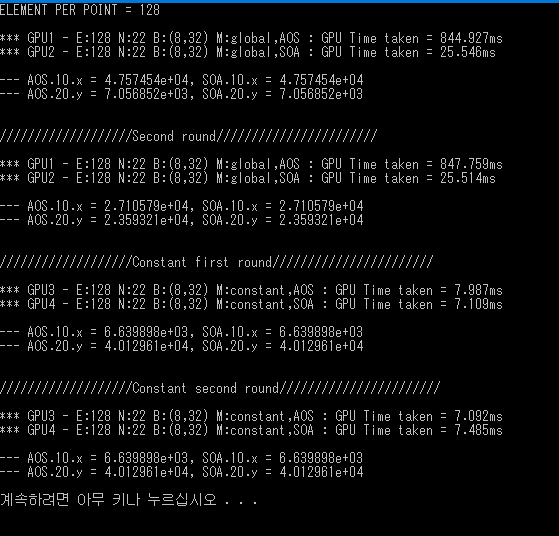
**3. 시험 내용: 자신들이 개발한 소프트웨어를 자체 평가하기 위한 기준 및 수행한 평가 방법, 그리고 그 것에 기반을 둔 평가 결과를 상세히 기술하라. (반드시 평가 점수가 높은 것이 좋은 것이 아니라 자신들이 설정한 방법이 문제점을 찾는데 도움이 되었다는 것을 보이는 것이 더 중요함.)**

Block size와 Elem\_per\_point ,data size를 변경시키며 수행하였다.

**ELEM\_PER\_POINT=32 **

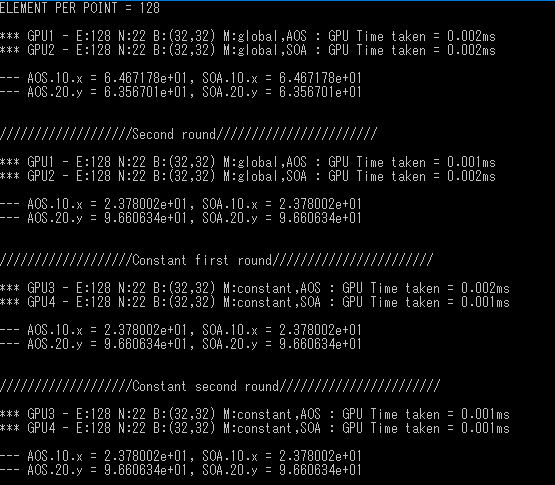
**ELEM\_PER\_POINT = 128 BLOCK\_SIZE=1024**

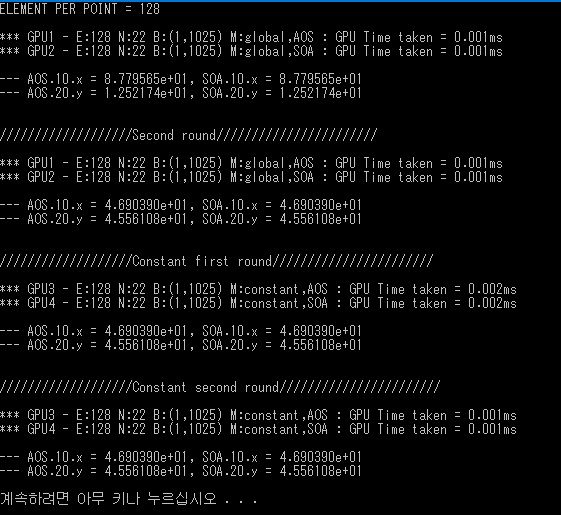
ELEMENT PER POINT 를 증가시켰을 때 global memory를 사용한 GPU time이 극적으로 증가하였다.



BLOCK\_SIZE= 256

블록사이즈가 감소할수록 분산처리의 정도가 높아져 더 빨라짐을 확인 할 수 있다. 하지만 블록사이즈를 너무 작게 하면 한번의 분산처리할 수 있는 블록의 개수의 한계를 초과하여 더 느려지게 된다.





BLOCK\_SIZE= 1025

BLOCK\_SIZE가 1024를 넘어가게 되면 오류가 발생하여 연산이 실행되지 않는다.

4. **평가** 내용: 상기 내용을 바탕으로 자신들이 개발한 소프트웨어의 장단점을 정확히 분석하고 그 내용을 기술할 것. 특히 개발 도중 예상치 않았던 문제가 발생하였다면 상세히 기술하고, 추후 어떠한 방법으로 해결할 수 있을지에 대하여 해결책을 기술할 것. 또한 공학 외적인 부분을 포함하여 경제 요건, 안정성, 신뢰성, 윤리성, 사회적 영향, 실행 가능성 등에 측면에 대해서도 연구 결과와 관련지어 논하라. 본 설계 프로젝트의 경우 **안정성**과 **신뢰성**에 대하여 반드시 기술할 것.

개발한 소프트웨어가 이론과 같이 ELEMENT\_PER\_POINT가 증가할수록 AOS, global방식에서 급격하게 계산시간을 증가시켰으며, 데이터의 개수 2^N이 증가할수록 시간이 증가하며, BLOCK\_SIZE가 일정크기까지 작아질수록 빨라지고 특정크기 이후에는 느려졌으며, constant memory를 사용한 연산에서 global보다 훨씬 빠르고 element\_per\_point에 영향을 적게 받는 것을 확인할 수 있다. 이론과 모든 실행결과가 일치하였으며, block\_size의 제한과 같은 CUDA프로그래밍상의 예외를 제외하고는 안정성을 가졌으며, 신뢰성 또한 보장받음을 확인하였다.

**V. 기타**

- 기타 관련 내용을 기술할 것.

- 다음과 같은 내용을 포함해야함. 그 외의 내용은 자유롭게 기술할 것.

1. **자체 평가**: 본 프로젝트의 수행에 있어 왜 다른 사람보다 더 좋은 점수를 받아야 하는지를 기술하라. 위에서의 (5. **평가** 내용)은 개발한 소프트웨어에 대한 자체 기준을 통한 평가이고, **여기서의 자체 평가**는 프로젝트 수행 전반에 걸쳐 얼마나 창의적으로 목표 및 방법을 설정하고, 수행 중 발생한 문제를 얼마나 현명하게 해결 했는지, 얼마나 독창적인 아이디어가 들어가 있는지, 자신의 결과가 얼마나 우수한지 등의 프로젝트 수행 자체에 대한 평가임. 간략한 보고서 작성도 중요한 평가 척도임.

모든 프로그래밍상 계산되는 변수들을 #define상에서 쉽게 조정할 수 있도록 코딩하였기 때문에 확장성과 유연성이 상대적으로 뛰어나다. 출력되는 N 또한 #define N\_ELEM\_LOG 으로 작성하였으며, 이 N을 수정하면 #define N\_ELEMS도 (1 << N\_ELEM\_LOG)이기 때문에 변경된다.

2. 기타 본 설계 프로젝트를 수행하면서 **느낀 점**을 요약하여 기술하라. 그 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

CUDA프로그래밍의 BLOCK\_SIZE 제한이나 여러가지 프로그래밍의 제약요소가 최신 그래픽카드의 성능적 한계보다 상당히 적게 잡혀 있는 느낌을 받았다. CUDA 프로그래밍의 언어적 발전이 필요하다고 생각된다.