**Assignment 3**

(GPU 컴퓨팅)

담당교수: 공영호 교수님

학과: 컴퓨터정보공학부

학년: 3

학번: 2018202074

이름: 김상우

#코드 작성

과제를 들어가기 앞서, 과제 조건에 대하여 살펴보았습니다.

* Vertor addition code의 넓이는 WIDTH로 따로 정의되어 코드상에서 바뀔 수 있다.
* 해당 코드는 두 행렬에 대하여 행렬 곱 연산을 한다.
* a,b array는 가로와 세로 길이가 같은 n\*n 행렬로, 결과 또한 n\*n 행렬이다.

즉, WIDTH \* WIDTH의 크기의 행렬 2개를 a,b를 선언하고 gpu환경에서 행렬곱을 실행, 해당 결과를 저장한 C 행렬을 CPU에서 출력해주는 것을 목표로 하면 될 것입니다. 위 정보를 바탕으로 다음과 같이 코드를 진행하였습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Google colab에서 해당 코드를 Assignment2cuda.cu라는 코드로 인식하기 위해 위에 %%writefile Assignment3cuda.cu로 기술하였고 아래 필요한 헤더들을 기술하였다.

해당 행렬의 가로, 세로의 길이를 나타내기 위해 WIDTH를 따로 DEFINE해주었다.

또한 행렬곱을 수행하는 함수를 CPU에서 GPU로 명령하기 위해 \_\_global\_\_을 추가하고, threadidx.x, threadidx.y를 이용, 이차원 배열에서의 가로, 세로 위치를 나타내도록 하였다. 행렬곱의 경우 cij를 구하기 위해선 a의 i번째 행의 n번째 요소와 b의 j번째 열의 n번째 요소를 곱한 것들의 합을 구하면 되므로 위와 같이 표현하였다. 각 요소에 대한 계산을 통해 구해진 값들을 sum 내에 더하고 해당 값을 c에 저장해주며 해당 코드를 구현했다.

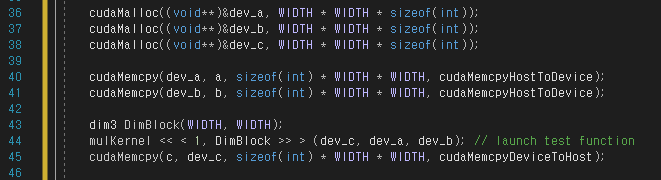
이때 2차원 배열에 대하여 가로 x, 세로 y번째에 위치한다면 y \* width + x를 적용하는 식으로 2차원배열을 1차원 배열로서 표현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Array는 width \* width 행렬이므로 가 5이므로 2차원 배열의 형태로 a,b,c를 선언하고 c에 대해서는 0으로 초기화해주었다. 이후 해당 배열들에 대해 행렬곱이 잘 진행되는 지 보기 위해 위와 같이 구분이 가는 값들로 초기화해주었다.

이후 GPU에서 a, b 그리고 행렬곱의 결과를 담을 c에 대하여 처리하기 위해 WIDTH \* WIDTH 행렬, c를 선언하고 위 배열들의 값을 GPU에서 받거나 저장할 dev\_a, dev\_b, dev\_c 또한 선언해주었다.



cudaMalloc을 이용, dev\_a, dev\_b, dev\_c에 int가 WIDTH \*WIDTH갯수 만큼 들어갈 수 있도록 GPU메모리를 할당시켰다.

이후 cudaMemcpy를 이용, GPU의 메모리를 할당받은 dev\_a, dev\_b에 CPU내의 a,b값들을 복사해준다. 이때 HOST에서 Device로 값을 int가 WIDTH \* WIDTH개 들어가는 만큼 복사하므로 3번째 인사는 sizeof(int) \* WIDTH \* WIDTH, 4번쨰 인자는 cudaMemcpyHostToDevice가 들어가게 된다.

이후 dim3를 이용, 2차원 형렬에 맞는 형태로 GPU에 값을 넣어 동작시킬 수 있도록 하고, 이를 mulkernel의 내부에 넣어 GPU에서 모든 2차원 배열의 요소 대해 실행할 수 있도록 하였다. 위의 mulKernel의 결과로 gpu로부터 받은 값을 받은 dev\_c의 값을 cudaMemcpy를 이용 cpu의 c배열에 복사한다. Device에서 Host로 값을 int가 WIDTH \* WIDTH개 들어가는 만큼 복사하므로 3번째 인사는 sizeof(int) \* WIDTH \* WIDTH, 4번쨰 인자는 cudaMemcpyDeviceToHost가 들어가게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이후 a,b,c의 배열의 각 요소를 출력한다. 예상이 맞다면 난수의 값들인 a[i],b[i]의 처음에 초기화한 값들과 함께 행렬곱이 진행된 값이 c[i][j]를 통해서 출력될 것이다.

이후 dev\_a, dev\_b, dev\_c에 할당된 GPU 메모리를 cudaFree로 해제하고 return 0로 코드를 종료한다.

#Google Colab

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

위에서 작성된 코드를 적용하였다. 맨 위에 적힌 코드에 의해 Assignment3cuda.cu가 생기게 된다.

테이블이(가) 표시된 사진

자동 생성된 설명

이후 nvcc -o를 이용 cuda capability를 sm\_35 버전으로 컴파일 할 수 있도록 실행파일을 생성하며 이때 이름을 Assignment3cuda로 지정해주었다.

이렇게 생긴 실행파일을 !./Assignment3cuda를 통해 실행하자, 기존의 초기화한 a,b의 행렬과 이들에 대해 행렬곱을 실시한 값이 알맞게 출력되는 것을 확인할 수 있다.

다른 값에서도 알맞게 나와야하므로 다른 값 또한 넣어보았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3x3행렬과 더불어 내부의 값을 간단하게 바꿔주었다.

테이블이(가) 표시된 사진

자동 생성된 설명

이 또한 알맞은 값으로 반영되는 것을 확인할 수 있었다.

#고찰

GPU 컴퓨팅 Assignment3을 진행하면서 google colab에서 Matrix Multiplication을 진행해볼 수 있었습니다. 이러한 과정을 통해 GPU를 가진 가장 큰 이점인 병렬에 대하여 2차원 배열에 대해 어떻게 적용하는지 알 수 있었습니다. 특히 dim3를 통해 범위를 확장하여 함으로서 더 넓은 범위에 대하여 GPU를 다룰 수 있게 되었습니다. 특히 2차원 배열을 1차원 적으로 표현하는 방법을 배우며 다양한 경우에 대해 유동적으로 코드나 시스템을 바꾸며 방법을 찾아야 한다는 필요성을 알 수 있었습니다.