GPU 컴퓨팅 Assignment 5

이름 : 이준휘

학번 : 2018202046

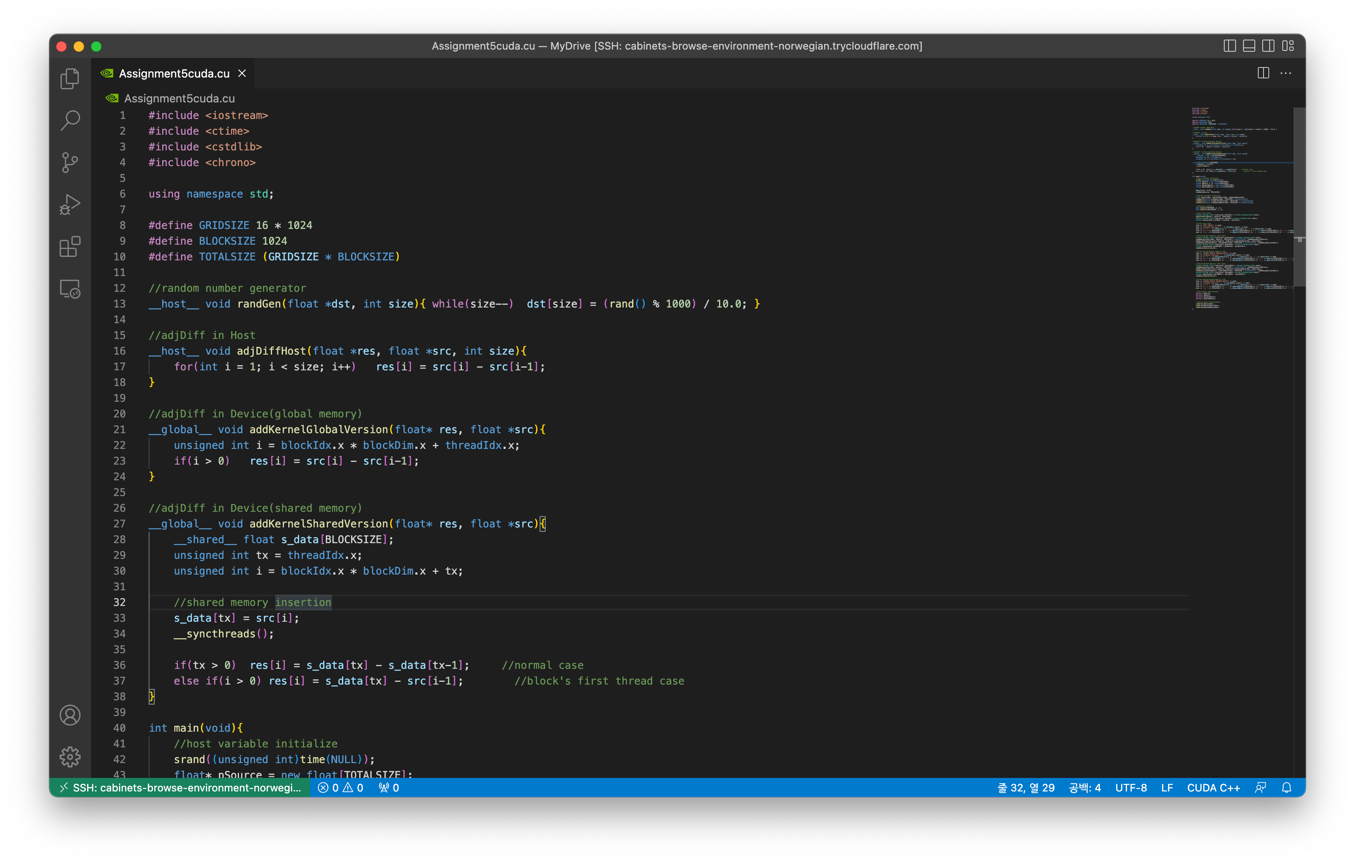
교수 : 공영호 교수님

강의 시간 : 월 수

1. Introduction

해당 과제는 다음 조건에 맞는 코드를 구현한다. 하나의 array에서 adjust difference 연산을 수행한다. 해당 연산은 하나의 index에 위치한 값에서 이전 위치의 값을 뺀 결과를 새로운 배열의 index 위치에 저장한다. 해당 연산을 host에서 수행하였을 때, device에서 global memory를 사용하였을 때, device에서 shared memory를 사용하였을 때를 나누어 결과를 확인한다.

1. Approach

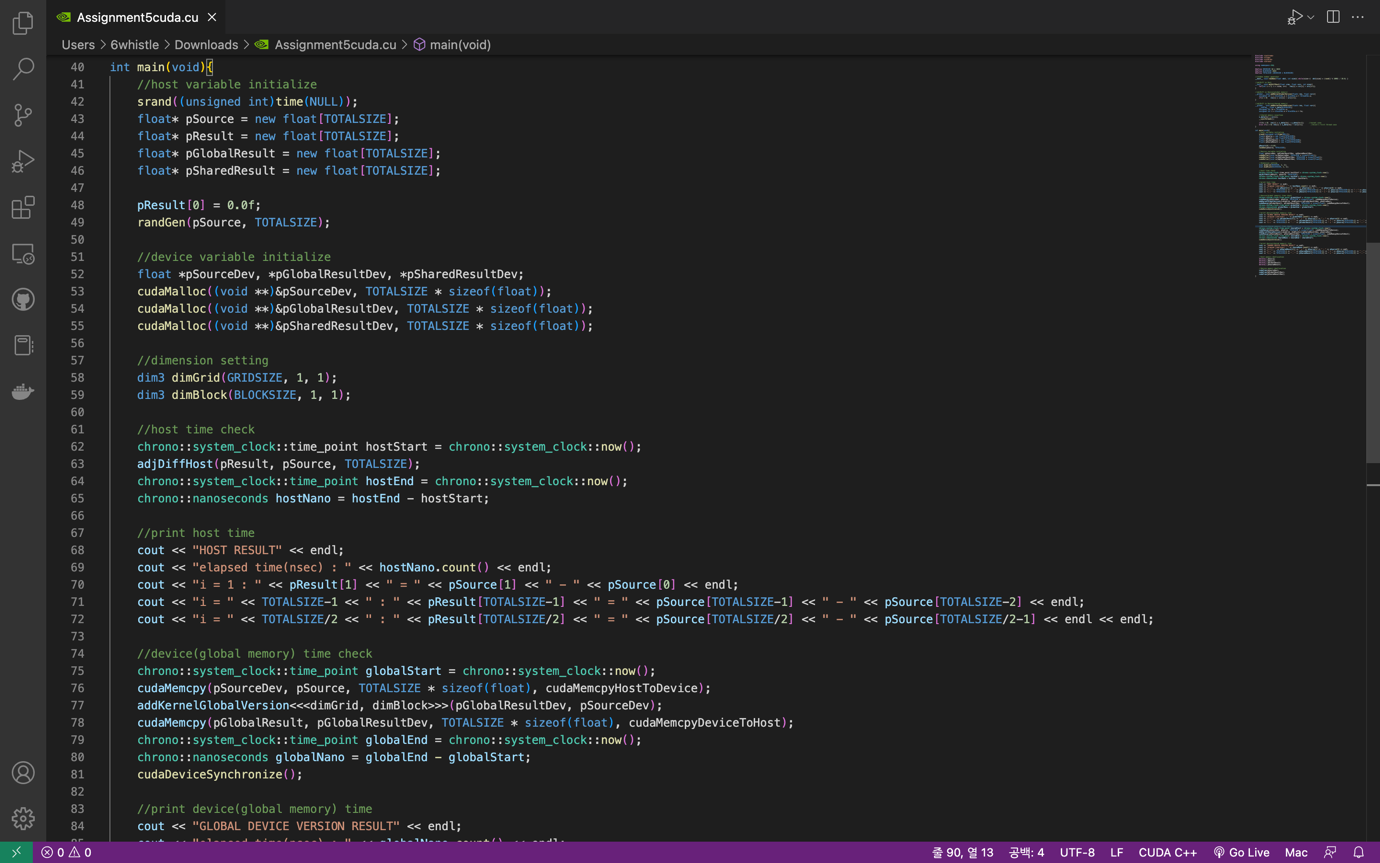


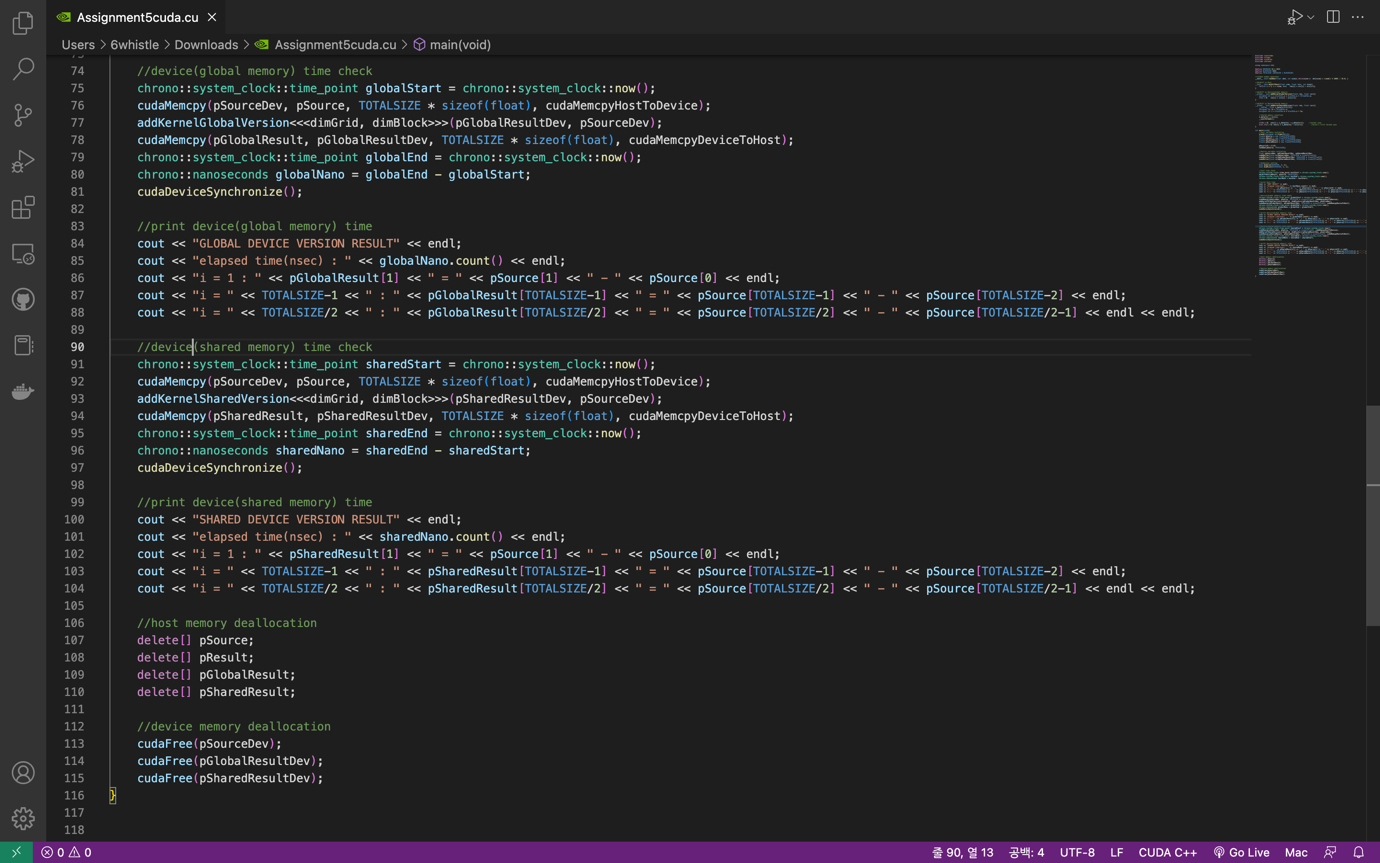
host에서 수행하는 randGen() 함수는 난수를 발생시키는 함수이며, 해당 포인터 위치에 입력한 size만큼 float형 난수를 발생시킨다.

host에서 수행하는 adjDiffHost()함수는 host에서 array 연산을 수행하는 함수다. 함수 내에서는 1번 index부터 순차적으로 올라가며 res의 위치에 src의 i위치에서 i-1위치의 값을 뺀 결과를 저장한다.

addKernelGlobalVersion()함수는 global로 선언되어 있으며 global memory를 활용하여 array 연산을 수행한다. 현재 index에 해당하는 값을 blockIdx.x, blockDim.x, threadIdx.x를 계산하여 얻는다. 해당 i 값을 바탕으로 계산을 한 번 수행하고 함수를 종료한다.

addKernelSharedVersion()함수는 global로 선언되어 있으며 shared memory를 활용하여 array 연산을 수행한다. 우선 \_\_shared\_\_ 인자를 통해 float형 배열 s\_data를 BLOCKSIZE 만큼 선언한다. 그리고 threadIdx.x를 빠르게 가져오기 위해 register에 tx값에 이를 저장한다. index는 blockIdx.x, blockDim.x, tx를 통해 값을 찾는다. 그 후 shared memory를 채워넣기 위해 s\_data[tx]에 src[i]값을 삽입한다. 그 후 다른 thread와 동기화를 맞추기 위해 \_\_syncthreads() 함수를 사용한다. 만약 tx가 0이 아닌 경우에는 연산에 필요한 두 값이 모두 shared memory에 있을 것이기에 shared memory를 활용하며, 만약 tx가 0일 경우 block의 첫 번째 thread이기 때문에 하나의 이전 위치의 데이터는 src에서 가져오는 연산을 수행한다.





Main 함수는 다음과 같다.

우선 rand() 함수에 현재 시간을 통해 시드를 부여한다. 그 후 연산 결과를 저장할 메모리를 동적 할당을 수행한다. 그 다음에는 연산에 사용할 Source에 해당하는 값을 randGen() 함수를 통해 생성한다.

다음으로는 device에서 사용할 메모리 공간을 확보한다. 메모리를 확보한 후에는 dim3 변수에 위에서 Define을 통해 설정한 GRIDSIZE와 BLOCKSIZE 값으로 크기를 결정한다.

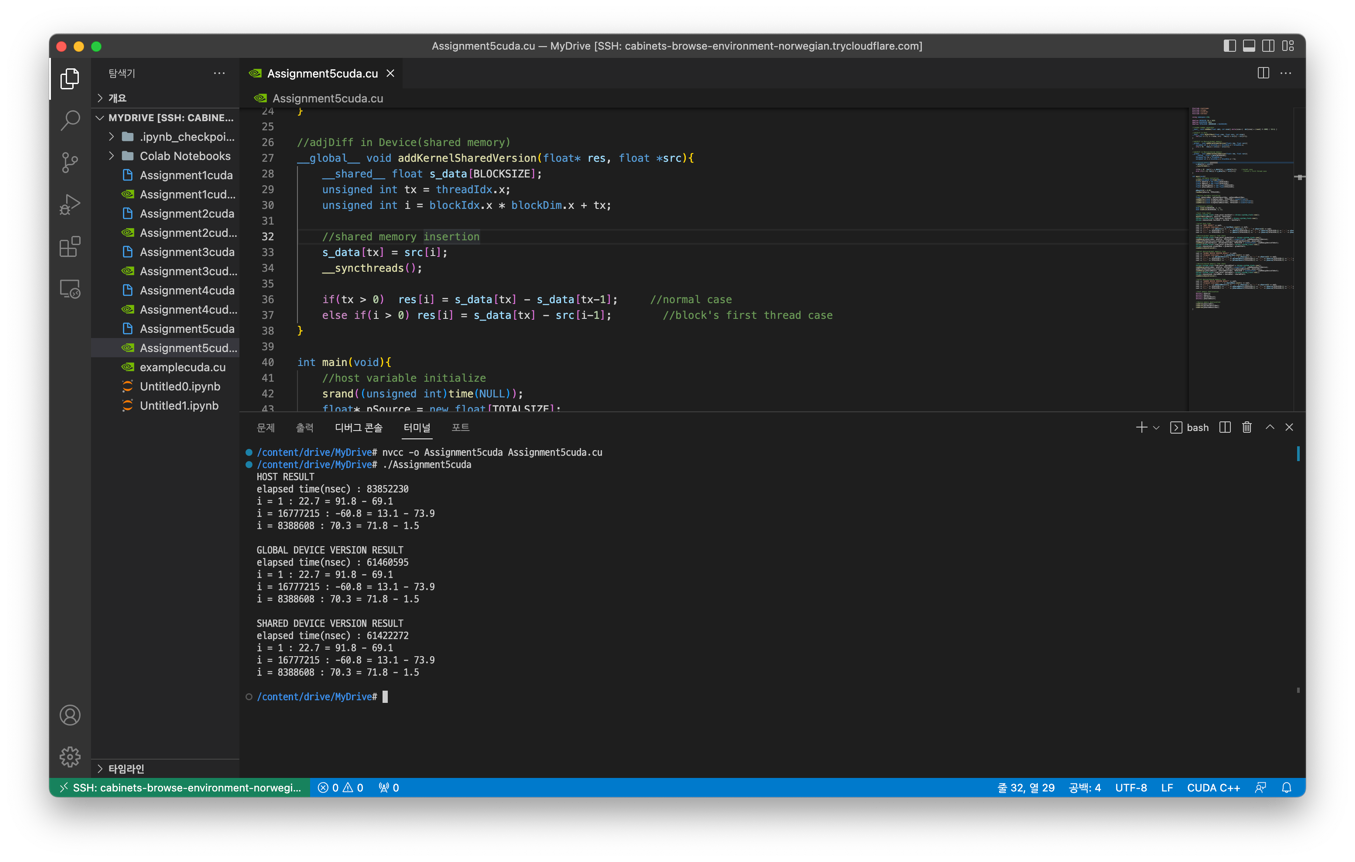
std::chrono 라이브러리를 사용하여 시간을 측정한다. chrono::system::clock의 now 함수를 사용하여 현재 시간을 정확히 측정할 수 있다. 이를 통해 함수를 시작하기 전에 시간을 측정하고 adjDiffHost 함수를 수행한 후에 시간을 측정하여 걸린 시간을 찾는다. 그 후 해당 결과의 일부와 걸린 시간을 출력한다.

device에서 global memory를 사용하였을 때의 시간 측정 구간은 다음과 같다. 우선 pSource의 데이터를 VRAM으로 옮긴다. 그 후 커널을 수행시키고 나온 결과를 pGlobalResult로 옮긴다. 해당 구간까지의 시간을 측정하고 이전과 같이 시간과 결과를 출력한다.

device에서 shared memory를 사용하였을 때는 global memory와 동일하지만 실행하는 커널만 다르게 한다. 시간과 결과를 출력한다.

이후 host와 device의 메모리를 할당 해제한 후 함수를 종료한다.

1. Result



해당 화면은 Colab을 SSH로 연결하여 해당 프로그램을 컴파일, 수행한 모습이다. 위와 같이 정상적으로 컴파일이 되며, 결과가 출력된 것을 확인할 수 있다. 연산의 결과를 보았을 때 host의 시간은 약 83,850,000 nsec가 걸리는 결과가 나왔다. global memory를 사용한 device의 시간은 약 61,460,000 nsec가 걸렸으며 결과는 host와 동일하게 나왔다. shared memory를 사용한 device의 시간은 약 61,420,000 nsec가 걸렸으며 결과는 동일하다. 속도 순으로는 host < global < shared 순으로 빠르게 수행되었지만 global과 shared의 차이는 크지 않았다. 해당 이유는 google의 colab 환경이 GPU를 독점하여 쓰지 않기에 다른 block을 수행하는데 기다리는 시간이 포함되어 elapsed time이 이에 대한 영향을 받는 것으로 추측된다. 만약 이를 실제 desktop 환경에서 수행하게 된다면 의미 있는 결과를 얻을 것이다. 또한 colab 환경은 CPU와 GPU 사이의 간격이 크기에 이에 대한 시간도 길어 host와 device 간의 시간 간격도 커질 것이다.

1. Consideration

해당 과제를 통해 동일한 연산을 다양한 형태(host, device의 global memory, device의 shared memory)의 형태로 수행할 수 있었다. 이를 통해 device에서 사용하는 memory 구조를 직접 이해할 수 있었다. 특히 shared memory를 직접 활용해보는 시간을 가지고 이를 활용할 코드를 직접 구성할 수 있었다. shared memory를 사용할 때 barrier를 사용하는 이유를 추가적으로 공부할 수 있다. 마지막으로 chrono library를 사용하여 시간을 측정하는 방법을 실습할 수 있는 과제였다.

1. Reference

강의자료만을 참고