Heterogeneous Parallel Computing : Class Project Proposal

Sumin Song*

2023.11.10

1 Introduction

1.1 Objective

이번 프로젝트에서는 Project Ideas: 2. CUDA versions for Problem-Based Benchmark Suite (PBBS)를 주제로 진행하고자 한다. 세부 주제로는 선정한 것은 reduceDuplicate이다. 해당 algorithm은 unsigned integer를 담고 있는 array에 중복된 원소를 모두 제거하여 unique element만 남게 만드는 것이다. 예를 들어, [1,2,3,3,4,5,2,6] 이라는 array가 있다고 가정하면, output은 [1,2,3,4,5,6]이 되어야 한다. 해당 알고리즘 설명에 따르면 large size input으로는 1억개의 unsigned integer, small size input으로는 1000만개의 unsigned integer 상황을 두고 있다고 언급한다. 가장 보편화 된 방법으로는 array를 정렬 한 후, 이전 element와 비교하여 같으면 output에 넣지 않고, 같지 않다면 output에 포함하는 방법이 있다. 그렇다면 전체적인 process가 sorting - reduceDuplicates가 될 것이며, sorting 또한 GPU로 성능을 높이고자 하는 항목 중 하나로 overhead가 작지 않을 것으로 예상하여 sorting 없이 시도하고자 한다.

1.2 Related Work

A. Compute Unified Device Architecture (CUDA)
CUDA는 NVIDIA에서 개발한 병렬 컴퓨팅 플랫폼 및 프로그래밍 모델이다. 주로 GPU를 사용하여 고성능 병렬 계산을 수행하는 데 사용된다. CUDA는 GPU를 일반적인 병렬 컴퓨팅 장치로 변환하며, 이를 통해 과학 및 공학 계산, 머신러닝, 그래픽 처리 등의 다양한 응용 프로그램에서 성능 향상을 이끌어낸다. C++와 같은 형식을 채택하여 사용하기 어렵지 않게 설계되었으며, CUDA compiler는 CPU code는 gcc, GPU code는 nvcc를 이용하여 자동적으로 분리하여 컴파일을 이루어준다.
B. Problem-Based Benchmark Suite (PBBS))
PBBS Benchmarks는 병렬 정렬에 중점을 둔 컴퓨터 과학 분야에서 사용되는 벤치마크 모음이다. 이 벤치마크는 다양한 정렬 알고리즘의 성능을 평가하고 비교하는 데 사용된다. PBBS Benchmarks는 병렬 처리 환경에서 정렬 알고리즘의 효율성을 테스트하는 데에 중점을 둠으로써 병렬 알고리즘의 설계 및 성능 향상을 위한 연구를 촉진하는 데 기여하고 있다. PBBS Benchmarks에는 여러 가지 정렬 알고리즘이 포함되어 있으며, 이러한 알고리즘들은 다양한 병렬 컴퓨팅 플랫폼에서 효율적으로 동작하도록 설계되었다.

*Department of Computer Science of Enginerring, POSTECH, South Korea; e-mail: songsm921@postech.ac.kr

2 Main Idea

해당 알고리즘의 목표를 이루기 위해서는 각 element마다 output에 하나만 존재하도록 만들어야 한다. 다시 말해, input array에 있는 각 element의 개수는 중요하지 않고, element가 존재하는지 안하는지의 여부가 중요하다. 따라서, element를 하나의 thread가 다루어 해당 element가 존재하였다는 것을 표시만 해주면 된다. Unsigned int의 경우 0 ∼ 4,294,967,295의 범위를 가지고 있으며 가장 작은 자료형인 char를 사용해도 약 4.3GB의 메모리가 필요하다. 이 메모리 사용량을 줄이는 것이 필요하다고 생각하여 OS File system에서 사용하는 bitmap 개념을 사용하고자 한다. Char는 1byte로 8bit로 표현된다. 즉, char array 한 칸으로 8개의 수 존재 여부를 표현할 수 있으며, 이는 4.3GB 메모리 요구량에서 약 536MB로 줄일 수 있다. 각 thread에서는 몫과 나머지 연산으로 char array의 target index와 bit index를 구할 수 있으며, 해당 자리에 어려운 연산이 아닌 존재 여부 판단을 위해 1을 대입해주는 과정만 필요하다. 이는 서로 다른 thread에서 이전 값이 중요하지 않기 때문에 synchronization 문제에서 자유로울 것으로 예상한다. 결과를 취합할 때는 현재 proposal 단계에서 두가지 방향으로 설정하고 있다. Output의 크기를 unsigned integer만큼 지정하여 해당 value를 index로 하여 취합하는 것과, output size는 input size보다 작거나 같을 것이므로 input size만큼 할당한 후, 하나의 thread가 가질 수 있는 integer의 수가 최대 8개이므로 이점을 고려하여 indexing하는 방법으로 고려하고 있다. 또한, reduction을 적용하여 취합하는 방법도 있으나, 이는 synchronization을 고려해주어야 해 overhead가 발생할 것이므로 차선책으로 둔다. 아래는 대략적인 pseudo code이다.

Algorithm 1 reduceDuplicates

Stage 1
Each thread marks 1 or nothing on char bit array
if element = 12
index = element / 8, position = element % 8
bitarray[index] = bitarray[index] | (1 ≪ position)
Stage 2
Each thread translate one char element
thread can get maximum 8 integer
Stage 3
Merge all results

3 Plan

∼11.18 Implement Stage 1
∼11.25 Implement Stage 2
∼12.02 Implement Stage 3
∼12.14 measure performance and prepare presentation