데이터 수에 따른 CPU와 GPU 기반 병렬 처리의 성능 비교[1]

심건우[1] ○, 이선열[1], 김동영[2], 채흥석[1]
[1]부산대학교 전기컴퓨터공학부
[2]국방과학연구소 해양기술연구원 함정전투체계단
[1]{bkn04073, sun302412, hschae}@pusan.ac.kr, [2]dckyoung@add.re.kr

Parallel Processing Performance Comparison Based on CPU and GPU Depending on the Number of Data

Geonwoo Shim[1] ○, Seonyeol Lee[1], Dongyoung Kim[2], Heungseok Chae[1]
[1]Dept. of Electrical & Computer Engineering, Pusan National University
[2]Agency for Defense Development Maritime Technology Research Institute Naval Combat Systems PMO

요 약

이종 시스템에서의 데이터 처리는 프로그램의 구조와 처리 데이터 수에 따라 다르게 처리되는 것이 효율적이라고 알려져 있다. 단순하고 병렬화 가능한 데이터 처리는 GPU가, 복잡하고 순차적인 데이터 처리는 CPU가 연산하여 보다 더 효율적인 데이터 처리를 할 수 있다. 본 논문에서는 데이터의 수와 프로그램 구조에 따라 CPU 및 GPU연산 처리 속도의 경향을 제시하여 이종 시스템을 효율적으로 사용하기 위한 가이드를 제공한다.

1. 서 론

이종 시스템은 통상 범용 CPU (Central Processing Unit)와 고속의 병렬 계산에 특화된 가속기(accelerator)를 함께 장착한 시스템을 일컫는다. 이종 시스템은 복잡하고 순차적인 작업에 효율적인 범용 CPU와 많은 양의 병렬화 가능한 간단한 반복 작업에 특화된 GPU와 같이 특성이 다른 프로세서를 함께 사용하여, 한 종류의 프로세서만 사용하는 동종(homogeneous) 시스템에 비하여 높은 성능 및 에너지 효율을 달성한다[1]. 과학 계산, 기계학습 등 대규모 데이터와 병렬 연산을 요구하는 응용이 중요해지면서, 이종 시스템을 효율적으로 사용하기 위한 프로그래밍 모델이 요구되고 있다.

GPU는 이종 시스템에서 가속기로 가장 널리 사용된다. GPU는 3D처리를 위해 수많은 객체에 대해 수학적 물리적 계산을 해야 하기 때문에 대규모 파이프라인 방식과 대규모 멀티코어, 매니코어 형태로 발전하였다. 저전력· 고성능 처리를 위해 OpenCL, CUDA, OpenMP 등 GPGPU (General Purpose computing on GPU)를 도입하면서 기존 CPU가 맡았던 응용 프로그램들의 계산도 담당하게 되었다[2-4].

그러나 CPU와 GPGPU가 사용하는 메모리 공간이 다르다. GPGPU를 이용해 연산하려면 CPU와 GPGPU 간 데이터 송수신이 필요하다. 만약, CPU와 GPU 간 데이터 송수신 시간이 연산 처리 시간보다 더 길다면, GPU 기반의 병렬 처리보다 CPU 기반의 병렬 처리가 오히려 효과적일 것이다.

따라서 본 논문에서는 처리 데이터 수에 따라 GPU 기반 병렬 처리와 CPU 기반 병렬 처리의 성능을 비교한다.

2. 연구배경

CPU와 GPGPU가 사용하는 메모리 공간이 다르다. 이는 GPU 기반의 병렬 처리에 CPU와 GPU 간 데이터 송수신이 필요함을 의미한다. 그림 1은 GPU 기반의 데이터 병렬 처리의 절차를 나타낸다. 1) CPU의 메인 메모리에서 GPU 메모리로의 데이터를 복사한다. 2) CPU에서 GPU로 복사된 데이터를 이용하여 병렬 처리를 수행하도록 명령한다. 3) GPU 내부 코어에서 병렬 처리를 수행하여 GPU 메모리에 저장된다. 4) Host에서 결과물 데이터를 GPU에서 메인 메모리로 복사하여 모든 처리가 완료된다.

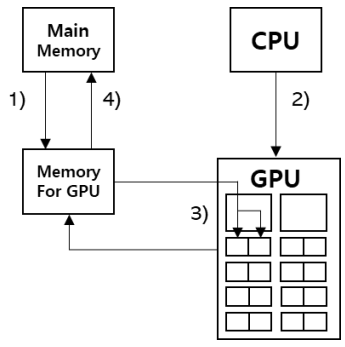

그림 1. GPU 기반 데이터 병렬처리 프로세스

[1] 이 논문은 국방과학연구소의 지원을 받아 수행된 연구 결과입니다. (계약번호: 202107089D9 - 00)

GPU 기반 병렬 처리를 할 때 그림 1의 3) 과정은 GPU 사용의 장점인 많은 코어 수를 활용할 수 있으나 1), 4)과정에서의 CPU와 GPU간 데이터 송수신에 소요되는 시간이 변수로 작용하여 GPU 기반 병렬 처리가 항상 최적의 결과를 보장한다고 할 수 없다. 따라서 본 논문에서는 처리 데이터 수에 따른 성능 비교를 진행하여 CPU와 GPU의 데이터 병렬 처리 성능 경향을 알아내고자 한다.

3. 실험 설계

3.1 실험 목적

본 실험은 데이터 수에 따른 CPU와 GPU 기반 병렬 처리 성능을 비교한다. 실험 결과는 데이터 수에 따른 GPU 기반 병렬 처리와 CPU 기반 병렬 처리의 선택에 대한 가이드라인으로 사용될 수 있다.

3.2 실험 환경

그림 2는 실험 환경을 나타낸다.

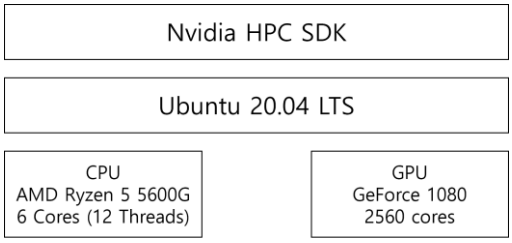

그림 2. 실험 환경

Nvidia HPC SDK에서 지원하는 OpenACC는 병렬 컴퓨팅을 위한 프로그래밍 표준이며 지시어(Directive)를 추가하여 데이터 처리를 수행한다.

3.3 벤치마크 프로그램

1) 반복문

표 1은 두 정수형 배열의 각 인덱스의 합을 for문을 통해 구하는 프로그램이다. 배열의 크기를 1000만부터 1억까지 늘려가며 결과를 확인하였다. #pragma acc parallel loop 지시어를 추가하여 명시적으로 GPU에서 병렬처리를 할 수 있도록 하였다.

표 1 반복문 벤치마크 프로그램

```
#pragma acc parallel loop
for(int i = 0; i < SIZE; i++) {arr3[i] = arr1[i] + arr2[i]};
```

2) 분기문을 포함한 반복문

표 2은 무작위 정수로 초기화 된 두 정수형 배열의 합이 양수일 때, 음수일 때, 0일 때를 계산하여 빈도를 확인하는 프로그램이다. 반복문에 분기문을 추가하여 GPU의 연산에 미치는 영향을 확인한다.

배열의 크기는 1000만부터 시작하며 1억까지 늘려가며 결과를 확인하였다.

표 2 분기문을 포함한 반복문 벤치마크 프로그램

```
#pragma acc parallel loop
for(int i = 0; i < SIZE; i++) {
    if(arr1[i]+ arr2[i] > 0) { arr3[i] = 1; }
    else if(arr1[i] + arr2[i] < 0) { arr3[i] = -1; }
    else { arr3[i] = 0; }
}
```

3) 중첩된 반복문

표 3은 두 배열에 저장된 점들 사이의 유클리드 거리를 계산한다. 점은 Coordinate 클래스로 x축, y축의 정수형 좌표를 가지며, 임의로 초기화 한다. 한 배열에 저장된 점의 수(SIZE)는 500개로 고정하고 다른 배열에 저장된 점의 수(NEW_SIZE)를 1만 개부터 10만 개까지 늘려가며 두 점간 유클리드 거리를 계산한다. 중첩된 반복문을 독립적으로 처리하는 #pragma acc loop independent 지시어를 추가하여 GPU에서 연산하였다.

표 3 중첩된 반복문 벤치마크 프로그램

```
#pragma acc parallel
#pragma acc loop independent
for(int i = 0; i < SIZE; i++) {
    #pragma acc loop independent
    for(int j = 0; j < NEW_SIZE; j++) {
        dist[j] = euclideanDist(x1[i], y1[i], x2[j], y2[j]);
    }
}
```

4. 실험 결과

그림 3, 그림 4, 그리고 그림 5는 반복문, 분기문을 포함한 반목문, 그리고 중첩된 반복문에서 CPU와 GPU 기반 병렬 처리에 대한 실험 결과를 나타낸다. 병렬 처리 결과의 임의성을 고려하여 5회 병렬 처리에 대한 평균을 실행 시간으로 나타낸다.

그림 3과 같이, 반복문에서 CPU 기반의 병렬 처리는 약 4천만개의 데이터까지 GPU 기반의 병렬 처리 보다 더 높은 성능을 보였다.

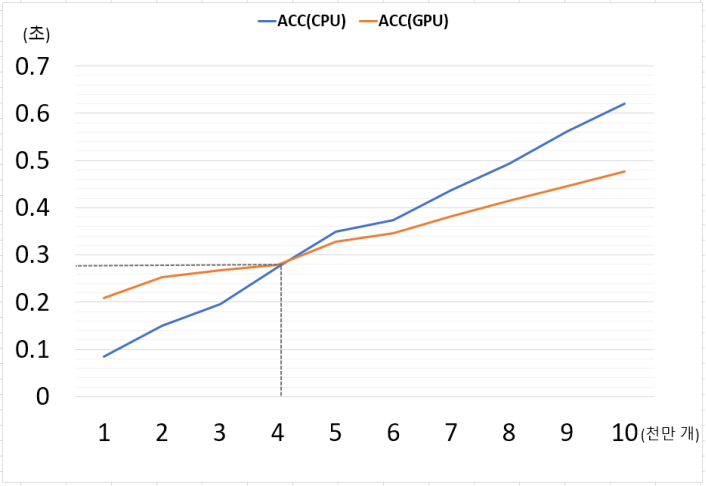

그림 3. 반복문 벤치마크 프로그램 실험결과

그림 4와 같이, 분기문을 포함한 반복문에서 CPU 기반의 병렬 처리는 3천만개의 데이터까지 GPU 기반의

병렬 처리 보다 더 높은 성능을 보였다. 다른 결과에 비해 CPU와 GPU 기반 병렬 처리의 실행 시간 차이가 적은데, 분기문은 GPU 기반 병렬 처리의 성능을 저해하는 요소인 것으로 생각한다.

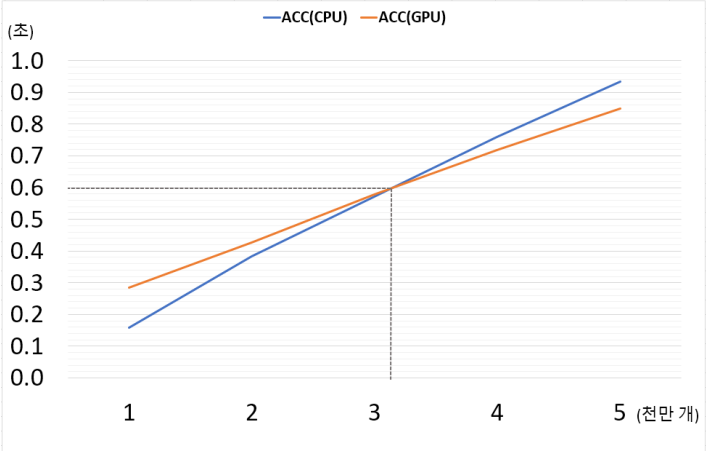

그림 4. 분기문을 포함한 반복문 벤치마크 프로그램 실험결과

그림 5와 같이, 중첩된 반복문에서 CPU 기반의 병렬 처리는 약 2만개의 데이터까지 GPU 기반의 병렬 처리보다 더 높은 성능을 보였다. 다른 벤치마크 프로그램보다 CPU와 GPU 기반 병렬 처리의 실행 시간 차이가 가장 큰데, GPU가 중첩된 반복문을 독립적으로 처리하여 CPU 기반 병렬 처리보다 더 효율적으로 처리할 수 있었던 것으로 생각한다.

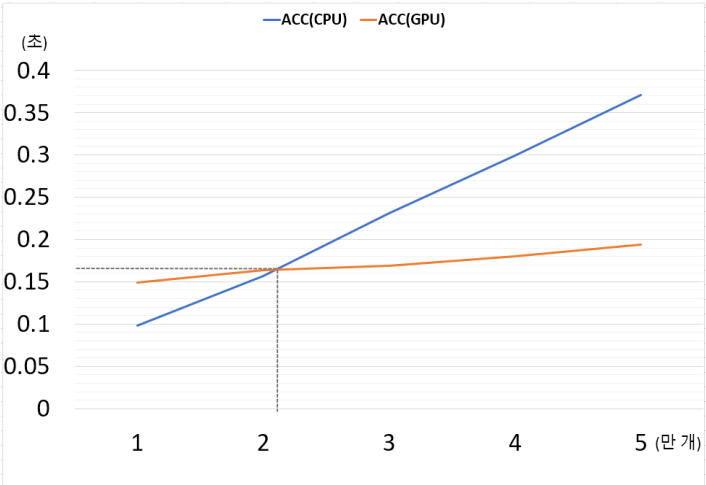

그림 5. 중첩된 반복문 벤치마크 프로그램 실험결과

표 4는 그림 3, 그림 4, 그림 5 그래프의 교차점에서 GPU 사용에 대한 프로파일링 결과를 나타낸다. GPU Activities는 데이터 복사 및 전송에 소요되는 시간을 나타내며 API calls는 OpenACC 지시어에서 CUDA 드라이버나 라이브러리를 호출할 때 사용되는 시간을 의미한다.

교차점 이전의 데이터 크기에서는 CPU와 GPU사이의 데이터 복사 및 전송에 걸리는 시간이 전체 실행시간의 대부분을 차지하며 성능이 저하되는 경향을 보였다. 교차점 이후의 데이터 크기에서는 GPU에서의 데이터 복사 및 전송시간을 포함한 연산 시간이 CPU의 병렬 처리 시간보다 짧아지며 더 높은 성능을 보였다.

표 4 실험 결과에 대한 프로파일링 결과

유형	시간	이름
GPU activities	48.3%	CUDA memcpy
API calls	41.3%	cuDevicePrimaryCtxRetain

5. 결 론

본 논문에서는 데이터 수에 따른 CPU와 GPU 기반 병렬 처리의 성능을 비교하기 위해 세 가지 벤치마크 프로그램을 사용하였다. 단순 반복문, 분기문을 포함한 반복문, 중첩된 반복문 에서의 각각의 성능변화를 확인하였다.

실험 환경과 벤치마크 코드를 제시하여 각 데이터 처리방법에 대한 일반적인 경향을 확인할 수 있었고 이를 통해 데이터 수에 따른 병렬 처리 프레임워크의 선택에 가이드를 제공할 수 있을 것으로 기대한다.

참고문헌

[1] Y. Gao and P. Zhang. “A Survey of Homogeneous and Heterogeneous System Architectures in High Performance Computing,” In Proceedings of 2016 IEEE International Conference on Smart Cloud, pp. 170-175, 2016.

[2] S. Kim and Y. Choi, “Analysis of Human Activity Using Motion Vector and GPU,” J. of the Korea Institute of Electronic Communication Sciences, vol. 9, no. 10, 2014, pp. 1095-1102.

[3] J. Park, “Comparison Speed of Pedestrian Detection with Parallel Processing Graphic Processor and General Purpose Processor,” J. of the Korea Institute of Electronic Communication Sciences, vol. 10, no. 2, 2015, pp. 239-246.

[4] S. Lee and W. Jeong, “Design of the Entropy Processor using the Memory Stream Allocation for the Image Processing,” J. of the Korea Institute of Electronic Communication Sciences, vol. 7, no. 5, 2012, pp. 1017-1026.