고급 소프트웨어 실습 I
CUDA 프로그래밍의 기초 1

20140938 임다은

실습문제 1-1

1. 자신이 사용하는 컴퓨터에 장착된 GPU 의 기종
2. 현재 설치된 CUDA 시스템의 Compute Capability
3. 현재 CUDA Compute Capability 가 제공하는 각종 성능 및 스펙

답:

1. GeForce GT 635
   CUDA 코어: 384, 메모리 데이터 속도: 1800MHz,
2. 3.0
3.

Maximum number of resident grids per device: 16
Maximum number of resident blocks per multiprocessor: 16
Number of 32-bit registers per multiprocessor: 64K
Maximum number of SM per thread block:: 48KB
Maximum dimensionality of grid of thread blocks: 3
Maximum x dimension of a grid of thread blocks: 2^31-1
Maximum dimensionality of thread block: 3
Maximum x or y dimension of a block: 1024
Maximum z dimension of a block: 64
Warp size: 32
Maximum number of resident warp per multiprocessor: 64

실습문제 1-2

(iii) 다음 이 CUDA 프로그램에 대하여 블럭 크기를 다양하게 변화 시켜가면 서 시간을 측정한 후, 그 결과를 보고서에 테이블로 요약하라. 참고로 블 럭의 크기는 와프의 크기인 32 의 배수로 하고, 한 블럭이 가질 수 있는 크기에 어떠한 제한이 있는지 파악하기 위하여 실습 문제 1-1 을 통하여 얻은 정보를 활용하라. 또한 의미있는 결과를 얻기 위하여 전체 데이터의 크 기 (1,048,576)를 자신의 시스템이 허용하는 범위 내에서 최대로 한 후 실험을 진행할 것.

• (이 프로그램은 매우 간단한 프로그램이라 큰 차이가 없을 수 있으나) 블럭 크기에 따른 수행 시간 변화가 있는지 확인하고 자신이 발견한 사항을 보고서에 기술하라.
• 현재 커널 프로그램이 수행해주는 계산을 좀 더 (의미가 있고) 복잡하게 하여 한 쓰레드의 계산 시간을 길게 한 후, 블럭 크기에 따른 수행 시간 변화를 분석하라.

답:
데이터 사이즈를 2^25 로 설정하여 실험하였다. (스크린샷은 첨부자료에 첨부되어있다.)
CPU time average: 278.8823

번호	BlockSize	CPU time	GPU time
1	32	276.22644	33.537025
2	24	274.500366	35.739616
3	16	281.924225	33.115170
4	8	282.878143	45.138783

블락사이즈의 변화에 따른 GPU time 의 변화가 불규칙하다. 32 에서 24 로 블락사이즈가 줄어들어 한 블락에 할당되는 스레드가 많아진 경우 gpu time 이 증가하는데, 16 으로 줄어들면 다시 감소하게 된다. 이는 블락사이즈가 줄어들어 더 많은 스레드를 메모리 접근 시간 때문에 딜레이될 때 계속 돌리기 때문에 utilization 이 높아지는 것으로 보인다. 그러나 블락사이즈가 너무 작아지면 너무 많은 스레드들이 한 블락에 할당되기 때문에 gpu time 이 갑자기 치솟게 되는 것으로 보인다.

실습문제 1-3
이 문제에 대하여 CPU 및 GPU 기반의 코드를 각각 작성하여 그 성능을 비교하여 보자.

(i) 먼저 for 문장을 사용하여 순차적으로 n 번의 행렬-벡터 곱셈을 수행해주는 C 함수를 작성한 후 CPU 수행 시간을 측정하라. 시간 측정 방법은 이전 실습 시간에 사용한 방법을 사용하고, 가급적 정확한 시간 측정을 위하여 여러 번 반복적으로 함수를 수행시킨 후 평균값을 취할 것.

(ii) 다음 이 문제를 해결해주는 CUDA 커널 프로그램을 작성한 후 가급적 정확하게 GPU 수행 시간을 측정하라. 앞의 문제에서와 같이 쓰레드 블럭의 크기를 변화시켜가면서 수행 시간 관점에서 CPU 방법과 비교 분석한 후, 그 결과를 보고서 기술하라.

답:
데이터 사이즈를 2^25 로 설정하여 실험하였다. (스크린샷은 첨부자료에 첨부되어있다.)
CPU time average: 278.8823

번호	BlockSize	CPU time	GPU time
1	32	701.499512	0.293216
2	24	688.022583	0.214304
3	16	708.530672	0.194816

4	8	703.138672	0.206592

이번 실습 문제도 위와 마찬가지로 블락 사이즈가 32 에서 24,16 로 줄어들 때는 gpu time 이 유의미하게 줄어든다. 이는 thread 의 배정을 통해 gpu 의 utilization 이 높아질만큼 블락사이즈가 줄어들었기 때문으로 보인다. 그리고 블락사이즈가 8 로 더 줄어들었을 경우 한 블락에 너무 많은 작업이 할당되어 다시 gpu time 이 늘어나게 되는 것으로 분석된다.

숙제 1

(i) 첫 번째 두 배열 X0 와 X1 에는 자신이 구한 두 개의 실근 x 0 과 x 1 이 저장되어 있는데 반드시 x 0 ≤ x 1 조건을 만족시키도록 저장이 되어야 한다. 다음 두번째 두 배열 FX0 와 FX1 에는 각각 x 0 과 x 1 을 대응하는 이차 방정식에 대입 하여 계산한 함수값을 저장해주어야 한다. 물론 이론적으로는 모두 0 값이 계산되어야 하지만 수치 계산 시 발생하는 계산 오차로 인하여 정확히 0 인 아닌 값이 나올 수 있음을 상기하라.

(ii) 이제 조교가 지정한 N EQUATIONS 값에 대하여 자신의 CUDA 프로그램이 가장 빠른 속도를 보이는 블럭의 크기를 실험적으로 결정한 후 그 값을 보 고서에 기술하라.

번호	BlockSize	Grid	CPU time	GPU time
1	32	2^10/32, 2^10/32	5.446019	2.614304
2	24	2^10/24, 2^10/24	7.537066	2.602848
3	16	2^10/16, 2^10/16	6.731377	2.427904
4	8	2^10/8, 2^10/8	5.819733	2.506784
5	16	2^5/32, 2^15/32	5.721625	2.429856

(iii) 위의 C 함수와 자신의 CUDA 커널 프로그램의 속도를 가급적 정확히 측정하여 그 결과를 보고서에서 비교 분석하라.

이번 실험에서는 블락 사이즈가 32 에서 16 까지 작아질때까지 GPU time 이 계속해서 감소하고 있다. 즉 더 우수한 성능을 보여주고 있다. 이 또한 마찬가지로 하나의 블락에 가장 utilization 이 높아지는만큼 작업이 할당되어 빠른 성능을 보여주는 것으로 생각된다. 그러나 블락 사이즈가 8 로 감소하면 다시 gpu time 이 증가한다. 즉 특정한 블락사이즈까지 성능이 지속적으로 감소하다가 일정 threshold 를 지나면 다시 증가하는 것을 알 수 있다. 또한 그리드를 정사각형이 아니라 직사각형으로 나눠줄 경우 thread 가 적절히 분배되지 않아 그리드를 잘 나눠주었을 때보다 조금 더 시간이 증가함을 알 수 있다. 다만 데이터 사이즈가 충분히 크지 않아 유의미한 차이인지는 알 수 없으므로 데이터 사이즈가 더 큰 숙제 2 에서 확인해보겠다.

숙제 2

(ii) 다음 이에 대응하는 CUDA 프로그램를 작성한 후 다양한 크기의 블럭에 대하여 속도를 측정한 후, CPU 기반 코드에 비해 얼마나 성능이 향상이 되 는지 분석하라.

5 번 평균한 CPU time: 2927.054

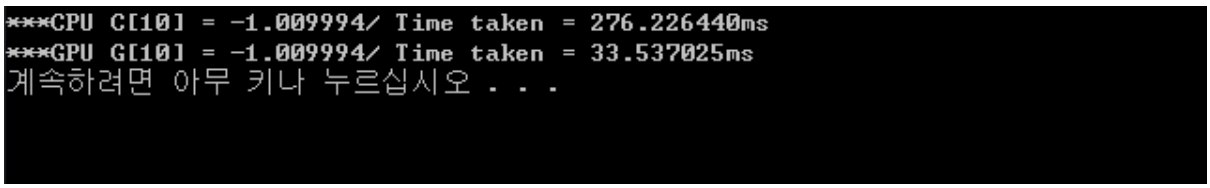
번호	BlockSize	Grid	CPU time	GPU time
1	32	2^13/32, 2^13/32	2911.885254	52.016930
2	24	2^13/24, 2^13/24	2923.379150	55.147839
3	16	2^13/16, 2^13/16	2949.7775888	47.302017
4	8	2^13/8, 2^13/8	2901.184082	70.568733
5	32	2^10/32, 2^16/32	2949.041504	70.829025

(iii) 위의 CPU 코드와 CUDA 코드로 실험한 내용을 자신의 분석 결과와 함께 보고서에 명확히 기술하라.
실습문제와 마찬가지로 Block Size 가 16 까지 작아질수록 gpu time 이 감소한다. 즉 더 높은 성능을 보여주고 있는 것이다. 이 또한 마찬가지로 하나의 블락에 가장 utilization 이 높아지는만큼 작업이 할당되어 빠른 성능을 보여주는 것으로 생각된다. 그러나 16 에서 8 로 변할 때는 시간이 오히려 증가했다. 이는 하나의 블락에 너무 많은 작업이 할당되어 오히려 gpu 의 성능이 떨어지게되는 것을 보여준다. 또한 그리드를 정사각형으로 동일한 작업을 할당해주는 것이 그리드를 직사각형으로 5 번처럼 할당하면 더 많은 시간이 걸리게 된다. 따라서 이차원 배열을 만들 때는 x 와 y 길이가 동일하도록 해야 한다.

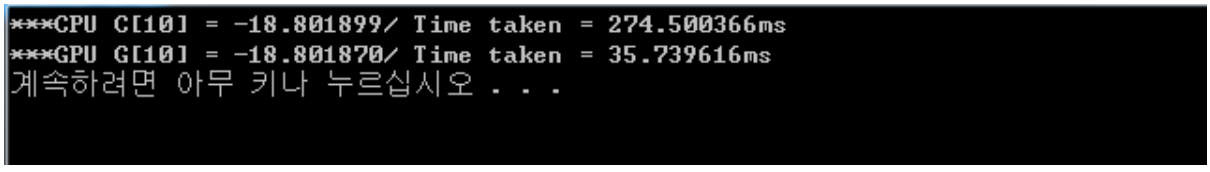
<첨부자료>

실습 1-2

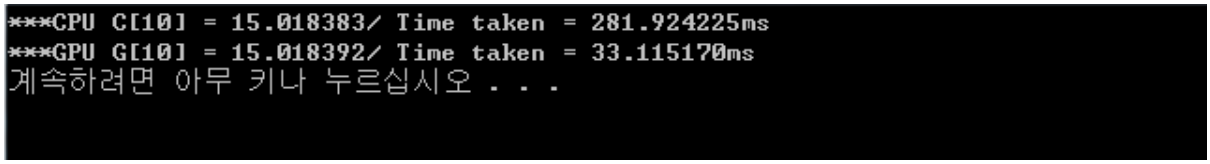
1 번 실험결과



2 번 실험결과



3 번 실험결과



4 번 실험결과

```
***CPU C[10] = 1.018927/ Time taken = 282.878143ms
***GPU G[10] = 1.018928/ Time taken = 45.138783ms
계속하려면 아무 키나 누르십시오 . . .
```

실습 1-3

1 번 실험 결과

```
n: 1048576
n = 1048576  file open ok.
***CPU C[10] = 0.774/ Time taken = 701.499512ms
***GPU C[10] = 0.774/ Time taken = 0.293216ms
계속하려면 아무 키나 누르십시오 . . .
```

2 번 실험결과

```
n: 1048576
n = 1048576  file open ok.
***CPU C[10] = 0.774/ Time taken = 688.022583ms
***GPU C[10] = 0.774/ Time taken = 0.214304ms
계속하려면 아무 키나 누르십시오 . . .
```

3 번 실험 결과

```
n: 1048576
n = 1048576  file open ok.
***CPU C[10] = 0.774/ Time taken = 708.530640ms
***GPU C[10] = 0.774/ Time taken = 0.194816ms
계속하려면 아무 키나 누르십시오 . . .
```

4 번 실험 결과

```
n: 1048576
n = 1048576  file open ok.
***CPU C[10] = 0.774/ Time taken = 703.138672ms
***GPU C[10] = 0.774/ Time taken = 0.206592ms
계속하려면 아무 키나 누르십시오 . . .
```

숙제 1

1 번 실험결과

```
n = 1048576  file open ok.
***CPU Time taken = 5.446019ms
***GPU Time taken = 2.614304ms
CPU result X0[200], fX0[200] = -1.750021, -0.000015
GPU result X0[200], fX0[200] = -1.750021, -0.000012
계속하려면 아무 키나 누르십시오 . . .
```

2 번 실험결과

```
n = 1048576  file open ok.
***CPU Time taken = 7.537066ms
***GPU Time taken = 2.602848ms
CPU result X0[150], fX0[150] = 1.358931, 0.000004
GPU result X0[150], fX0[150] = 1.358931, 0.000001
계속하려면 아무 키나 누르십시오 . . .
```

3 번 실험결과

```
n = 1048576  file open ok.
***CPU Time taken = 6.731377ms
***GPU Time taken = 2.427904ms
CPU result X0[150], fX0[150] = 1.358931, 0.000004
GPU result X0[150], fX0[150] = 1.358931, 0.000001
계속하려면 아무 키나 누르십시오 . . .
```

4 번 실험결과

```
n = 1048576  file open ok.
***CPU Time taken = 6.731377ms
***GPU Time taken = 2.427904ms
CPU result X0[150], fX0[150] = 1.358931, 0.000004
GPU result X0[150], fX0[150] = 1.358931, 0.000001
계속하려면 아무 키나 누르십시오 . . .
```

5 번 실험결과

```
n = 1048576  file open ok.
***CPU Time taken = 5.721625ms
***GPU Time taken = 2.429856ms
CPU result X0[1048575], fX0[1048575] = -0.800521, 0.000008
GPU result X0[1048575], fX0[1048575] = -0.800521, 0.000005
계속하려면 아무 키나 누르십시오 . . .
```

숙제 2

1 번 실험결과

```
*** The problem size is 67108864.
***_CPU_ Time taken for computing 67108864 Fibonacci numbers is 2911.885254ms

***_GPU_ Time taken for computing 67108864 Fibonacci numbers is 52.016930ms
*** Fibonacci number of 37 is (CPU :24157826  , GPU :24157826).
계속하려면 아무 키나 누르십시오 . . .
```

2 번 실험결과

```
*** The problem size is 67108864.
***_CPU_ Time taken for computing 67108864 Fibonacci numbers is 2918.122314ms

***_GPU_ Time taken for computing 67108864 Fibonacci numbers is 55.147839ms
*** Fibonacci number of 37 is (CPU :24157826  , GPU :24157826).
```

3 번 실험결과

```
*** The problem size is 67108864.
***_CPU_ Time taken for computing 67108864 Fibonacci numbers is 2947.813232ms

***_GPU_ Time taken for computing 67108864 Fibonacci numbers is 47.302017ms
*** Fibonacci number of 37 is (CPU :24157826  , GPU :24157826).
```

4 번 실험결과

```
*** The problem size is 67108864.
***_CPU_ Time taken for computing 67108864 Fibonacci numbers is 2901.184082ms

***_GPU_ Time taken for computing 67108864 Fibonacci numbers is 70.568733ms
*** Fibonacci number of 37 is (CPU :24157826  , GPU :24157826).
```

5 번 실험결과

```
*** The problem size is 67108864.
***_CPU_ Time taken for computing 67108864 Fibonacci numbers is 2949.041504ms

***_GPU_ Time taken for computing 67108864 Fibonacci numbers is 70.829025ms
*** Fibonacci number of 37 is (CPU :24157826  , GPU :24157826).
계속하려면 아무 키나 누르십시오 . . .
```