**고소실 11주차 보고서**

20191571 김세영

**1. 실습**

1) GPU: NVIDIA GeForce GTX 1660 SUPER

Compute Capability: 7.5

CUDA Compute Capability가 제공하는 각종 성능 및 스펙:

텍스트이(가) 표시된 사진

자동 생성된 설명

2) Compute Capability 가 7.5인 GPU의 Maximum number of threads per block은 1024이다.

데이터 사이즈를 1 << 24(16777216)으로 변경하였다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thread block size | (8,8) | (16,8) | (16,16) | (32,8) | (32,16) | (32,32) |
| CPU | 509.107086 | | | | | |
| GPU | 1.423392 | 1.419488 | 1.434752 | 1.333216 | 1.3856 | 1.513984 |

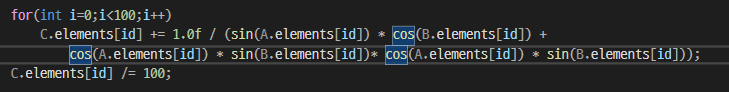
(단위:ms)

위의 표에 CPU의 수행시간을 기재해 두었으나 CPU와 GPU에서 수행하는 작업이 완전히 같지 않기 때문에 정확한 비교라고 볼 수 없다. GPU에서는 추가로 temp를 0부터 65536까지 증가시키는 작업을 반복문을 통해 수행하였다.

실험한 thread block size 중 (32,8)일 때의 수행시간이 제일 짧고, (32,32)일 때의 수행시간이 가장 길다.

(32,8)에서 수행시간이 가장 짧은 이유는 warp의 크기가 32이기 때문에 x dimension의 크기가 32이면 한 번에 계산될 수 있는 index가 모두 인접해있기 때문에 spatial locality가 높아져서 cache hit가 일어날 가능성이 높아지기 때문이다.

커널 프로그램이 수행해주는 계산을 복잡하게 하기 위해 커널 코드를 수정하였다.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thread block size | (8,8) | (16,8) | (16,16) | (32,8) | (32,16) | (32,32) |
| GPU | 107.7841 | 92.10844 | 92.30336 | 92.23587 | 90.51174 | 92.31239 |

(단위:ms)

실험한 thread block size 중 (32,16)일 때의 수행시간이 제일 짧고, (8,8)일 때의 수행시간이 가장 길다. (32,16)일 때 가장 짧은 이유는 이전 실험 결과에 설명한 것과 같고, (8,8)일 때는 spatial locality가 낮아서 hit miss가 날 가능성이 크기 때문에 수행시간이 길다.

3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 4 | 8 | 16 | 32 | 64 | 128 |
| CPU | 604.598877 | | | | | |
| GPU | 82.76992 | 74.7009 | 70.74605 | 68.38243 | 71.46906 | 72.09322 |

(단위:ms)

n(ELEMENT\_SIZE)를 1048576로 변경하여 실험을 진행하였다.

CPU에서는 동일한 행렬 계산을 반복문을 수행하면서 한 번에 한 index만 계산하지만, GPU에서는 동시에 여러 index에서의 계산이 일어나기 때문에 CPU보다 시간이 줄어든다. 실험한 thread block size 중 가장 수행시간이 짧은 것은 32, 가장 수행시간이 긴 것은 4이다. warp의 개수가 32개(의 thread)이기 때문에 32나 32의 배수일 때 수행시간이 짧다.

반대로 thread block size가 4면 warp 단위로 처리될 때 실제로 계산되는 thread는 4개 뿐이므로 처리해야하는 횟수가 늘어나기 때문에 수행시간이 늘어난다.

**2. 과제**

1) thread block을 1차원으로 구성하였고, N\_EQUATION을 1048576으로 설정하였다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
| CPU | 22.237499 | | | | | | |
| GPU | 3.943712 | 1.860256 | 1.004352 | 0.57632 | 0.359904 | 0.358752 | 0.360992 |

(단위:ms)

CPU에서는 동일한 계산을 반복문을 수행하면서 한 번에 한 index만 계산하지만, GPU에서는 동시에 여러 index에서의 계산이 일어나기 때문에 CPU보다 시간이 줄어든다. 실험한 thread block size 중 가장 수행시간이 짧은 것은 128, 가장 수행시간이 긴 것은 4이다.

이러한 실험 결과가 나온 이유는 1.실험 3)에서 설명한 것과 같다.

warp의 배수인 32,64,128,256의 실험결과가 warp보다 작은 thread block size보다 수행시간이 빠르다. thread block이 한 번에 병렬처리로 수행되는 단위가 warp이기 때문이다.

2) thread block을 1차원으로 구성하였다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
| CPU | 1079.524658 | | | | | | |
| GPU | 46.53968 | 24.21923 | 13.46301 | 7.086976 | 5.087328 | 5.02352 | 5.186496 |
| speed up | 23.19579 | 44.57303 | 80.18451 | 152.3251 | 212.1988 | 214.8941 | 208.1414 |

(단위:ms, speed up: cpu time/gpu time)

CPU에서는 동일한 계산을 반복문을 수행하면서 한 번에 한 index만 계산하지만, GPU에서는 동시에 여러 index에서의 계산이 일어나기 때문에 CPU보다 시간이 줄어든다. 실험 결과, thread block size를 warp의 배수로 한 경우에는 152.3251~214.8941배 까지 빨라지는 것을 확인할 수 있다.

실험한 thread block size 중 가장 수행시간이 짧은 것은 128, 가장 수행시간이 긴 것은 4이다.

이러한 실험 결과가 나온 이유는 1.실험 3)에서 설명한 것과 같다.

warp의 배수인 32,64,128,256의 실험결과가 warp보다 작은 thread block size보다 수행시간이 빠르다. thread block이 한 번에 병렬처리로 수행되는 단위가 warp이기 때문이다.

3)( 단위:ms)

nf=1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 8 | 16 | 32 | 64 | 128 | (32,8) | (32,16) |
| CPU | 250.661 | | | | | | |
| GPU | 22.805 | 12.411 | 6.696 | 3.56 | 2.983 | 3.013 | 3.06 |

nf=4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 8 | 16 | 32 | 64 | 128 | (32,8) | (32,16) |
| CPU | 285.035 | | | | | | |
| GPU | 21.232 | 11.84 | 5.87 | 3.255 | 2.92 | 3.114 | 3.37 |

nf=16

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 8 | 16 | 32 | 64 | 128 | (32,8) | (32,16) |
| CPU | 367.685 | | | | | | |
| GPU | 27.428 | 14.987 | 8.584 | 6.551 | 6.21 | 5.886 | 5.98 |

nf=64

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 8 | 16 | 32 | 64 | 128 | (32,8) | (32,16) |
| CPU | 738.104 | | | | | | |
| GPU | 83.751 | 47.184 | 29.063 | 24.556 | 22.655 | 20.637 | 19.685 |

nf=256

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 8 | 16 | 32 | 64 | 128 | (32,8) | (32,16) |
| CPU | 2812.597 | | | | | | |
| GPU | 271.305 | 151.94 | 108.891 | 90.294 | 92.892 | 82.322 | 74.657 |

nf=1024

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread block size | 8 | 16 | 32 | 64 | 128 | (32,8) | (32,16) |
| CPU | 9328.749 | | | | | | |
| GPU | 888.258 | 477.241 | 298.771 | 314.896 | 333.219 | 305.973 | 282.048 |

nf 값이 증가할수록 [index-nf,index+nf] 구간 길이가 늘어나서 반복문 횟수가 늘어나기 때문에 CPU와 GPU의 수행시간 모두 증가한다.

nf=1,4일 때는 thread block size가 128(1차원)일 때 가장 수행시간이 적게 걸렸고, nf=16일 때는 (32,8), nf=64,256,1024일 때는 (32,16)에서 가장 수행시간이 적게 걸렸다.

thread block size가 8일 때 가장 수행시간이 오래 걸렸다.

이러한 실험 결과가 나온 이유는 1.실험 3)에서 설명한 것과 같다.

GPU는 nf=1과 nf=4에서는 큰 수행시간 차이가 없었지만, nf=4와 nf=16에서는 최소 수행시간이 2.015753배 증가했고, nf=16과 nf=64에서는 3.344376배, nf=64, nf=256에서는 3.792583배, nf=256, nf=1024에서는 3.777918배 증가했다. 두 nf가 작을 때의 계산량의 차이보다 두 nf가 클 때의 계산량의 차이가 더 크기 때문이다.

nf가 클수록 (CPU에서의 수행시간)/(GPU에서의 수행시간)의 값이 줄어든다. nf=1일 때는 97.61473이었지만, nf=1024일 때는 33.07504이다.