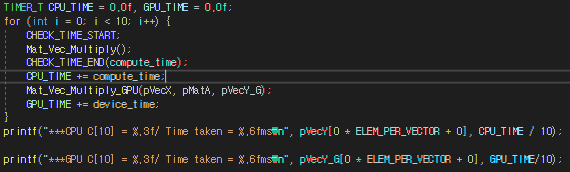
[고소실\_11주차과제]2반\_20161595\_배성현

실험조건: GTX 1660Ti, Release 모드 이용

**실습 3. (신빙성을 높이기 위하여 10번씩 측정하여 평균 이용, CUDA프로그램의 경우 커널 프로그램의 속도 측정, 데이터 이동시간 제외)**



먼저 위와 같이 for문을 통해 10번씩 실행시켜주어 평균을 내어 시간을 측정하였다.

**(i) CPU수행시간(n=2^20), (ii) GPU 수행시간(n=2^20)**

* **BLOCK SIZE=32일 때**

(약 12.92배 성능 향상)

* **BLOCK SIZE=64일 때**

(약 12.7배 성능 향상)

* **BLOCK SIZE=128일 때**

(약 12.68배 성능 향상)

* **BLOCK SIZE=256일 때**

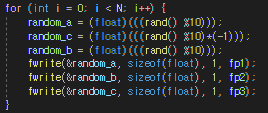
(약 12.53배 성능 향상)

* **BLOCK SIZE=512일 때**

(약 12.51배 성능 향상)

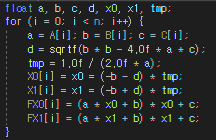
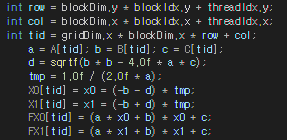
CPU수행시간과 GPU수행시간을 비교하여 보게 되면 모든 경우에 있어서 GPU코드의시간이 훨씬 적게 걸린 것을 확인할 수 있다. 이는 CPU코드의 경우 하나의 행렬의 원소를 구하는 연산을 한 개씩 순차적으로 계산을 하지만 GPU코드의 경우 이를 병렬적으로 처리하기 때문이다. 그리고 또한 GPU코드 내에서도 BLOCK SIZE에 따라서 성능이 약간씩 다른 것을 알 수 있는데 BLOCK SIZE가 32일 때 가장 시간이 적게 걸리고, 가장 성능향상이 좋게 일어난 것을 확인할 수 있다.

**숙제 1. (신빙성을 높이기 위하여 10번씩 측정하여 평균 이용, CUDA프로그램의 경우 커널 프로그램의 속도 측정, 데이터 이동시간 제외)**



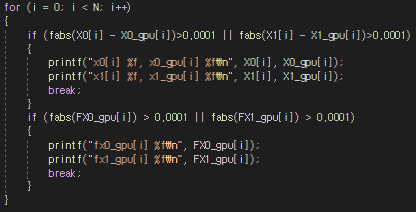
먼저 이차방정식이 두개의 실근을 무조건 가질 수 있도록 위와 같이 random하게 a, b, c를 생성하여 주었다.

1. x0<=x1이 만족되도록 저장되도록 하기 위하여 아래와 같이 CPU와 GPU모두 x0에는 의 해가 저장되도록, x1에는 의 해가 저장되도록 하고 이를 통하여 각각 fx0과 fx1을 구해줄 수 있도록 하였다.

CPU코드 GPU 커널 코드

또한 이론적으로는 FX의 값이 0이 나와야 하지만 수치 계산 시 발생하는 계산 오차가 있기 때문에 이를 고려하여 아래와 같이 FX0과 FX1의 절대값이 0.0001보다 큰 경우가 아니면 잘 맞게 나온 것으로 하였다.



1. **N\_EQUATIONS=67108864(2^26), 각각 10번씩 수행 후 GPU코드 평균 계산**

|  |  |
| --- | --- |
| **BLOCK SIZE** | **수행 시간의 평균** |
| **32** | 7.685878 |
| **64** | 7.685664 |
| **128** | **7.660608** |
| **256** | 7.663264 |
| **512** | 7.680992 |
| **1024** | 7.894831 |

위의 block size에 따른 CUDA프로그램의 결과를 보았을 때 BLOCK SIZE가 128일 때 가장 빠른 속도를 보이는 것을 확인 할 수 있다.

1. **C/C++함수(67108864일 때)와의 CUDA 커널 프로그램과의 비교**

* CPU\_TIME=315.5824156ms

C/C++함수의 수행시간과 CUDA 프로그램의 수행시간을 비교하여 보게 되면 CUDA 프로그램의 시간이 약 1/41로 적게 걸린 것을 확인할 수 있다. 이는 CPU코드의 경우 67108864개의 이차방정식들의 해와 해에 따른 FX값을 구하는 연산을 순차적으로 계산을 하지만 CUDA 프로그램의 경우 이를 병렬적으로 처리하기 때문이다.

**숙제 2. (신빙성을 높이기 위하여 10번씩 측정하여 평균 이용, CUDA프로그램의 경우 커널 프로그램의 속도 측정, 데이터 이동시간 제외)**

1. **CPU코드 속도(n=67108864일 때)의 평균**

* CPU\_TIME=1065.925659ms

1. **CUDA 프로그램 속도(n=67108864일 때)의 평균**

|  |  |  |
| --- | --- | --- |
| **BLOCK SIZE** | **수행 시간의 평균** | **성능 향상** |
| **32** | 6.616832 | 약 161배 |
| **64** | 4.521568 | 약 235배 |
| **128** | 4.550432 | 약 234배 |
| **256** | 4.687328 | 약 227배 |
| **512** | 5.507808 | 약 193배 |
| **1024** | 6.411616 | 약 166배 |

1. (i)(CPU코드에 따라 걸린 시간)와 (ii)(CUDA 프로그램)의 시간을 비교하여 보게 되면 CUDA 프로그램의 시간이 훨씬 적게 걸린 것을 확인할 수 있다. 이는 CPU코드의 경우 n=67108864개의 피보나치 수를 구하는 연산을 한 개씩 순차적으로 계산을 하지만 CUDA 프로그램의 경우 이를 병렬적으로 처리하기 때문이다. 그리고 또한 CUDA프로그램 내에서도 BLOCK SIZE에 따라서 성능이 다른 것을 알 수 있는데 BLOCK SIZE가 64일 때 가장 시간이 적게 걸린 것을 확인 할 수 있다.

**숙제 3. (신빙성을 높이기 위하여 10번씩 측정하여 평균 이용, CUDA프로그램의 경우 커널 프로그램의 속도 측정, 데이터 이동시간 제외)**

1. **N=2^24일 때**
2. CPU수행시간(n=2^24일 때)(N\_ITERATION을 10으로 해서 평균이용)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 70.271ms | 116.293ms | 125.537ms | 208.653ms | 676.456ms | 2630.351ms |

1. GPU 수행시간(n=2^24)

* block size=32(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 1.584ms | 1.518ms | 2.221ms | 6.586ms | 21.809ms | 62.404ms |

* block size=64(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 0.822ms | 0.865ms | 1.562ms | 5.187ms | 18.182ms | 68.905ms |

* block size=128(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 0.757ms | 0.788ms | 1.378ms | 4.829ms | 18.778ms | 79.325ms |

* block size=256(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 0.688ms | 0.781ms | 1.390ms | 4.482ms | 18.234ms | 68.691ms |

* block size=512(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 0.701ms | 0.801ms | 1.408ms | 4.153ms | 15.118ms | 55.550ms |

* block size=1024(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 0.802ms | 0.888ms | 1.591ms | 4.448ms | 14.859ms | 53.078ms |

1. **N=2^26일 때**
2. CPU수행시간(n=2^26일 때)(N\_ITERATION을 10으로 해서 평균이용)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 366.851ms | 510.817ms | 545.981ms | 883.053ms | 2753.456ms | 10536.152ms |

1. GPU 수행시간(n=2^26)

* block size=32(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 6.555ms | 6.426ms | 9.169ms | 27.681ms | 101.211ms | 274.575ms |

* block size=64(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 3.395ms | 3.441ms | 6.371ms | 22.316ms | 84.238ms | 280.331ms |

* block size=128(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 2.803ms | 3.139ms | 5.888ms | 20.816ms | 89.721ms | 320.202ms |

* block size=256(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 2.852ms | 3.179ms | 5.564ms | 19.292ms | 80.083ms | 287.846ms |

* block size=512(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 2.973ms | 3.308ms | 5.878ms | 18.340ms | 68.770ms | 244.912ms |

* block size=1024(10번 돌려 평균 취한 값)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nf=1 | Nf=4 | Nf=16 | Nf=64 | Nf=256 | Nf=1024 |
| 3.264ms | 3.759ms | 6.493ms | 18.649ms | 66.951ms | 230.172ms |

CPU수행 시간과 GPU 수행 시간을 비교하여 보게 되면 N값과 block size에 관계없이 모든 경우에서 GPU의 수행 시간이 훨씬 적게 걸린 것을 확인할 수 있다. 이는 CPU코드의 경우 연산을 한 개씩 순차적으로 계산을 하지만 GPU코드의 경우 연산을 병렬적으로 처리하기 때문이다. 그리고 또한 같은 조건(Block size가 같을 때)에서는 Nf가 커질 때마다 GPU 수행시간이 점점 증가하는 것을 확인할 수 있고, 또 CPU코드 역시 Nf가 커질 때마다 수행시간이 증가하는 것을 확인할 수 있다. 이는 Nf카 커질 때마다 원소를 포함하는 윈도우가 커져 더 많은 연산이 수행되기 때문이다. 그리고 GPU코드의 block size의 관점에서 보았을 때에 block size와 Nf에 따라 수행시간의 증가, 감소가 서로 다양하게 나타나는데 Nf가 큰 경우인 Nf=256, 1024에서 n=2^24, 2^26인 경우 모두 block size 128일 때 가장 수행시간이 오래 걸렸고, 이러한 block size 128일 때의 값을 극댓값으로 하는 것을 확인할 수 있다. 또한 Nf가 작은 경우인 Nf=1, 4에서 n=2^24인 경우에는 block size=256일 때 가장 수행 시간이 짧아, 이 때의 값을 극솟값으로 하는 것을 확인할 수 있고, n=2^26인경우에는 block size=128일 때 가장 수행시간이 짧아, 이 때의 값을 극솟값으로 하는 것을 알 수 있다.