Dense Matrix Multiplication

|  |  |  |  |
| --- | --- | --- | --- |
| Flag | N | Time | Speed-Up |
| -O0 | 1024 | 12986686 | N/A |
| -O3 | 1024 | 11762395 | 1.10408517993 |

|  |  |  |  |
| --- | --- | --- | --- |
| Flag | N | Time | Speed-Up |
| -O3 | 16 | 6 | N/A |
| -O3 | 32 | 30 | 5 |
| -O3 | 64 | 260 | 8.66666666667 |
| -O3 | 128 | 2212 | 8.50769230769 |
| -O3 | 256 | 19528 | 8.82820976492 |
| -O3 | 512 | 1312502 | 67.211286358 |
| -O3 | 1024 | 11762395 | 8.96181110581 |
| -O3 | 2048 | 113681595 | 9.66483399002 |
| -O3 | 4096 | 1085812491 | 9.55134814039 |

Ans: From array size N 256 to 512 and later, the time increases sharply. The reason for that is because the cache can’t hold that many blocks once it makes prediction. For example, when the cache tries to predict, there are 20 blocks of the array, waiting to be fetched. However, the cache only accepts first 10 blocks for one prediction. It costs one miss prediction to get the rest of the 10 blocks. That is why, the time increases dramatically.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Flag | N | Block Factor | Time | Speed-Up |
| -O3 | 16 | 8 | 10 | N/A |
| -O3 | 32 | 8 | 85 | 8.5 |
| -O3 | 64 | 8 | 564 | 6.63529411765 |
| -O3 | 128 | 8 | 4347 | 7.70744680851 |
| -O3 | 256 | 8 | 34610 | 7.96181274442 |
| -O3 | 512 | 8 | 299834 | 8.66321872291 |
| -03 | 1024 | 8 | 2543662 | 8.48356757406 |
| -O3 | 2048 | 8 | 20585895 | 8.09301510971 |
| -O3 | 4098 | 8 | 195196393 | 9.48204549766 |

Ans: When the size of the array is 256, using the Dense Multiplication without blocking is faster than with blocking. The reason for that is because now the cache does not need to get all blocks

And store into one lines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Flag | N | Block Factor | Time | Speed-Up |
| -O3 | 2048 | 4 | 31270240 | N/A |
| -O3 | 2048 | 16 | 18816890 | 0.60175073808 |
| -O3 | 2048 | 256 | 21032003 | 1.11771939996 |