Boyao Zhu

CMSE822 Parallel Computing

HW#2

Due Sept 27

1. **Performance Modeling**
2. The arithmetic intensity of this kernel is , where is the number of floating point operations, , and is the number of bytes transferred from/to slow memory, double-precision numbers.
3. The kernel is memory-bound if the most of its kernel time is spent in executing memory instructions. In contrast, the kernel is compute-bound if most of its operations are ALU-FPU instruction. Assume this kernel is to executed on a processor that has 30 GB/sec of memory bandwidth, that means the processor fetch 30 GB data per second. if CPU compute speed is much larger than that, then it is memory bound. Inversely, if CPU compute speed is much lower than that, then it is compute bound.
4. Peak Gflop/s = min(95 Gflop/s, 30 GB/s \* 3/8 flop/byte) = 90 Gflop/s

A close up of a map

Description automatically generated

1. **Cache optimization: Matrix Vector Multiplication**
   1. *optMultiplication* function was attached.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | N | M | B | Naïve\_time | Opt\_time | Acceleration |
| 1 | 10000 | 10000 | 100 | 0.761 | 0.699 | 1.09 |
| 2 | 10000 | 10000 | 256 | 0.762 | 0.700 | 1.09 |
| 3 | 1000 | 1000000 | 100 | 12.488 | 7.268 | 1.72 |
| 4 | 1000 | 1000000 | 128 | 12.415 | 7.435 | 1.67 |
| 5 | 10000 | 1000 | 100 | 0.077 | 0.070 | 1.10 |
| 6 | 10000 | 1000 | 512 | 0.079 | 0.083 | 0.95 |
| 7 | 20000 | 20000 | 100 | 3.037 | 2.767 | 1.10 |
| 8 | 20000 | 20000 | 500 | 3.039 | 2.978 | 1.02 |
| 9 | 20000 | 20000 | 200 | 3.040 | 2.492 | 1.22 |
| 10 | 20000 | 5000 | 100 | 0.774 | 0.697 | 1.11 |
| 11 | 5000 | 20000 | 100 | 0.767 | 0.682 | 1.13 |
| 12 | 20000 | 5000 | 500 | 0.755 | 0.745 | 1.01 |
| 13 | 5000 | 20000 | 500 | 0.760 | 0.746 | 1.02 |

As shown in the table, the speedup of the optimized implementation over the naïve one is indicated in the right column. As indicated, when the matrix is tall, which means the number of columns is much larger than that of row, the speedup get significant improved. If it is square matrix, then the speedup is not that clear. If the matrix is fat, which means the number of columns is much smaller than that of row, the speedup is unclear, depending on the block size.

* 1. The reason of speedup in optmultiplication is because we use block matrix multiplication, where cache miss is less. For example, for matrix size N\*M\*B = 10000\*10000\*100, naïve method has 212570056 count/call in L1, while optimized method has only 19844654 count/call, which means the processor takes less times to fetch data from L2, L3 or even memory.

L1 load-use latency is more like 4 cycles.

L2 load-use latency is more like 10 cycles.

* 1. The size of L1 cache is 32K.

The size of L2 cache is 256K.

While the estimated L1 cache size based on performance is 25K.

The estimated L2 cache size is about 600K.