Part 1. Multiply and Multiply-Add

1. What machine you ran this on

Rabbit

1. Show the tables and graphs
2. What patterns are you seeing in the performance curves?
3. Why do you think the patterns look this way?
4. What is the performance difference between doing a Multiply and doing a Multiply-Add?
5. What does that mean for the proper use of GPU parallel computing?

Part 2. Multiply-Reduction.

1. show the table and graph

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Array Size (Mega Numbers) | 1 | 2 | 4 | 6 | 8 | 16 | 32 | 64 |
| Performance (Mega Multi and Reduction Per Second) | 1570.919 | 2231.946 | 2741.834 | 3045.247 | 3200.265 | 4010.672 | 4435.883 | 4960.188 |

1. What pattern are you seeing in this performance curve?

The performance increases as the array size increase and the top performance is close to 5000 Mega Multiplied and Reduced Per Second.

1. Why do you think the pattern looks this way?

When the array size is less than 60 Mega Numbers, the GPU is not so busy and the overhead may cost too much time.

1. What does that mean for the proper use of GPU parallel computing?

If the data size is too small, it is not worth to do the calculations on GPU. Only when the data size is big enough, the GPU parallel computing can play its performance advantage.

1. Tell what machine you ran this on

I ran the code on rabbit.

1. Create a table with your results

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NUMS  NUMT | 1K | 10K | 100K | 32M | 64M | 128M | 256M |
| 1(NON-VEC) | 121.56 | 132.65 | 140.21 | 209.74 | 235.56 | 284.76 | 356.49 |
| 2(NON-VEC) | 188.07 | 222.35 | 267.58 | 318.49 | 372.01 | 378.24 | 542.61 |
| 4(NON-VEC) | 239.01 | 268.72 | 440.33 | 624.02 | 683.82 | 772.39 | 927.27 |
| 1 | 442.12 | 516.38 | 594.6 | 695.24 | 710.46 | 865.17 | 1062.51 |
| 2 | 457.23 | 937.9 | 1148.89 | 1159.87 | 1160.03 | 1105.07 | 1275.17 |
| 4 | 236.68 | 1023.56 | 1784.46 | 2236.66 | 2409.53 | 2577.42 | 2855.6 |

The first row shows the size of data set, from 1K to 256M. The first column indicates the number of threads (with or without vectorized), from 1 to 4.

1. Draw graphs
2. What patterns are you seeing in the performance curves?
   1. When the size of data set is very small, the speedup of SIMD is not as much as expected.
   2. The speedup is up to 4.
   3. The speedup of SIMD over non-SIMD will drop when the size of data set is more than 32M.
3. Why do you think the patterns look this way?
   1. If the data size is too small, the benefit of SIMD will be shaded by the additional work for SIMD.
   2. I guess the processor of RABBIT supports for the 4-way SIMD. So the maximum speedup of SIMD over non-SIMD is 4.
   3. The temporal coherence is violated when the size of data set is too much. The data in cache line is possibly only used once and then be replaced.
4. What does that mean for the proper use of vectorized parallel programming?

The data set size is essential for performance. If the dataset size is too small, the benefit of SIMD will be shaded by the additional work for SIMD. If the dataset size is too much, the temporal coherence is violated and the performance will decrease.