Andrew Johnson

CS475

PROJECT 6

To start, I used my own computer for this project and decided to run it through Visual Studio. I had a hard time getting it to run on my Linux machine, and then I found it was easy to start with the zipped first file you supplied.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LOCAL 32 | MULTIPLY | | Local 32 | MULTIPLYADD | |
| NUM\_ELEMENTS | GigaMults/Sec | # of Workgroups | Global Size | GigaMults/Sec | # of Workgroups |
| 1024 | 0.018 | 32 | 1024 | 0.018 | 32 |
| 2048 | 0.036 | 64 | 2048 | 0.026 | 64 |
| 4096 | 0.046 | 128 | 4096 | 0.047 | 128 |
| 8192 | 0.148 | 256 | 8192 | 0.14 | 256 |
| 16384 | 0.311 | 512 | 16384 | 0.227 | 512 |
| 32768 | 0.403 | 1024 | 32768 | 0.338 | 1024 |
| 65536 | 1.08 | 2048 | 65536 | 0.684 | 2048 |
| 131072 | 0.547 | 4096 | 131072 | 1.579 | 4096 |
| 262144 | 1.061 | 8192 | 262144 | 0.736 | 8192 |
| 524288 | 1.998 | 16384 | 524288 | 1.108 | 16384 |
| 1048576 | 2.883 | 32768 | 1048576 | 2.463 | 32768 |
| 2097152 | 6.403 | 65536 | 2097152 | 4.621 | 65536 |
| 4194304 | 5.471 | 131072 | 4194304 | 10.222 | 131072 |
| 8388608 | 25.233 | 262144 | 8388608 | 8.113 | 262144 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Global 262144 | MULTIPLY | | Global 262144 | MULTIPLYADD | |
| LOCAL\_SIZE | GigaMults/Sec | # of Workgroups | LOCAL\_SIZE | GigaMults/Sec | # of Workgroups |
| 8 | 0.782 | 32768 | 8 | 0.78 | 32768 |
| 16 | 0.825 | 16384 | 16 | 0.787 | 16384 |
| 32 | 0.872 | 8192 | 32 | 0.773 | 8192 |
| 64 | 0.891 | 4096 | 64 | 0.543 | 4096 |
| 128 | 0.771 | 2048 | 128 | 0.615 | 2048 |
| 256 | 0.742 | 1024 | 256 | 0.625 | 1024 |
| 512 | 0.778 | 512 | 512 | 0.685 | 512 |
| 1024 | 0.763 | 256 | 1024 | 0.675 | 256 |

Graphs from the given Tables

When changing the Array sizes while using OpenCL, I see with small arrays the performance isn’t very drastic. But once I got into the millions there was an exponential growth of performance. It seems the MultAdd function was a little tougher for the computer to tackle, but not by a lot. I believe this exponential growth happens because with smaller arrays the computer is mostly just taking the time to set up and the goes through the calculations lightning fast. Although, with much larger arrays the computer sets up the arrays in OpenCL and the gets lots of computations while going through the array as if it were like the smaller ones.

When fixing the array size and changing the Local\_Size, there seems to be no change in performance from using more or less local size. I think this is because the OpenCL mostly would focus on getting the array ready which the LOCAL\_SIZE doesn’t affect even if you raise it to a large amount.

It seems that just multiplying is better computing than multiplying and adding. So I would assume that the proper use for the GPU is to have multiple single math expressions and then you can keep combining them. Kind of like using the PEDMAS with mathematics.

TABLE FOR REDUCE ARRAY

|  |  |  |
| --- | --- | --- |
| LOCAL\_SIZE 32 | MULTREDUCE | |
| Global | GigaMults/Sec | WorkGroups |
| 1024 | 0.018 | 32 |
| 2048 | 0.038 | 64 |
| 4096 | 0.079 | 128 |
| 8192 | 0.168 | 256 |
| 16384 | 0.319 | 512 |
| 32768 | 0.65 | 1024 |
| 65536 | 1.277 | 2048 |
| 131072 | 0.583 | 4096 |
| 262144 | 0.934 | 8192 |
| 524288 | 2.061 | 16384 |
| 1048576 | 3.519 | 32768 |
| 2097152 | 7.565 | 65536 |
| 4194304 | 15.565 | 131072 |
| 8388608 | 26.013 | 262144 |

Graph of the MultReduce Data

The pattern in the reduce graph is the same as the Mult graph where it’s growing exponentially. Although, I do see that it’s growing faster exponentially than the Mult graph. Sadly, I’m not sure if this is how it’s supposed to look. From the slides I saw in class, I believe this needs to be a logarithmic style graph.

If I go off the data I have here I believe that proper use of the GPU should take advantage of the reduction, so it gives off more computations faster than if you wouldn’t reduce. Although, if I side with my gut feeling about a logarithmic graph, I could say the GPU should absolutely use reduction. Because even at lower array sizes you would have so much processing power, and at the plateau that happens is just the GPU’s physical working limit. So if you used reduction you can get max performance with almost anything.