**CS 475/575 -- Spring Quarter 2017**

**Project #6**

### OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

**Professor Mike Bailey**

**Xiao Tan**

I. Runtime environment

In this project, I ran my program on OSU’s server rabbit.engr.oregonstate.edu. Then I got the following results, and graph.

Part 1.

II. Results and graph

Multiply and Multiply-Add performance versus Global Work Size

Multiply

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | m8 | m128 | m256 | m512 |
| 1 | 1.032 | 1.452 | 1.417 | 1.389 |
| 2 | 1.485 | 2.536 | 2.534 | 2.529 |
| 3 | 1.715 | 3.354 | 3.34 | 3.391 |
| 4 | 2.226 | 4.147 | 4.125 | 4.166 |
| 5 | 2.074 | 4.976 | 6.436 | 6.834 |
| 6 | 2.375 | 7.729 | 7.727 | 7.454 |
| 7 | 2.529 | 8.276 | 7.713 | 8.554 |
| 8 | 2.576 | 8.41 | 9.229 | 8.42 |
| 9 | 2.621 | 9.382 | 9.817 | 9.013 |
| 10 | 2.65 | 10.132 | 10.101 | 6.852 |
| 11 | 2.677 | 10.085 | 10.53 | 9.716 |
| 12 | 2.669 | 10.937 | 10.505 | 10.63 |
| 13 | 2.731 | 11.32 | 10.425 | 8.638 |
| 14 | 2.592 | 11.47 | 11.512 | 11.447 |
| 15 | 2.771 | 12.094 | 11.893 | 11.649 |

Multiply-Add

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ma8 | ma128 | ma256 | ma512 |
| 1 | 1.002 | 0.807 | 1.499 | 1.611 |
| 2 | 1.141 | 1.22 | 2.023 | 3.373 |
| 3 | 1.216 | 3.064 | 2.43 | 3.049 |
| 4 | 1.328 | 2.921 | 3.608 | 4.545 |
| 5 | 1.751 | 4.034 | 4.45 | 2.99 |
| 6 | 1.978 | 4.19 | 5.811 | 4.178 |
| 7 | 2.308 | 6.946 | 3.067 | 4.964 |
| 8 | 1.942 | 7.534 | 7.685 | 6.139 |
| 9 | 2.216 | 6.191 | 7.175 | 5.506 |
| 10 | 2.168 | 7.038 | 5.139 | 5.156 |
| 11 | 2.57 | 8.511 | 6.699 | 7.728 |
| 12 | 2.516 | 5.886 | 8.473 | 6.2 |
| 13 | 2.884 | 9.021 | 8.71 | 6.929 |
| 14 | 2.411 | 7.867 | 8.619 | 7.69 |
| 15 | 3.032 | 9.359 | 9.511 | 9.434 |

Second part is Multiply and Multiply-Add performance versus Local Work Size.

Multiply

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | m1 | m2 | m4 | m8 | m16 | m32 |
| 8 | 0.945 | 1.496 | 2.111 | 2.507 | 2.729 | 2.851 |
| 16 | 1.238 | 1.97 | 3.565 | 3.925 | 4.903 | 5.31 |
| 32 | 1.368 | 2.279 | 4.291 | 6.5 | 7.858 | 8.944 |
| 64 | 1.417 | 2.503 | 4.33 | 6.53 | 10.739 | 12.735 |
| 128 | 1.432 | 4.82 | 5.409 | 8.071 | 11.257 | 13.785 |
| 256 | 1.422 | 4.22 | 5.128 | 8.181 | 11.258 | 13.583 |
| 512 | 1.43 | 4.027 | 4.183 | 8.401 | 11.242 | 13.358 |

Multiply-Add

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ma1 | ma2 | ma4 | ma8 | ma16 | ma32 |
| 8 | 1.289 | 1.767 | 2.098 | 2.435 | 2.637 | 2.758 |
| 16 | 1.608 | 2.519 | 3.27 | 4.132 | 4.728 | 5.077 |
| 32 | 1.888 | 3.045 | 4.289 | 5.944 | 7.154 | 8.096 |
| 64 | 1.924 | 3.362 | 4.75 | 7.167 | 9.123 | 10.762 |
| 128 | 1.995 | 3.273 | 5.065 | 7.344 | 9.68 | 11.307 |
| 256 | 1.998 | 3.296 | 5.048 | 7.353 | 9.608 | 11.265 |
| 512 | 1.968 | 3.441 | 4.943 | 7.301 | 9.521 | 11.223 |

III. Pattern analysis

With this two graphs, I can see, Multiply and Multiply-Add have same patterns, with constant local size, while global size rising, performance rising. For constant global size, with local size rising, they will meet horizons, once reach those tops, performance will be horizonal.

These two patterns are reasonable, as we know number of groups are Array size divide local size, and array size depends on NMB, which is global size. While global size rising, number of work groups will rise, there will be more groups working parallelly, so that performance will increase. For local size rising, because GPU can deal with some work items in the local memory, there will be a better use in each group, however, there will be a peak size for each group, when the size grew to that value, the performance will not increase.

The only different between these two conditions is Multiply performance will better than Multiply-Add, but not too much.

Assign a proper number of local size that will get a proper GPU performance, because GPU can deal with a constant work items, if lower than that will waste GPU performance, but higher than that will have nothing increase.

Part 2.

II. Results and graph

Because it is really hard to find a lot of appropriate values. I picked all success value to get the following graphs.

VS Global

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 5 | 6 | 7 | 8 |
| 8 | 0.775 | 1.129 | 1.312 | 1.073 | 1.571 | 1.589 | 1.62 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 4 | 5 | 6 | 7 | 8 |
| 16 | 1.328 | 1.237 | 1.96 | 2.249 | 2.374 | 2.448 | 2.467 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 4 | 6 | 7 | 8 |
| 32 | 1.161 | 2.333 | 3.16 | 3.425 | 3.44 | 4.405 |

VS Local

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 8 | 14 | 16 | 32 | 56 |
| 7 | 1.589 | 2.192 | 2.448 | 3.692 | 4.259 |

III. Pattern analysis

Both two situations, I got them increasing patterns, at first, they will increase faster, however, they will slow down.

I think the reason is the same as previous, when they won’t have conflicts, they will have a better performance while global size or local size rising.

Also, for the proper use of GPU parallel computing are also the same as previous, it is about setting appropriate size of local size.