Project #6

The OpenCL program to perform multiply and multiply and add was carried out on "rabbit machine (Intel Xeon)", which is running Cent-OS and has NVIDIA-Titan Black graphics card installed in it. The experimental results of Multiply and Multiply and Add are tabulated as shown below. (Note: Arrow horizontal in tables indicate local size and Vertical arrow pointing downwards indicate Global size)

**Multiply:**

*Table 1: Performance parameters of Multiply with respect varying Global and Local work size. All Performance Parameters in GigaMults/Sec*

| | 8 | 16 | 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|---|---|
| 1000 | 0.017 | 0.032 | 0.027 | 0.033 | 0.032 | 0.034 | 0.036 |
| 2000 | 0.035 | 0.035 | 0.054 | 0.061 | 0.056 | 0.051 | 0.036 |
| 4000 | 0.05 | 0.051 | 0.046 | 0.146 | 0.109 | 0.108 | 0.133 |
| 8000 | 0.112 | 0.099 | 0.099 | 0.109 | 0.205 | 0.243 | 0.212 |
| 16000 | 0.306 | 0.209 | 0.231 | 0.257 | 0.233 | 0.428 | 0.422 |
| 32000 | 0.418 | 0.552 | 0.466 | 0.559 | 0.436 | 0.446 | 1.013 |
| 64000 | 0.664 | 0.817 | 1.109 | 1.192 | 1.175 | 0.815 | 1.151 |
| 128000 | 0.399 | 0.446 | 0.542 | 0.433 | 0.52 | 0.541 | 0.543 |
| 256000 | 0.79 | 0.258 | 1.175 | 1.16 | 0.881 | 0.941 | 0.919 |
| 512000 | 1.194 | 1.356 | 1.801 | 1.801 | 1.695 | 1.417 | 1.778 |
| 1024000 | 1.752 | 0.932 | 2.789 | 3.037 | 3.236 | 2.654 | 3.15 |
| 2048000 | 2.157 | 3.151 | 4.2 | 6.05 | 5.164 | 5.951 | 5.12 |
| 4096000 | 2.497 | 3.895 | 5.903 | 7.732 | 7.81 | 7.689 | 7.943 |
| 8192000 | 2.759 | 3.098 | 7.441 | 9.629 | 10.482 | 10.829 | 10.518 |

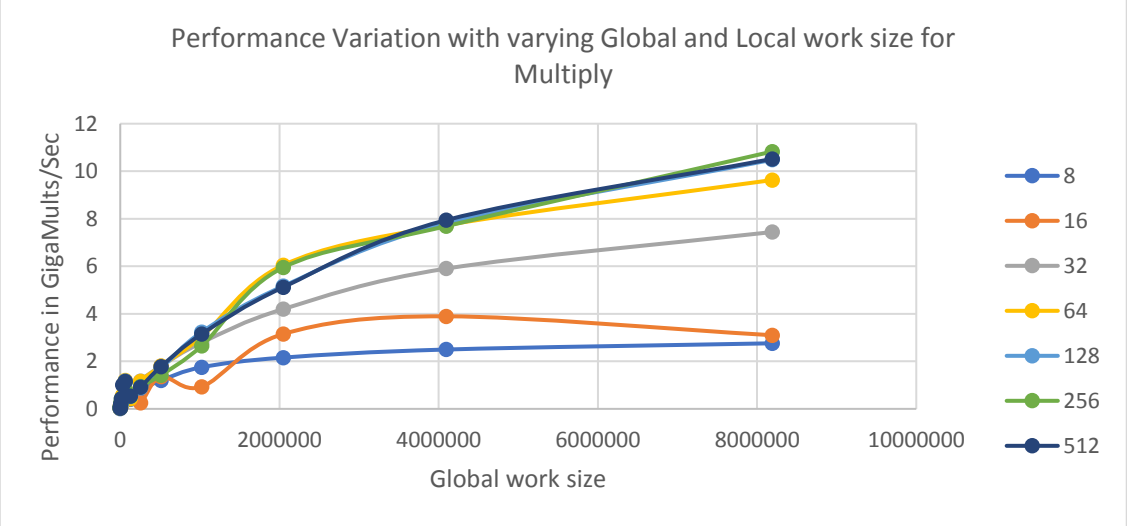



*Figure 1: Graphical Representation of Performance measured for multiply and store using OpenCL, with constant local size*

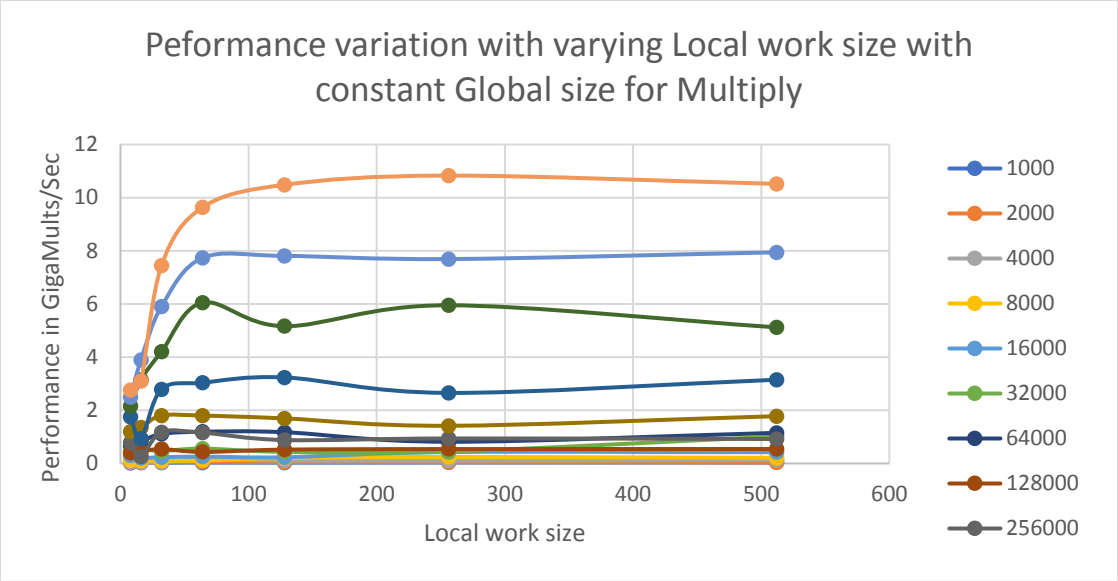


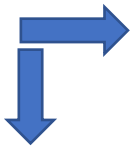

*Figure 2: Graphical Representation of Performance measured for multiply and store using OpenCL, with constant global size*

From the table 1 it is evident that the keeping global size as constant and varying the local work sizes given better performance than its predecessor, for instance for global size of 2048000 and local size of 32 the performance observed is 4.2 GigaMults/Sec, same simulation carried out with local size of 256 gives 5.951 GigaMults/Sec. This is mainly because the number of computational threads needed at GPU is very less for local work size of greater number this makes it more efficient than its predecessor local work sizes. Graphical representation of the Table 1 is shown in Figure 1.

**Multiply and Add:**

The performance measurement of multiply and add is like that of multiply only addition that is being made is d=a*b+c, unlike just a*b is prior measurement. The wall clock measurements start and stop time are still the same as that of multiply only. The performance increases with increasing global size, and much better performance is observed for varying local size with keeping constant global size.

*Table 2 : Performance parameters of Multiply and Add with respect varying Global and Local work size in GigaMults/Sec*



| | 8 | 16 | 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|---|---|
| 1000 | 0.014 | 0.032 | 0.035 | 0.036 | 0.037 | 0.035 | 0.035 |
| 2000 | 0.05 | 0.041 | 0.055 | 0.054 | 0.077 | 0.072 | 0.075 |
| 4000 | 0.072 | 0.067 | 0.095 | 0.16 | 0.159 | 0.153 | 0.158 |
| 8000 | 0.179 | 0.201 | 0.216 | 0.203 | 0.349 | 0.227 | 0.367 |
| 16000 | 0.418 | 0.472 | 0.446 | 0.314 | 0.326 | 0.537 | 0.422 |
| 32000 | 0.644 | 0.737 | 0.78 | 0.779 | 0.79 | 0.811 | 1.223 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 64000 | 0.901 | 1.241 | 1.075 | 1.541 | 1.265 | 1.045 | 1.37 |
| 128000 | 1.255 | 2.079 | 2.487 | 1.903 | 2.921 | 2.077 | 2.932 |
| 256000 | 1.844 | 3.139 | 3.43 | 3.726 | 4.662 | 4.516 | 4.824 |
| 512000 | 2.172 | 4.056 | 5.776 | 6.014 | 7.071 | 7.216 | 5.983 |
| 1024000 | 2.543 | 4.728 | 7.295 | 9.199 | 8.673 | 9.41 | 8.429 |
| 2048000 | 2.753 | 5.072 | 8.186 | 10.722 | 11.587 | 11.328 | 11.392 |
| 4096000 | 2.842 | 5.303 | 8.583 | 11.73 | 12.907 | 12.812 | 12.897 |
| 8192000 | 2.883 | 5.544 | 9.296 | 12.655 | 13.875 | 13.352 | 13.224 |

*Table 3: Performance measurement in GigaMults/Sec for Mutiply and Add*

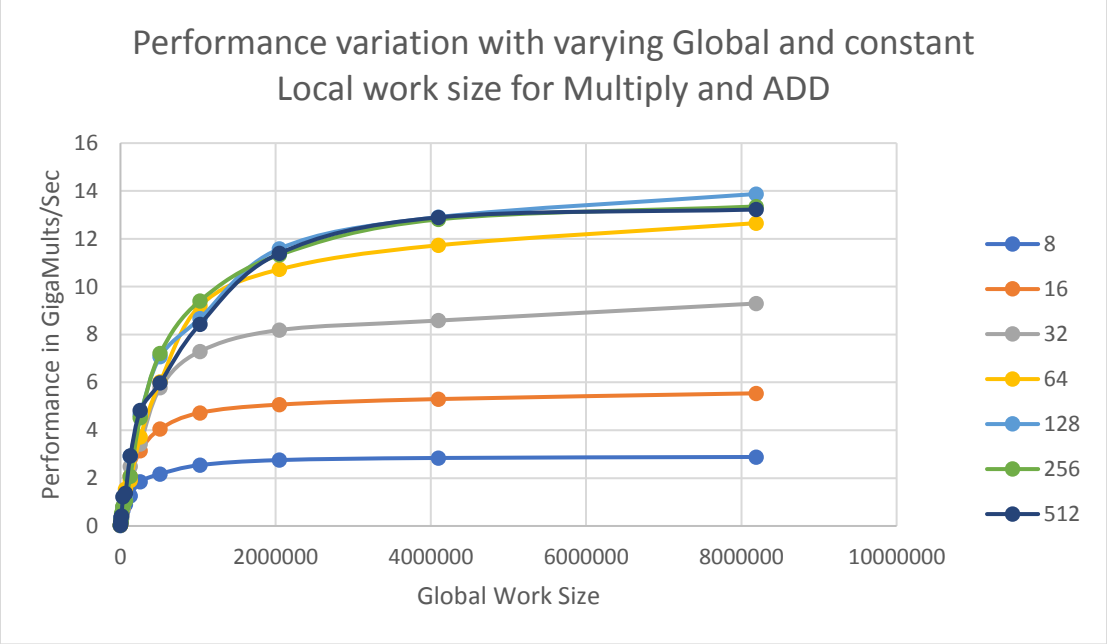



*Figure 3: Graphical Representation of Performance measured for multiply and add using OpenCL, with constant local size*

**Multiply and Reduction:**

The simulation to perform multiply and sum i.e. sum+=a*b, was run on the same machine as that of prior simulation. The strange behavior observed in this case was that, there is a saturation point for parallelizing code which much less than that of multiply and multiply and add. For a local size of 64 and global size of 64000 we hit ***Wait: clWaitForEvents failed*** which is observed immediately after enqueuing the buffer. This is observed for any combinations higher

than this. I could simulate for local work sizes of 8,16,32 for global size varying from 1K-8M and results are presented in Table 3 and graphical representation of the same in Figure 3. From the graph, it is evident that the varying global and local work sizes till saturation point can be done to get better performance out the of the system. The maximum performance observed in my simulation for multiply and reduction is 4.6 GigaMults/Sec with a global work size of 8M and local work size of 32

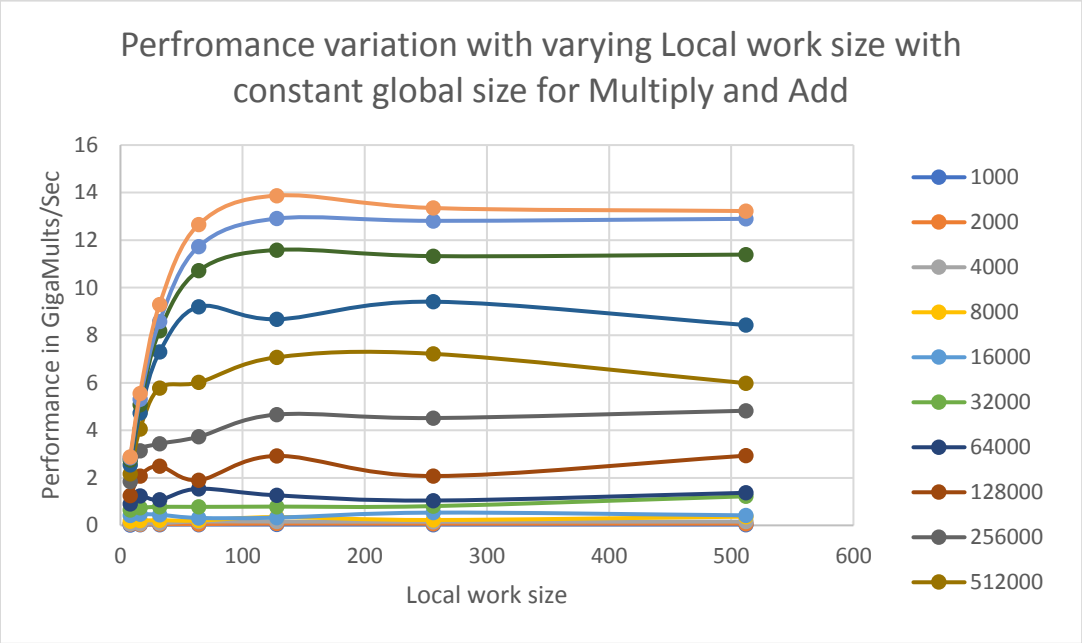



*Figure 4 : Graphical Representation of Performance measured for multiply and add using OpenCL, with constant global size*

*Table 3: Performance measurement in GigaMults/Sec for Mutiply and Add*

| | 8 | 16 | 32 |
|---|---|---|---|
| 1000 | 0.004 | 0.029 | 0.035 |
| 2000 | 0.01 | 0.01 | 0.069 |
| 4000 | 0.02 | 0.017 | 0.019 |
| 8000 | 0.036 | 0.041 | 0.039 |
| 16000 | 0.075 | 0.056 | 0.078 |
| 32000 | 0.131 | 0.119 | 0.136 |
| 64000 | 0.236 | 0.261 | 0.237 |
| 128000 | 0.42 | 0.436 | 0.52 |
| 256000 | 0.648 | 0.807 | 0.877 |
| 512000 | 1.003 | 1.213 | 1.448 |
| 1024000 | 1.365 | 1.665 | 2.467 |

| | | | |
|---|---|---|---|
| 2048000 | 1.49 | 2.243 | 3.215 |
| 4096000 | 1.606 | 2.528 | 3.826 |
| 8192000 | 1.718 | 2.712 | 4.274 |

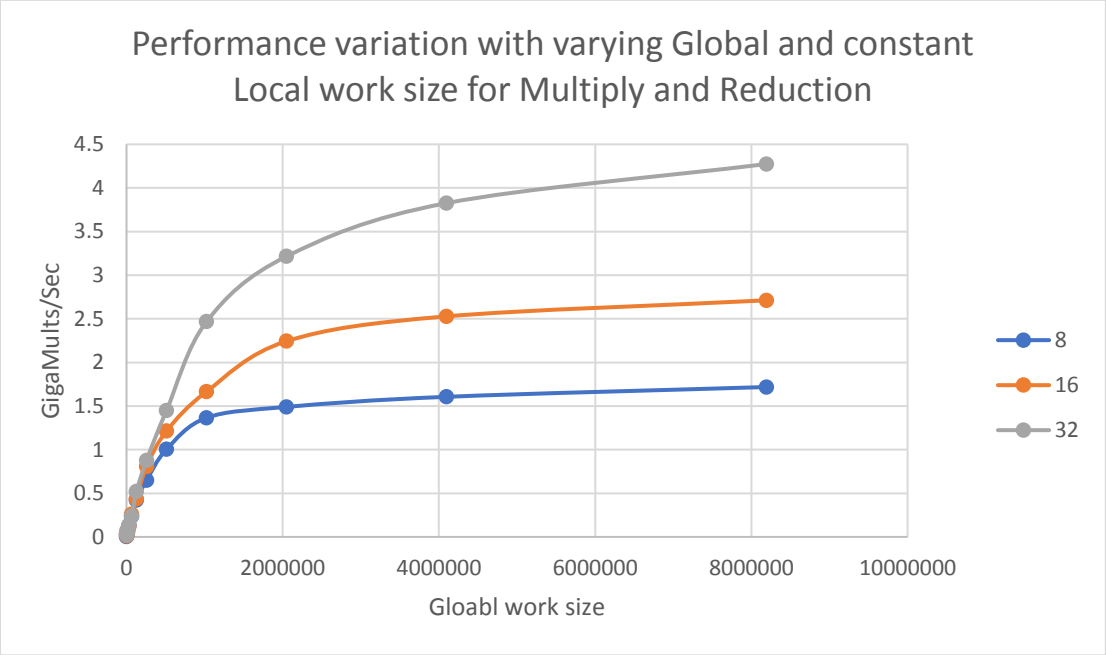



*Figure 5: Graphical Representation of Performance measured for multiply and reduction using OpenCL, with constant local size*

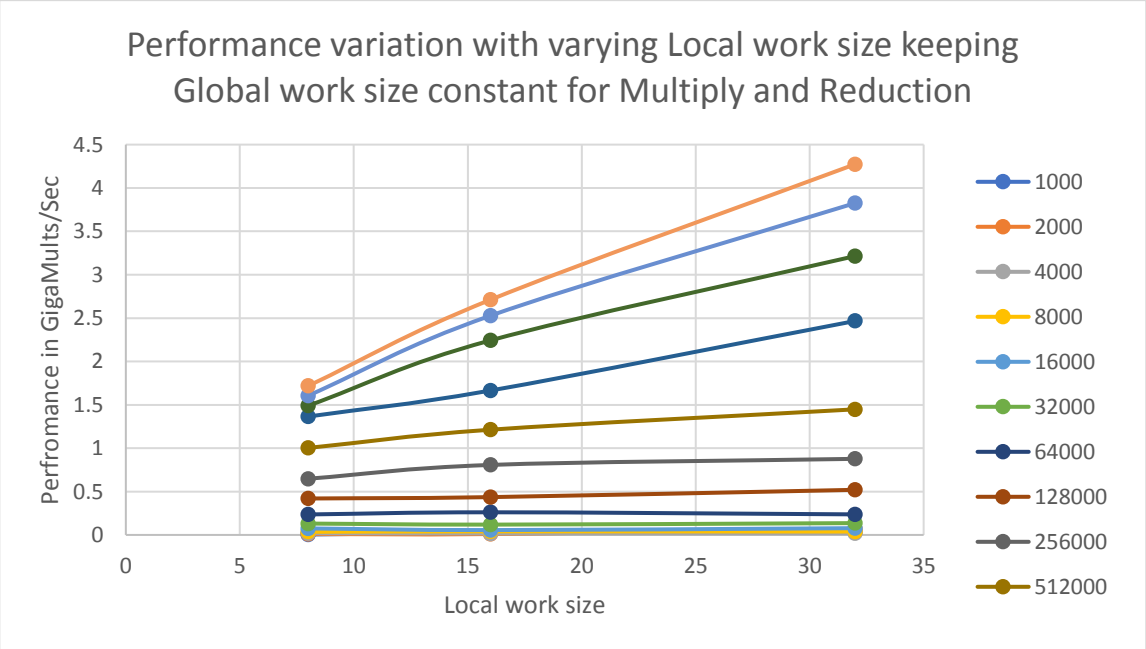



*Figure 6: Graphical Representation of Performance measured for multiply and reduction using OpenCL, with constant global size*