Nicholas Skinner

CS 475

Spring 2019

**Project 5 - OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce**

**Machine:**

I ran this on both rabbit and my local machine, but used the results from rabbit.

**Tables:**

**MULT TABLE**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **4** | **8** | **16** | **32** | **128** | **256** | **512** | **1024** |
| **1024** | **21.842** | **26.036** | **25.879** | **25.638** | **27.24** | **9.411** | **27.977** | **26.913** | **13.781** |
| **524288** | **328.903** | **963.975** | **1307.343** | **1853.882** | **1940.217** | **2276.661** | **2397.633** | **2410.363** | **2279.777** |
| **1048576** | **350.618** | **416.547** | **1664.661** | **2752.408** | **3562.636** | **3428.119** | **3422.867** | **3739.413** | **3255.614** |
| **2097152** | **285.068** | **1243.178** | **1007.259** | **3783.825** | **5291.256** | **6118.256** | **5750.171** | **5293.395** | **4076.831** |
| **4194304** | **359.633** | **1367.179** | **1290.97** | **1813.591** | **2580.673** | **2618.686** | **2643.754** | **2231.489** | **2279.828** |
| **8388608** | **366.305** | **1247.638** | **2646.751** | **4852.86** | **7789.525** | **11126.65** | **11141.98** | **11561.36** | **10192.25** |

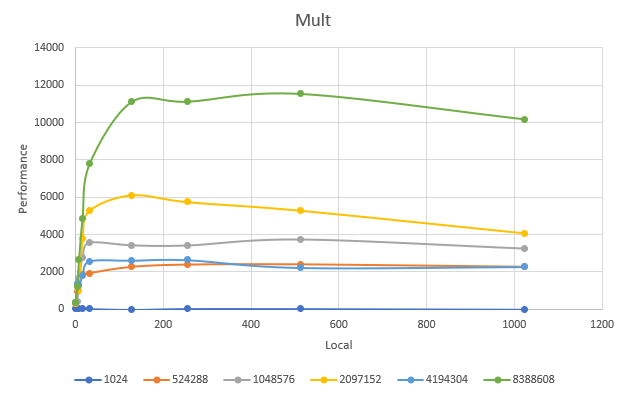
**MULT ADD TABLE**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **4** | **8** | **16** | **32** | **128** | **256** | **512** | **1024** |
| **1024** | **15.147** | **26.836** | **14.237** | **26.634** | **27.526** | **26.87** | **28.377** | **25.953** | **26.539** |
| **524288** | **301.75** | **827.836** | **1431.466** | **1919.562** | **2262.37** | **1871.472** | **2409.063** | **2384.593** | **1688.31** |
| **1048576** | **337.508** | **1140.173** | **1613.139** | **2741.55** | **3492.765** | **3986.7** | **3221.404** | **2870.363** | **3901.626** |
| **2097152** | **343.236** | **1229.464** | **2155.777** | **3075.742** | **4583.201** | **4807.614** | **5379.591** | **4428.701** | **5375.686** |
| **4194304** | **347.801** | **1288.609** | **2386.137** | **4023.428** | **5552.747** | **8095.253** | **7268.959** | **7018.385** | **6345.011** |
| **8388608** | **352.406** | **1368.113** | **2609.539** | **4631.846** | **7059.918** | **9265.198** | **9248.758** | **9694.563** | **9383.086** |

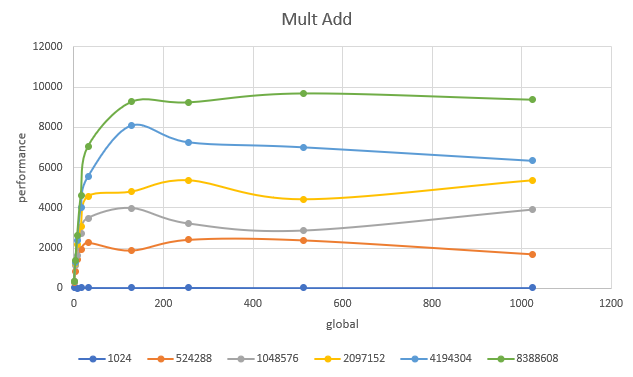
**Graphs:**

**Local vs Performance**

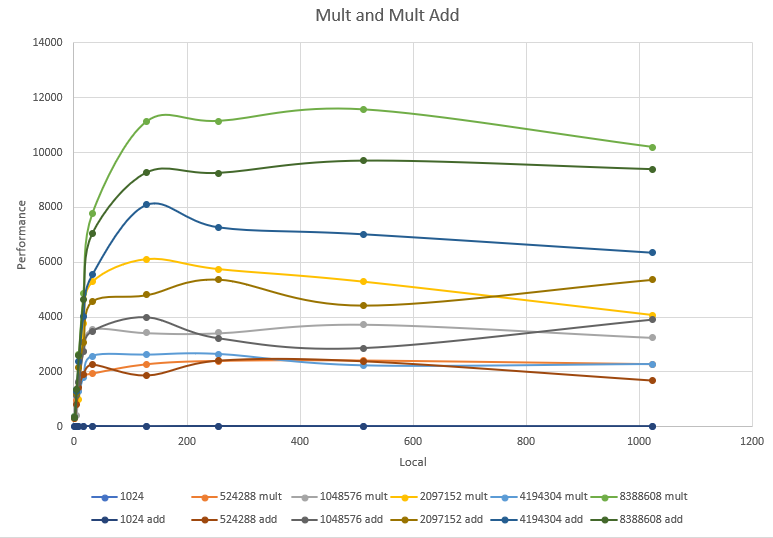
**MULT**

****

**MULT ADD**

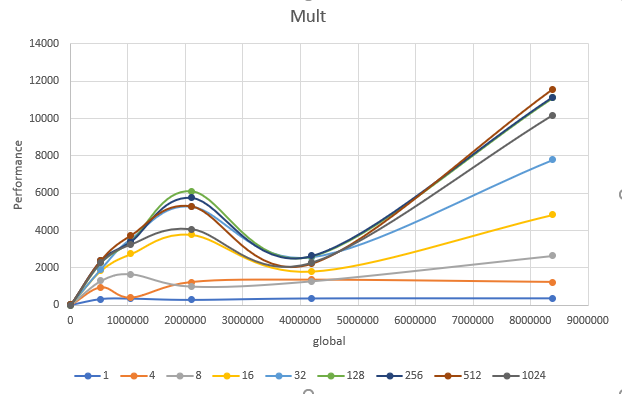
****

**BOTH**

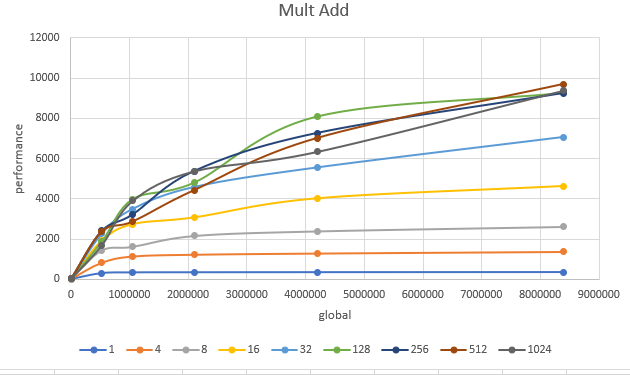
****

**Global vs Performance**

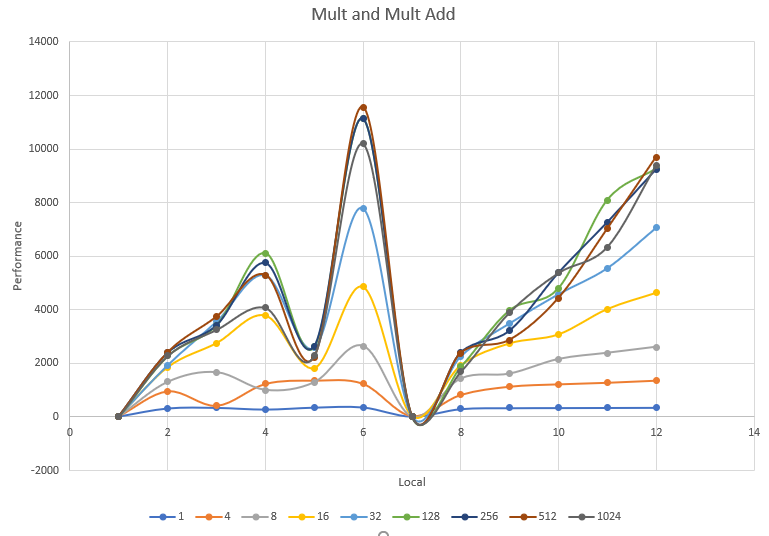
**MULT**

****

**MULT ADD**

****

**BOTH**

****

**Patterns:**

In the local work size charts, I notice that the performance will increase when the global work size increases.

I notice that performance, at least in the global work size, increases when the local work size increases. It appears from the global chart that performance above a work size of 128 is about equivalent to the performance of 128.

**Patterns Explained:**

In the local example, I believe that there is overhead when setting up to run, if the data size is too small, the system isn’t able to adequately adjust and run using the full power of the gpu.

For the global, I imagine the issue is that setting up new work groups with the associated work sizes is more overhead than benefit that you will be getting. Maybe we’re not dealing with datasets large enough to take advantage of the system?

**Performance Difference between multiply and multiply add:**

The multiply and multiply add seem to not be too different. The add might have some additional overhead when setting up the addition that knocks off a few points of performance, but otherwise it didn’t seem to be to affected. I would have imagined that it would have been an additional cycle, thus performance would be reduced by half. It must be parallelizing the addition somewhere.

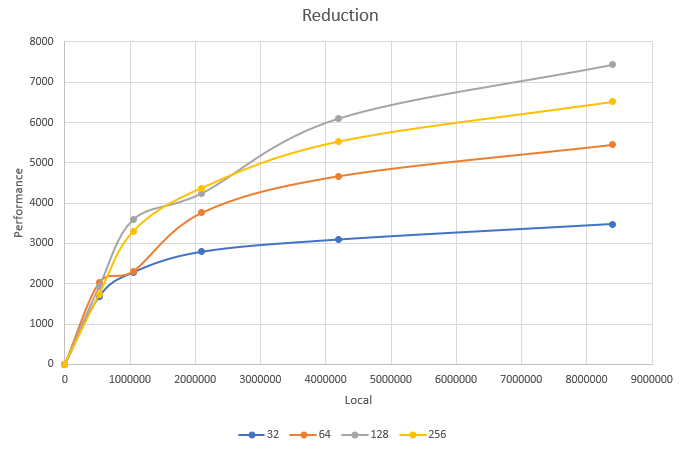
**What does this mean for proper use of gpu parallel computing?**

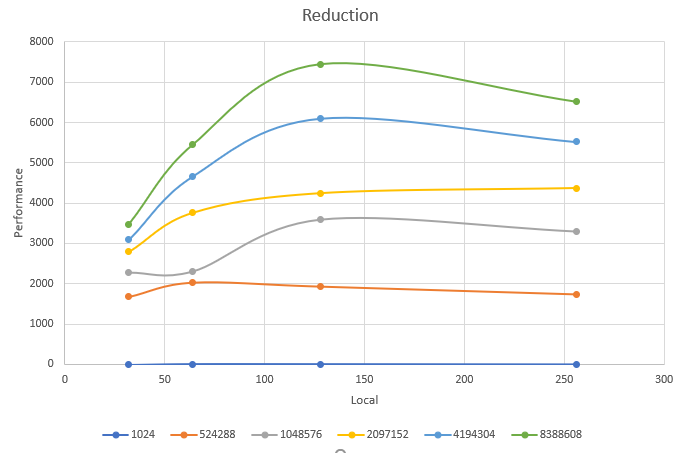
There seems to be a performance peak at a work size of ~128, and you should only be using the gpu if you have massive amounts of work to do, as overhead for setup is a big tax on the overall performance.

**Reduction Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1024** | **32** | **32** | **3.166** | **MegaMultiply-ReductionsPerSecond** |
| **1024** | **64** | **16** | **5.343** | **MegaMultiply-ReductionsPerSecond** |
| **1024** | **128** | **8** | **5.294** | **MegaMultiply-ReductionsPerSecond** |
| **1024** | **256** | **4** | **4.668** | **MegaMultiply-ReductionsPerSecond** |
| **524288** | **32** | **16384** | **1678.283** | **MegaMultiply-ReductionsPerSecond** |
| **524288** | **64** | **8192** | **2021.35** | **MegaMultiply-ReductionsPerSecond** |
| **524288** | **128** | **4096** | **1926.947** | **MegaMultiply-ReductionsPerSecond** |
| **524288** | **256** | **2048** | **1740.756** | **MegaMultiply-ReductionsPerSecond** |
| **1048576** | **32** | **32768** | **2281.32** | **MegaMultiply-ReductionsPerSecond** |
| **1048576** | **64** | **16384** | **2310.505** | **MegaMultiply-ReductionsPerSecond** |
| **1048576** | **128** | **8192** | **3596.871** | **MegaMultiply-ReductionsPerSecond** |
| **1048576** | **256** | **4096** | **3302.205** | **MegaMultiply-ReductionsPerSecond** |
| **2097152** | **32** | **65536** | **2791.434** | **MegaMultiply-ReductionsPerSecond** |
| **2097152** | **64** | **32768** | **3760.244** | **MegaMultiply-ReductionsPerSecond** |
| **2097152** | **128** | **16384** | **4246.179** | **MegaMultiply-ReductionsPerSecond** |
| **2097152** | **256** | **8192** | **4373.582** | **MegaMultiply-ReductionsPerSecond** |
| **4194304** | **32** | **131072** | **3095.653** | **MegaMultiply-ReductionsPerSecond** |
| **4194304** | **64** | **65536** | **4669.255** | **MegaMultiply-ReductionsPerSecond** |
| **4194304** | **128** | **32768** | **6105.913** | **MegaMultiply-ReductionsPerSecond** |
| **4194304** | **256** | **16384** | **5525.577** | **MegaMultiply-ReductionsPerSecond** |
| **8388608** | **32** | **262144** | **3481.641** | **MegaMultiply-ReductionsPerSecond** |
| **8388608** | **64** | **131072** | **5448.237** | **MegaMultiply-ReductionsPerSecond** |
| **8388608** | **128** | **65536** | **7440.043** | **MegaMultiply-ReductionsPerSecond** |
| **8388608** | **256** | **32768** | **6509.908** | **MegaMultiply-ReductionsPerSecond** |

**Reduction Graph:**

****

****

**Patterns:**

In the above example, the performance for the local chart seems to peak at 128, and dip back down after 128, as exampled by a work group of 256 having less performance. This seemed consistent with every time I ran the program, but there may be some variance due to other users on the machine.

**Patterns Explained:**

I imagine Idle work units are using a large amount of compute time in the smaller examples, leading to dropping performance. This is likely due to the GPU not getting enough data, and the overhead for setup may take up most of the time

For the larger groups like the 256 work group, I imagine that having a larger work group takes away from any performance increases that could occur.

**What does this mean for the proper use of parallel computing:**

You should make sure that you have a large enough data size to take full advantage of the resources available to you, so that the tax from overhead setup is negligible.

You should also be attentive of what the best local work sizes are for your program, as from the reduction example, you can more than double your performance when you select the righ one, but you can lose performance if you overshoot it.