Xiaoru Chen

chenxia2@oregonstate.edu

Project #6

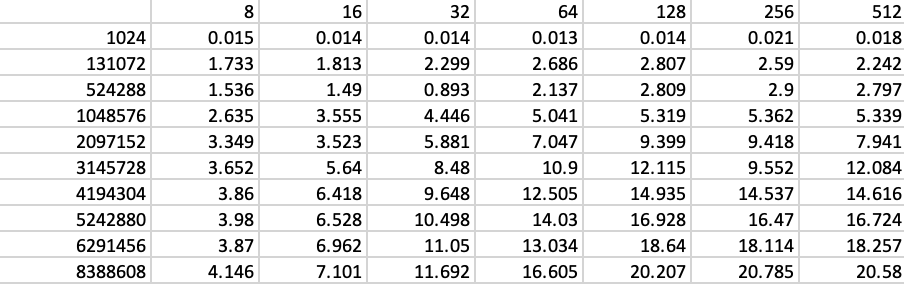
CS 475/575

2021-05-30

**Machine I ran this on:**

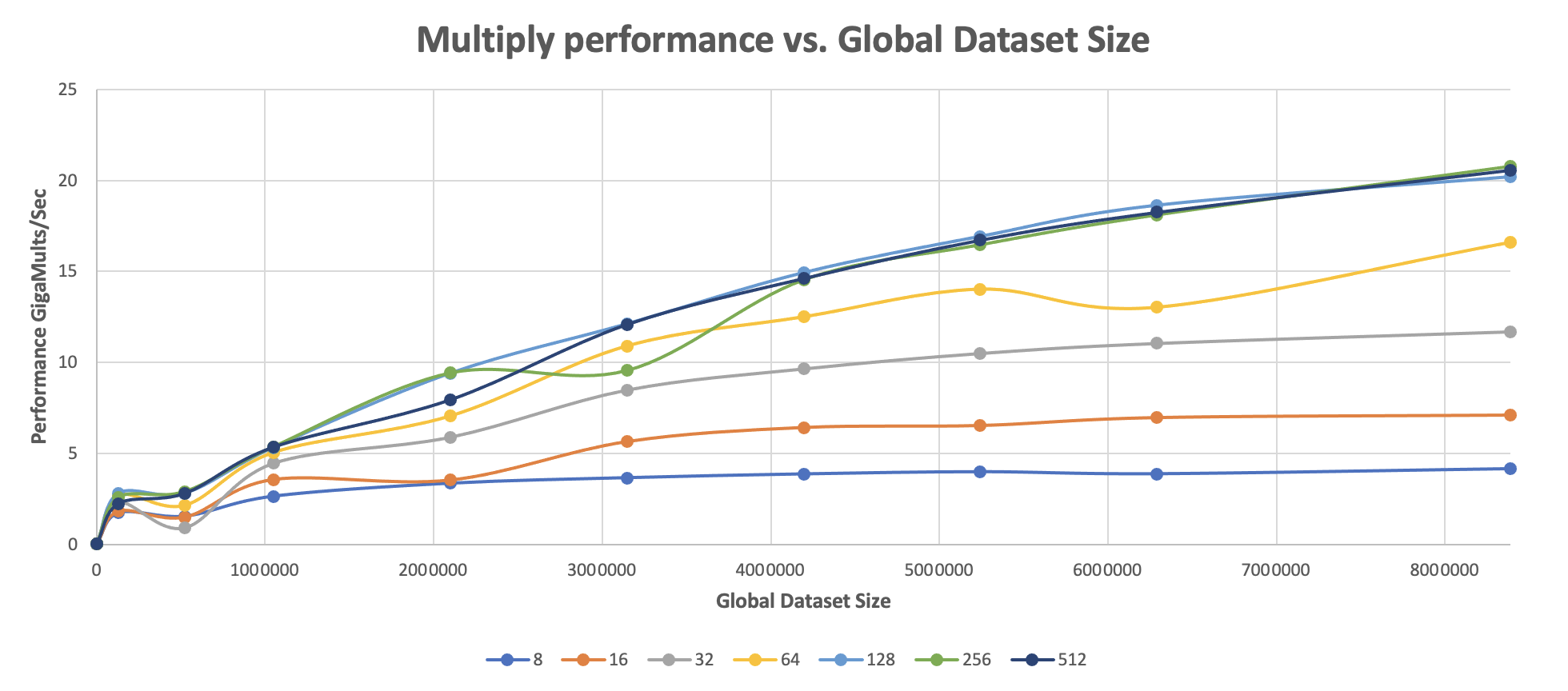
I used my MacBook pro to ssh to OSU ENGR server (DGX).

**Table for Multiply Performance:**

****

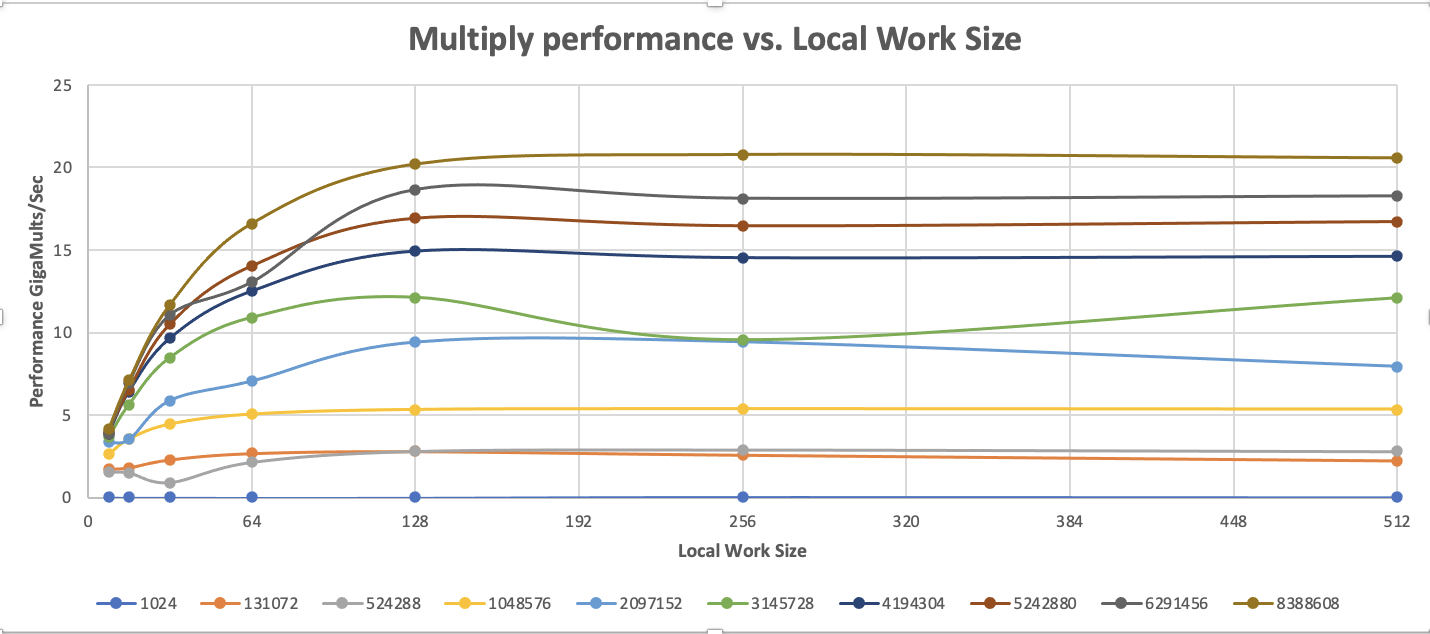
**Table.1**

**Graph for Multiply Performance vs Global Dataset Size:**

****

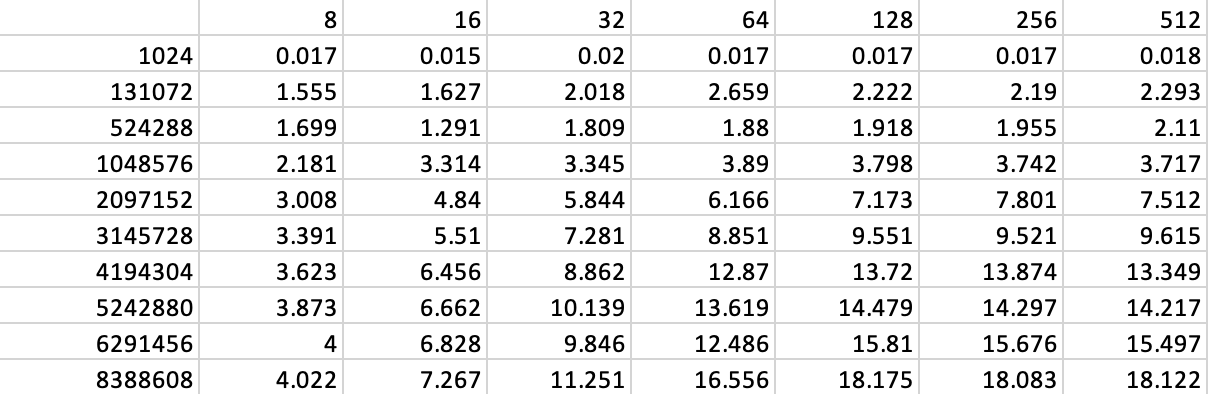
**Graph.1**

**Graph for Multiply Performance vs Local Work Size:**

****

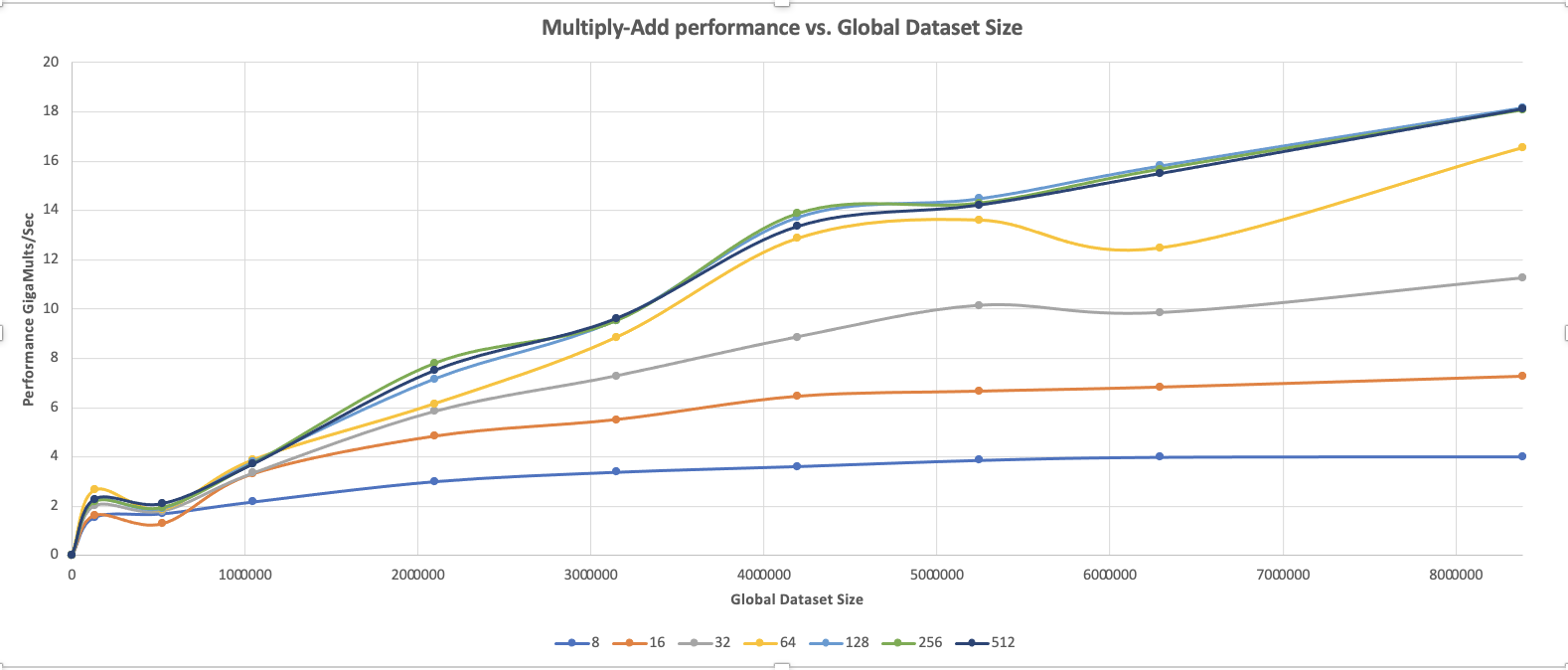
**Graph.2**

**Table for Multiply-Add Performance:**

****

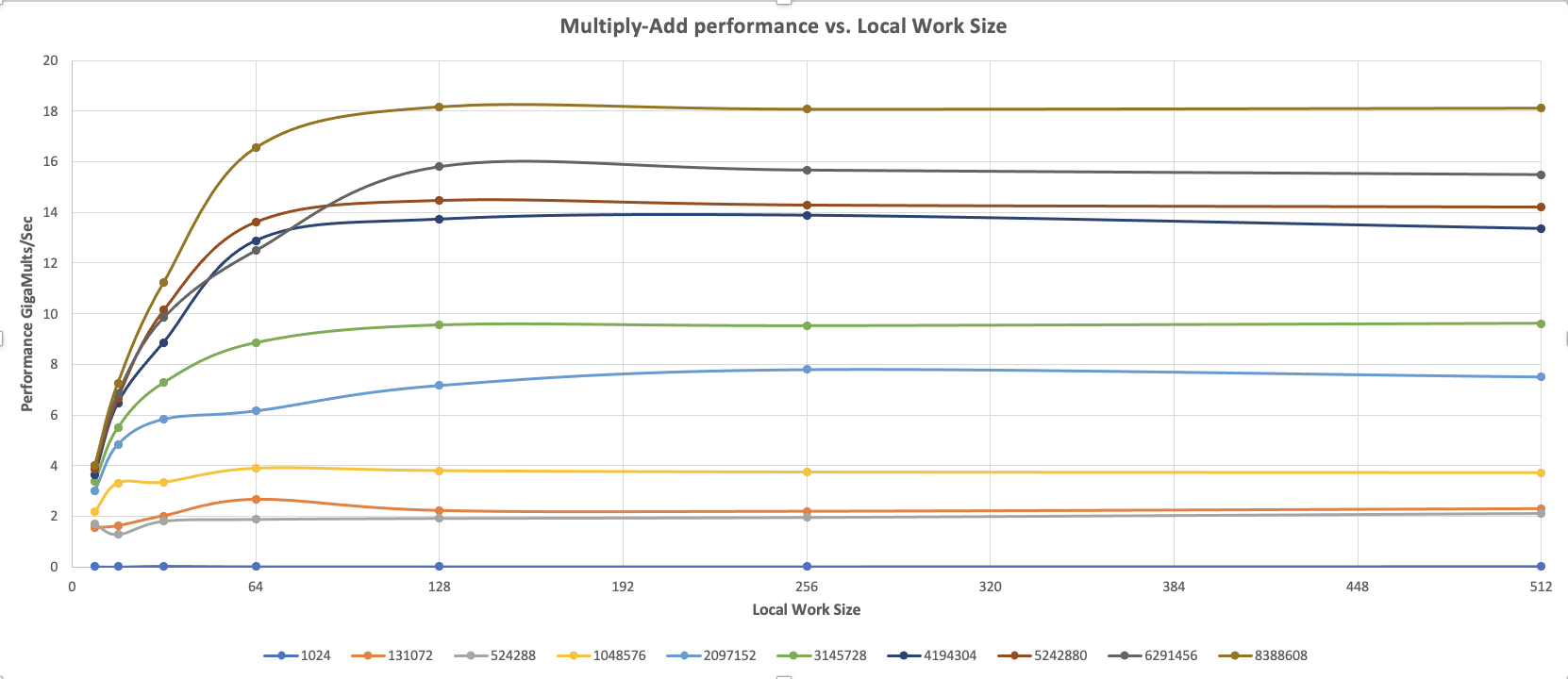
**Table.2**

**Graph for Multiply-Add Performance vs Global Dataset Size:**

****

**Graph.3**

**Graph for Multiply-Add Performance vs Local Work Size:**

****

**Graph.4**

**What patterns am I seeing in the performance curves?**

In both Multiply and Multiply-Add performance curves, when the Global Data Size increases, performance increases accordingly.

Before the local work size is less than 128, as the local work size increases, performance increases accordingly. When the local work size is greater than 128, the performance is almost flat and remains unchanged.

**Why do I think the patterns look this way?**

Because the GPU’s workload is divided into a Grid of Work-Groups, and each Work-Group runs on a Compute Unit and is organized as a grid of Work-items. Each Compute Unit is organized as a grid of Processing Elements, and Work-item runs on a Processing Element. One thread is assigned to each Work-item. The more Work-Groups we have the more threads we have, and the higher performance it performs.

When the local work size reaches 128, there are no more compute units, so the performance stays flat.

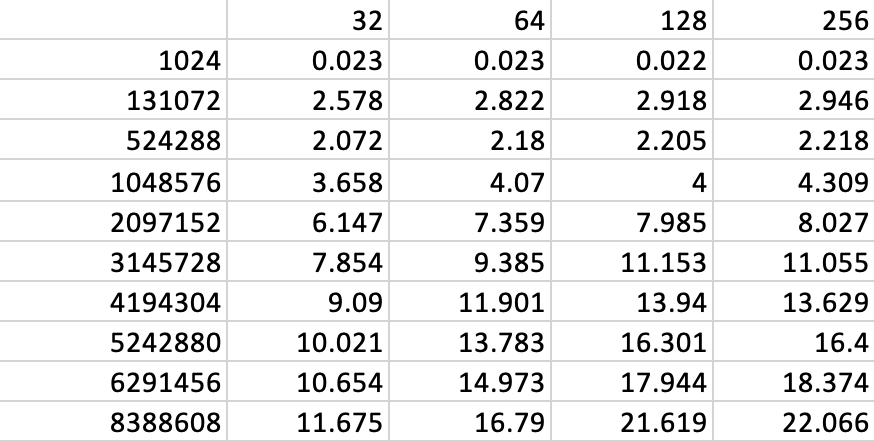
**What is the performance difference between doing a Multiply and doing a Multiply-Add?**

The performance of Multiply added is slightly lower than that of Multiply performance, since it performs an extra add operation. When the length/size of the array is getting very large, the speed gap is becoming slightly more noticeable, but it’s not significant.

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

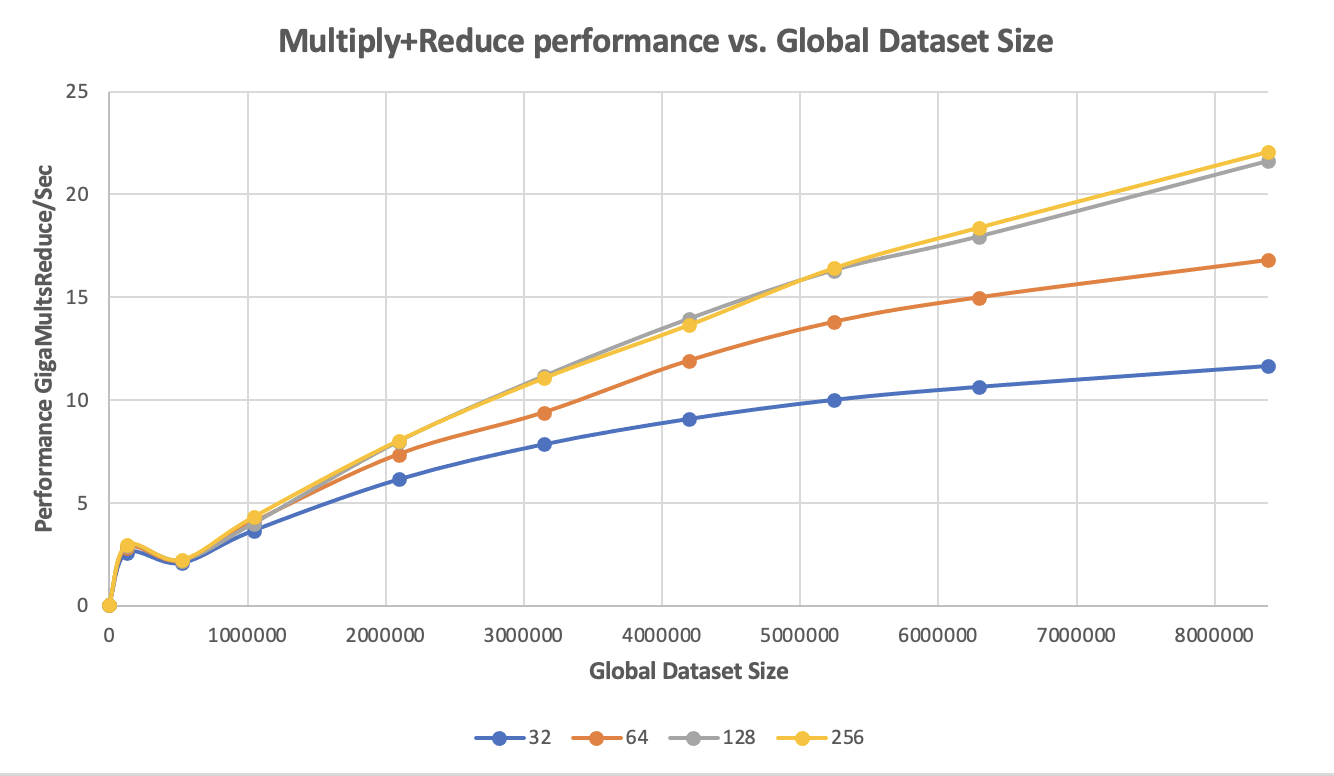
The performance difference is not significant, because of GPU’s massive parallel processing power. So the proper use of GPU parallel computing can give us a great performance boost, as it has a large number of cores.

**Table for Multiply+Reduce performance:**

****

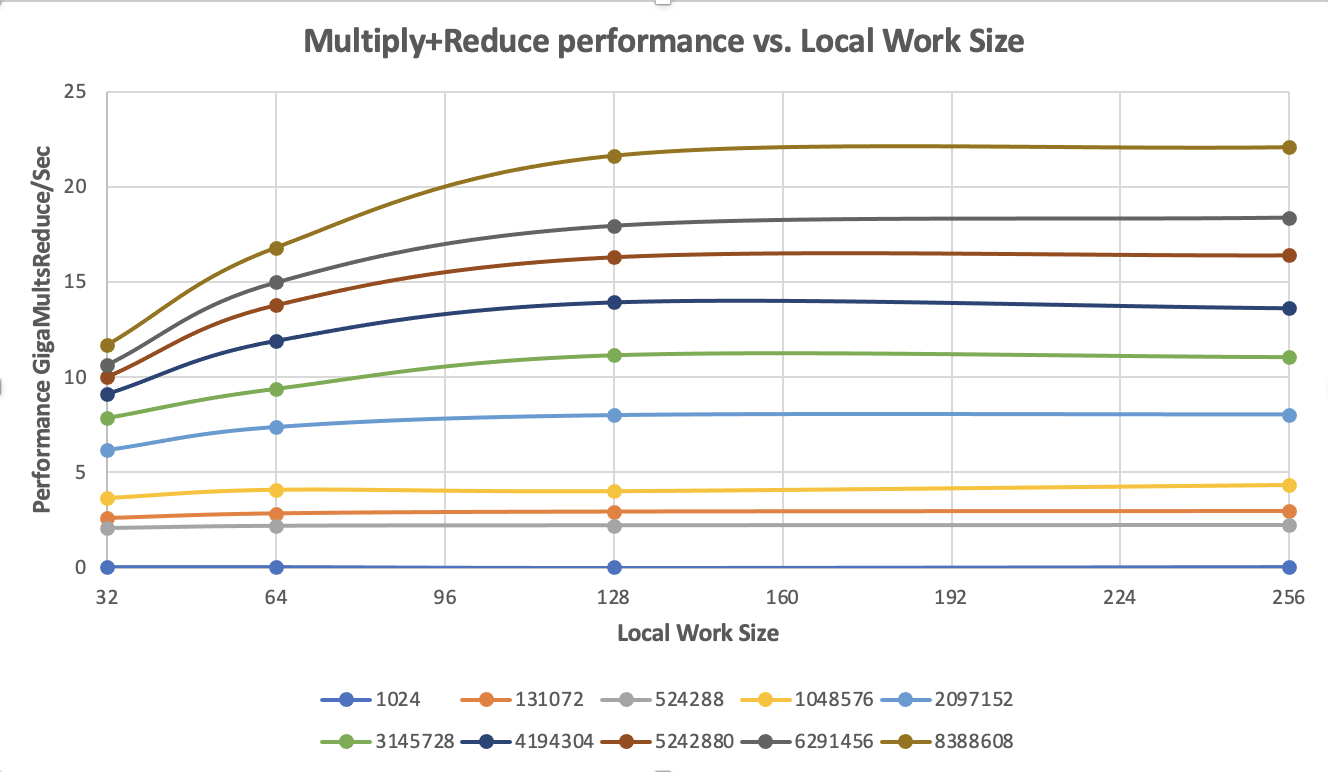
**Table.3**

**Graph for Multiply+Reduce performance vs. Local Work Size:**



**Graph.5**

**Graph for Multiply+Reduce performance vs. Local Work Size:**



**Graph.6**

**What pattern am I seeing in this performance curve?**

In the Multiply Reduce curves, when the Global Data Size increases, performance increases accordingly.

Before the local work size is less than 128, as the local work size increases, performance increases accordingly. When the local work size is greater than 128, the performance is almost flat and remains unchanged.

**Why do I think the pattern looks this way?**

Because the GPU’s workload is divided into a Grid of Work-Groups, and each Work-Group runs on a Compute Unit and is organized as a grid of Work-items. Each Compute Unit is organized as a grid of Processing Elements, and Work-item runs on a Processing Element. One thread is assigned to each Work-item. The more Work-Groups we have the more threads we have, and the higher performance it performs.

When the local work size reaches 128, there are no more compute units, so the performance stays flat.

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

The reason why Multiply reduction is faster than Multiply and Multiply-Add is that when the array multiplications done, we do not put the products into a large global device array, but into a prods[] array that is shared within its work-group.

Therefore, when using GPU parallel computing, taking this into account can improve the performance of computing.