Hsiang Lo

**Project #5**

**Goal** - Learn about Array Multiply, Multiply-Add and Multiply-Reduce using OpenCL

**Procedures**

1. Multiply two arrays together using OpenCL (benchmark against both input array size and the local work size).
2. Multiply two arrays together (benchmark against both input array size and the local work size).
3. Make two graphs – 1) Multiply and Multiply-Add performance versus Global Work Size with a series of colored Constant-Local-Work-Size-curves. 2) Multiply and Multiply-Add performance versus Local Work Size, with a series of colored Cosntant-Global-Work-Size curves
4. Commentary PDF questions 1- 6
5. **Then write another version of the code that turns it into a Multiply+Reduce application.** (Using same code as 1.) (Note: this will compute a single floating point)
6. Plot another graph showing Multiply-reduction performance versus Input Array
7. Add to the PDF the additional 1-4 quesitons on step 14.

**Notes**

* It looks like I won’t be able to use FLIP server for this assignment, ill be using Rabbit Machine instead.
* Array Multiply and the Array Multiply-Add can really be the same program. Write one program that creates the 4 arrays. Pass A,B and C into OpenCL and return D. Tehn all you have to do between the Multiply and the Multiply-Add tests is change one line in the .cl file.
* Global work sizes – 1K to 8M
* Local work sizes – 8 to 512
* Performance unit: MegaMultiplies Per Second and MegaMultiply-Adds Per Second

**Questions**

1. What machine you ran this on

I ran this assignment/homework using my MacBook Pro 2018 as I did with my previous assignments. Due to the fact that Flip is not capable of running this assignment, and the fact that I did not want to install any additional softwares, I ran this assignment on Rabbit (rabbit.engr.oregonstate). As requested, I scheduled a session and ran during my session.\

1. Show the tables and graphsMake two graphs:

* Multiply and Multiply-Add performance versus Global Work Size, with a series of colored Constant-Local-Work-Size curves
* Multiply and Multiply-Add performance versus Local Work Size, with a series of colored Constant-Global-Work-Size curve

Data for both Multiply and Multiply + Sum

ç

ç

1. What patterns are you seeing in the performance curves?

In the case of both Array Multiply and Array Multiply Add, the performance seems to increase as local size increases but only for a short while. Once it hits the a certain point, the performance becomes more steady and the slope in terms of increases in performance slows down.

On the other hand, as the size of the global size increases, it can be observed that the performance increases as well. However it can be noticed that reaching around some size, the performance again slows down just like the local sizes.

1. Why do you think the patterns look this way?

It would appear that when the size input is too small, we are not fully utilizing the resource available to us, in this case, the processing elements. As the processing elements reach a desirable size, which can be observed to be around 128, we are fully utilizing the resource, and therefore the performance reached its height.

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

Performance difference between doing a Multiply and doing a Multiply-Add can be observed in that the performance for Multiply-Add is lower than just Multiply. This can be categorized as a result for doing more work for the processors, which led to lower performance. Furthermore, the performance across Multiply-Add can be seen to be a step or two below Multiply at each variable sizes.

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

In order to fully utilize the benefit of GPU parallel computing, the more data parallelism the better. In the case of Multiply vs Multiply + Add, it can be argued that Multiply is a more straight forward than Multiply + Add in that there is less commands and more data processing. Regardless, processing an adequate amount of data sizes is ideal In using GPU parallel computing.

1. Show this table and graph - Multiply-reduction performance versus Input Array Size.

Data for Multiply + Reduction

ç

ç

1. What pattern are you seeing in this performance curve?­­­

In comparison to Multiply and Multiply + Add, Multiply + Reduction shows a graph of constant increases in performance when both the global and local sizes reaches an adequate level. This is truer for global sizes. For local sizes, signs of a slowdown for even large value of local size can be observed

1. Why do you think the pattern looks this way?

The most crucial difference between Multiply + Reduction and Multiply (And Multiply + Add) is that in the graph, the slope of the performance with a larger global size remains more relatively constant. Furthermore when the sample size is too small, GPU parallel programming isn’t the ideal tool for the job. But with a large dataset, it is a good tool.

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

GPU parallel programming truly is a data-focused method. With an adequate amount of data, GPU parallel computing is evident of a good tool.