Hardware and Software Application Testing

Topic and Interest:

- We chose this topic because after reading about Blockchain mining and other proof of work algorithms we realized that aside from it being “Common Knowledge” a look at what type of aphorisms run best on different software/hardware configurations is rather undocumented and untested. This led us to the idea of testing basic well known algorithms and their implementations on combinations of them. While we will be starting with basic algorithms such as prime factorization, pathfinding and Conway's Game of life we will be moving to more advanced topics once we are able to get up pipelines and estash an idea of what area of research is best suited for our question.

Project Pros and Cons:

- Pros:
    - Finding what each pair is best suited for can be a huge stride in future project development
    - Establishing basic knowledge of each technique and documenting it will assist in understanding the different applications of them
    - This will allow us to learn multiple advanced topics that we have not yet had exposure to in college
- Cons:
    - A lot of this is often taken for granted, we know GPUs are better at math so many people may assure our project is proving the obvious
    - Lots of set up! We have to create pipelines for each pair and setting those up, debugging, and getting them working could be quite time consuming

Main Point of Innovation:

- The value in this project is finding where it is that the hardware and software techniques really make the most difference. There are a lot of uses for all of these methods however we do not yet know when it actually makes a real difference and when implementing and releasing these features may be actually working against us. Finding where each method really shines and how to best use them will provide a good foundation for choosing the proper technology and methods for project development in games and commercial software.

Research Value:

- We want to be able to provide a place where engineers can go to identify best software hardware combinations to accomplish their goals. Efficiency has always been a headliner in the software world and a concrete source of data and numbers for understanding what approach you should take for your current project is as of now unavailable. If time permits we would also like to try the same method in multiple languages.

Tangible Goals:

- Documented results from running each hardware/software algorithm pair. We want to provide the start of a database that shows not only what if faster but also shows exactly how much on our hardware. As well as ranking and notes about ease of setup, learning curves and anything else that may be relevant when selecting a pair for your task.

Hardware and Software Needs:

- Dedicated, CUDA-Capable GPU
- Windows 10
- Visual Studio 2019

Steps we foresee:

- Basic project setup
    - Getting pipelines for each pair and making sure they run
    - Having a method of counting ms in runtime
- Creating a objective way to compare implementations by easy, startup costs and learning curve
- Testing our principle 4 algorithms
- Documenting time with each pair (on the same hardware) and propagating errors
- Expanding on our principle 4 once we are able to see where the largest differences appear

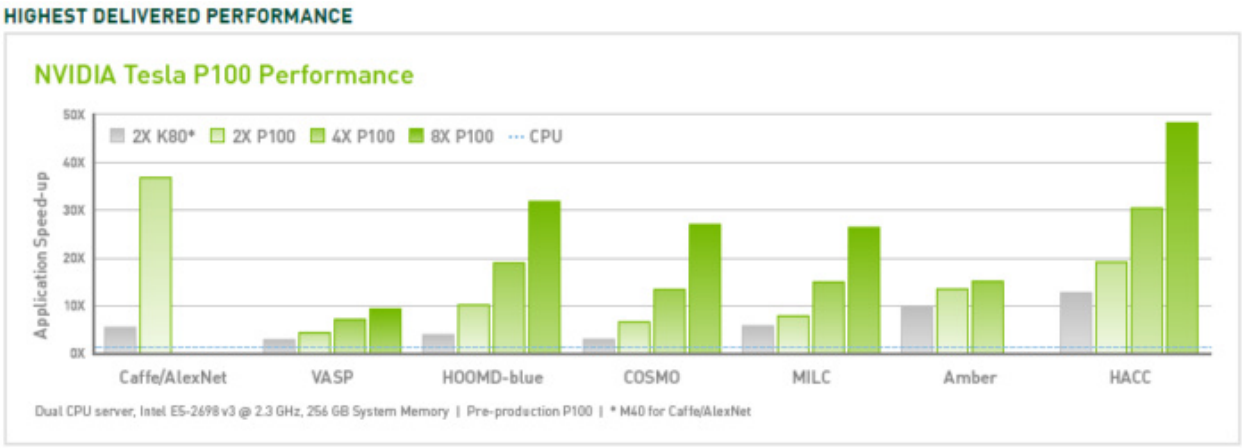
Relevancy:

- Every pillar of science has notable objective metrics, we aim to start defining where and how each hardware/software pair compare in different categories of real time development and algorithms. As modern computational power becomes more accessible these discrepancies get larger and larger. Knowing how to choose what you need for your project is becoming increasingly relevant.

Career Advancement:

- This is a good opportunity to show that we as programmers are able to not only work on research based projects but also that we are able to interweave a variety of advanced techniques. This opportunity is also good for us to learn new advanced skills that we have not yet had a change to learn in our college curriculum.

Annotated References (5 min):

- [What is CUDA? Parallel programming for GPUs]
    - Compute Unified Device Architecture
    - Works well for general computing and is similar to compute shaders however they can run in a thread like structure in parallel
    - Performs better than OpenGl however is less generic and many deep learning frameworks do not support it yet
    - CUDA Application domains in order of use: Bioinformatics, computational chem, finance, fluid dynamics. As you can see it is used for raw processing power in the pillar sciences.
    - 

- [OpenGL Wiki: Compute Shader]
    - Operated differently from other shaders, they have well defined input and output.
    - Can do tasks not specifically related to rendering, however they work best when their task is 3 dimensional as that is where their edge is.
    - While not part of the render pipeline you can inset them at the start or create their own pipeline all together

    - You can invoke a computer shader on command not just when it runs in the pipeline.
    - They share memory and can have a similar effect to threading while all working on the same pool of information.
- How to Take Advantage of Multithreaded Programming and Parallel Programming in C/C++.
    - The point of parallel programming is to increase performance, it's a broad concept and can have issues with any program that needs to run in sync. Threads all compete to do the same goal on multiple across threads. This can be solving an equation or completing different parts of the same task.
    - After you hit your max clock speed we can only get more efficient with parallelism.
    - Important for AI where we need quick decisions
    - Deadlock! When multiple threads get stuck waiting for a first
- Beyond CUDA: GPU Accelerated C++ for Machine Learning on Cross-Vendor Graphics Cards Made Simple
    - TensorFlow can already be CUDA accelerated, but it is possible to go even faster with Compute shaders
    - The potential speed increases from GPU acceleration are increasing every year, while CPU and FPGA are fairly stagnant. More and more experiments are being run on GPUs every year
    - Many of the current Machine Learning platforms are working on integrating Vulkan's *Kompute* platform
- GPGPU and Artificial Intelligence, The Future of Image and Data Processing for Rugged Computing
    - AI is being increasingly used in real-world applications, such as tactical situations. Image identification for threat realization is one such application
    - When your CPU processing isn't fast enough, where do you turn? GPU acceleration for your AI processes.
    - CPUs can process roughly 5 images per second per watt, while GPUs are processing roughly 65 images per second per watt.