COSC 480 – The Project

COSC 480 – The Subtitle

In groups of 4, you will research and implement a problem described below. Unless otherwise specified, you will do all of your implementations on AWS using CUDA. You may use whatever programming language you wish, and you may use alternate servers, but must provide sufficient documentation in regards to any change (see documentation, below). Each topic has specific tasks, so please be sure to read all of the instructions and ask for clarifications if necessary.

Topics:

1.) Data mining – If simulations are the #1 use of massively parallel general-purpose GPU (GPGPU) computing, data mining is likely to be #2. Turns out churning through lots of independent data is pretty parallelizable. Code task: utilizing one of the data sets here:

https://github.com/caesar0301/awesome-public-datasets

or

https://archive.stsci.edu/kepler/getting_started.html

accomplish some sort of interesting data mining task using CUDA. The Kepler one is particularly interesting because you could discover an exoplanet!

2.) GPU fractals for fun and profit – Fractals (mathematical sets that are repeating regardless of scale) are great targets for CUDA optimization since a number of them features tasks and calculations that are independent from each other. Plus, they're pretty. Code task: create the code for a CUDA implementation and visualization of 3 different fractals.

3.) Utilizing MPI to enable multi-GPU distributed computing resources – As we saw in class, sometimes one GPU is not enough. Well, sometimes one computer is not enough. To enable distributed GPU computing, folks like Oak Ridge National Labs use a system called the Message Passing Interface (MPI). Code task: create 4 MPI and CUDA examples of varying complexity.

4.) AI – there are a number of different AI concepts that can benefit from GPU acceleration. For this topic, I would like you to focus on minmax game playing intelligent agents. Code task: implement Go utilizing a CUDA based game playing AI.

5.) Image processing – as you would expect, GPUs are pretty good at the manipulation of graphics. We can create massively parallel code to do all sorts of useful things – motion detection, object tracking, image filtering, and object detection. You will, as part of this topic, look into and discuss each of these, but for you, a less noble code task awaits. Code task: using CUDA, create the "J. J. Abrams Filter", one that will automatically add scalable lens flares in appropriate spots on any image.

6.) Differences, pros, and cons of OpenCL vs CUDA - We focused on CUDA in this class, but certainly it is not the only API we could have used for general purpose GPU programming. Another option that is useable by a lot more hardware is OpenCL (open compute language). Code task: you must complete both miniprojects and translate 4 examples into OpenCL.

7.) CUDA based implementation of cryptocurrency mining – One of the more interesting trends in computing (and economics) over the last few years, cryptocurrencies are decentralized digital currencies that are secured and created through cryptographic tasks. Turns out, a number of these tasks can be accomplished in a massively parallel way, and thus increase the number of units one can "mine." Code task: create a cryptocurrency mining application utilizing CUDA. Be sure to choose your cryptocurrency carefully so that you can show that the mining works in a reasonable amount of time.

8.) N-body particle simulation – frequently used for physics simulations, an N-body particle simulation is a computation that models a set of particles and then iteratively adjusts them based on some principle. The real trick on this simulation is that to model the movement/state change of one particle, you have to consider all of the other particles in the simulation. Code task: choose one to implement - gravitational simulation, water simulation, or mass crowd simulation.

<u>Documentation information:</u>

For all topics except #3 and #6: You will write a 2500 word count minimum paper as part of your topic. In the paper, you will discuss the current state of the art of your topic in regards to GPGPU programming, provide a full description of the problem you solve with your code task, provide timing comparisons between CPU and GPU executions, and include observations on how specifically GPGPU programming benefits your assigned topic.

For topic #3: For this paper, you will discuss the history of MPI, discuss specific modern CUDA/GPGPU projects that utilize MPI, and provide a full tutorial, including examples, on how to setup and use MPI. This not only includes software aspects, but hardware as well. While there is no word count minimum, your tutorial must be comprehensive enough to be used as a teaching tool in future courses.

For topic #6: For this paper (2500 word count minimum) you will discuss the history of OpenCL, compare and contrast feature sets between OpenCL and CUDA, provide API comparison examples (note that code blocks will not count toward your word count), and include an execution timing comparison on your code tasks and the associated CUDA code. You will also create a full tutorial on how to setup and use OpenCL in AWS. The tutorial does count toward your 2500 word count minimum.

For all topics: at least 4 primary CS sources are required (this means no webpages outside of APIs; papers must be peer-reviewed unless they are whitepapers from the manufacturer; books are good). Please use American Mathematical Society citation style. If you choose (or are forced due to topic) to use a different language or different AWS server, you must provide additional documentation in regards to set up and execution of your code. Pages for citations and custom installation/execution do not count for the word limits above (except as noted above for topic #6).

<u>Presentation information:</u>

Each group will do a 20-minute professional presentation of their project including all code and executions. Groups will be graded on clarity, balance between group members, and coverage of the entire topic.

Source files and documents are due at December 15th, 11:59pm via whatever electronic conveyance you would prefer (email a zip file, dropbox link, github, and so on). Presentations will occur during the final period December 15th at 2pm in Schaefer 160. Group evaluations are due at 2pm December 15th via email. Evals should include a numeric grade (0-100) and a short justification for each grade. Your presentation is worth 10% of your grade for the course, and the project as a whole is worth 40% (code 25%, paper 10%, group evaluation 5%). Failure to attend the presentation will result in a 0 for both aspects of this project and an F in the course.

As has been the case for this course, no late materials will be accepted.