Jake Orben

CSC 470

November 14, 2014

**Introduction:**

I am working on the concept of graphic processing unit or GPU accelerated computational functions. The theory behind this strategy is that a GPU can handle large amounts of threads simultaneously, allowing for high levels of parallelization, permitting large algorithmic issues to be dealt with in a more rapid manner. This strategy will allow me to rapidly calculate the edit distance for large genetic sequences.

**Abstract:**

The work I am doing with Nvidia GPUs will be used to rapidly calculate edit distance for genetic sequences. This will be achieved using parallelization among GPU threads, thus allowing for rapid calculation of large genomic sequences. I believe this to be possible, however, it will be quite challenging.

**Current Progress:**

I am currently progressing at a reasonable pace with my project. The idea of computing on a GPU seems rather simple, however, there are many issues unique to this field of study. For instance, allocating memory becomes much more important, a simple array, if not allocated correctly will crash the entire kernel. This issue is compounded by the fact that one can easily run out of memory when making debugging statements also causing a kernel crash. A kernel crash can happen for various reasons, one of the most common reasons being that you have run out of allocated memory. A print statement in a double nested for loop is often enough to lead to a crash.

There are, however, boundaries to cross. For instance, depending on the generation of a given Nvidia graphics card, the number of threads per block may change. Using the current Kepler 2.0 and 2.1 architectures, a block supports 1024 threads. By contrast, older cards will only support 512 threads per block, drastically changing performance and hindering cross-platform support. This is important, as, when switching between my laptop and my desktop, I have to change a large amount of code, as the GPU on my desktop is around four years old, as a result, the chip architecture is different.

Another issue that I have run across is the rapid development cycle of the CUDA programming language. It is currently on release 6.5, however, it was still in 6.0 a year ago. This can cause confusion, as functions have been added and changed on a rather regular basis. This rapid development cycles makes it difficult to find a source with relevant data. Some of the core code remains the same, however, certain other functions have changed. As a simple example, originally, “prinf” required an import call, however, now, no such call is necessary.

Thus far, I have the algorithm running on one block with one thread. That being said, it is unstable, returning inconsistent answers. Once this particular issue is cleared, it should be a simple matter of introducing multiple threads to the algorithm, making the work parallel. This can be achieved by viewing the number of input characters and dividing it by the threads possessed, from there, the number of blocks needed can be extrapolated, allowing for parallelization across a wide number of edit distance tables. The performance benefit cannot currently be determined, however, I believe that it will be vast, made even more so by utilizing a more powerful GPU than the one I am currently testing with.

I have been using several sources for my work, many coming directly from Nvidia. Nvidia has a small series of basic tutorials that show the outline of what needs to be done in order to parallelize a simple program. From this information I have begun to convert the C code into CUDA code that can be run in parallel, theoretically improving performance.

Still more of my information came from other colleges, too numerous to list in fact, many of which have done research into the concept of GPU accelerated algorithms. Unfortunately few had sample code, much of the work done is simply theoretical. I found these papers somewhat useful, in concepts; however, they often talk about the 3-dimensional nature of blocks, which is a rather complicated idea to get one’s head around. I have recently ordered a book detailing CUDA programs from basic to advanced, which is scheduled to arrive on Friday and will be discussed in my final report.

There is still a great deal of trial and error to do before this program is finished. That being said, I believe that I might be able to achieve it before the deadline. It will be a challenge, but one that I believe I can overcome.