Jack Wadden
Donnie Newell
Homework 3
18Oct2011

Findings

The most challenging aspect of this assignment was performing the reduction between blocks. We spent some time trying to track down off-by-one errors where we were using either an incorrect number of blocks, or we were mis-managing the threads. For such a data parallel operation, the gpu is clearly superior in performance. The downside is that the memory transfer from the host memory to device memory is a bottleneck. On some of our runs the memory transfer to the GPU took just as long as the CPU run. As Professor Skadron noted in an email to the class, we observed somewhat large differences in mean and especially standard deviation as the number of elements topped one million or so. We believe this is due to single-precision floating point inaccuracies. Lastly, we encountered some memory allocation errors when working on barracuda13, so we moved to tesla and it had enough memory on the card for us to allocate our arrays.

Algorithms

Initially, we took the approach that each thread would operate on 2 adjacent elements in the arrays, and then write the result to array[blockIdx.x]. This approach required too much synchronization and, after looking at the slides, we followed the reduction example Michael Boyer gave. We calculated the results by operating on value i and i + th, and writing the result to the first value's location. After reducing the problem to one element in the first element of the block, we wrote the result to the device memory at the block id index. We repeated this reduction until we had one answer in the first element of the device memory array. For the mean, we just calculated the sum during the reduction and divided by N at the end. For the standard deviation, we first calculated the square of the difference between the mean and the element, wrote that value to shared memory, and then we performed the reduction by summing the squared differences. Finally, we took the square root of the average after the sum was complete. We split the standard deviation into a separate kernel call simply because it was more straightforward and we had spent enough time working on the first half of the lab and it seemed more difficult to calculate the mean and then continue in the same kernel for standard deviation. The result of this was a negative impact on performance for the GPU kernels versus the CPU.

Optimization

There was not much optimization performed. Most of our effort was in getting the reductions to be correct. We did use shared memory, which is much faster than device memory. We also calculated the min,max, and mean in the same kernel call, but I would not say that this was an optimization that we discovered, because we were explicitly suggested in the homework write-up.