Alperen Degirmenci
CS 205 - HW 3
10/26/12

Problem 2

Performance Benchmark:

Our aim should be to have enough threads in one block so that we don't waste any of the data transferred from the gloabal memory to the shared memory. This means we can have 128, 256, 512, or 1024 threads per block (1024 threads per block is the maximum for Nvidia C2070). However we should also consider the capacity of the shared memory. We have n bytes of z, of x, and of y, plus alpha. Having 1024 threads might require more memory than what is available in the shared memory, therefore it is better to have 256 or 512 threads per block. We can then determine the number of blocks necessary by dividing the number of computations to number of threads, which is $2^{18}/256 = 1024$. Table 1 shows the performance of different thread and block size combinations. Results confirm that (256, 1024) and (512, 512) are the fastest, together with (128, 2048), which makes sense since data is transfered from global memory to shared memory in chunks of 128. Having extra blocks is detrimental to performance. In comparison, it takes 1091 μs to perform the same SAXPY operation on the CPU.

Threads	Blocks	Time μs
8	32768	348
64	4096	70
128	2048	61
256	1024	61
512	512	61
512	1024	65
1024	256	71
1024	512	81
1024	1024	101

Table 1: SAXPY operation performance on the GPU.

Performance Model:

By transfering various sizes of data, I obtained the following results:
Latency from CPU to GPU = 297.784805 μ s
Latency from GPU to CPU = 112.473965 μ s
Bandwidth from CPU to GPU= 8307.884282 MBps
Bandwidth from CPU to GPU = 4516.147420 MBps
Based on these results, the performance model can be expressed as:

$$T = 3 * \left(L_{GPU}^{CPU} + \frac{4 * N}{BW_{GPU}^{CPU}} \right) + M * t_{SAXPY} + \left(L_{CPU}^{GPU} + \frac{4 * N}{BW_{CPU}^{GPU}} \right) \tag{1}$$

where L_{GPU}^{CPU} is the latency from CPU to GPU, BW_{GPU}^{CPU} is the bandwidth from CPU to GPU, t_{SAXPY} is the GPU computation time per SAXPY operation, M is the number of times the SAXPY operation is performed, and N is the size of the vectors. The reason we multiply N by 4 is to account for the fact that the vector elements are each 32 bits, which is 4 bytes. Dividing bytes by MBps yields μs.

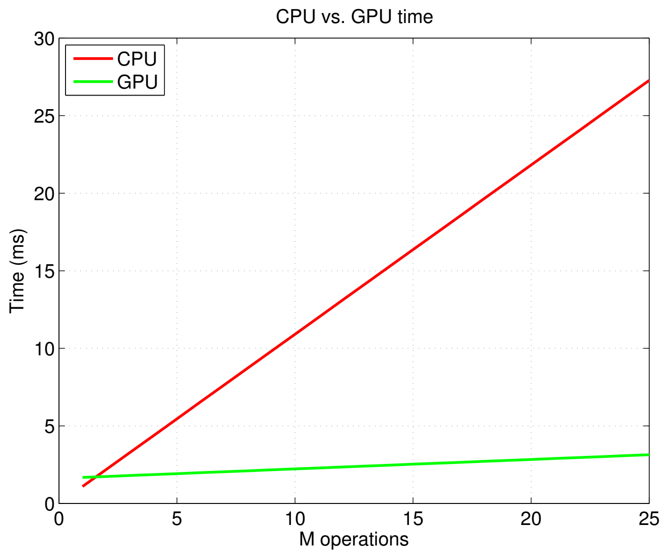

Figure 1: Overall computation time of the SAXPY operation on the CPU and GPU when $N = 2^{18}$

Using the results of our performance tests, we can determine the break-even point. Figure 1a shows the theoretical overall computation time for the SAXPY operation on the CPU and GPU when $N = 2^{18}$ and $M = [1, 25]$. The figure shows that for M larger than 1, we should use the GPU.