Hands-on Lab 2: Block Sizes Results

Name: Luke Goodall
Marks: 10

2D Block size	STEP 3		STEP 4		STEP 5
	2D Grid size (automatically determined)	Kernel execution time (ms)	1D Grid size	Kernel execution time (ms) (1 datum per thread)	Kernel execution time (ms) (16 data per thread)
32x32	128 x 128	2.070355	16384 x 1	2.088070	16.573593
32x16	128 x 256	2.044768	32769 x 1	2.072077	13.861440
16x32	256 x 128	2.075744	32768 x 1	2.091424	13.709043
16x16	256 x 256	2.081344	65536 x 1	2.099629	11.940448
24 x 24	171 x 171	2.185798	29128 x 1	2.326547	13.918662
20 x 20	205 x 205	2.295789	41944 x 1	2.395936	14.994176
28 x 28	147 x 147	2.338496	21400 x 1	2.398534	14.615993
8 x 8	512 x 512	2.226426	262144 x 1	2.192768	13.652941

Observations:

1. It would seem that blocks with dimensions that are powers of 2 are the fastest, especially in the range 32 - 16. For some reason 32 x 16 seems to perform the best. (exceptions: when threads performing 16x work)
2. The 2D grids seemed to perform slightly faster than the 1D grids
3. When threads performed 16x as much work they were an order of magnitude slower.
4. It would seem that we are actually performing array addition in the code, instead of matrix addition (not sure about this point?)

Conclusion:

Optimal conditions for kernel conditions are to try and spread the work evenly among threads, from observation 3. Block dimensions work best in powers of 2, and when there is a balance between the number of blocks and the number of threads per block from observation 1. Representing the data in a different grid dimension can have a slight effect on performance, but not as significant as other factors from observation 2.