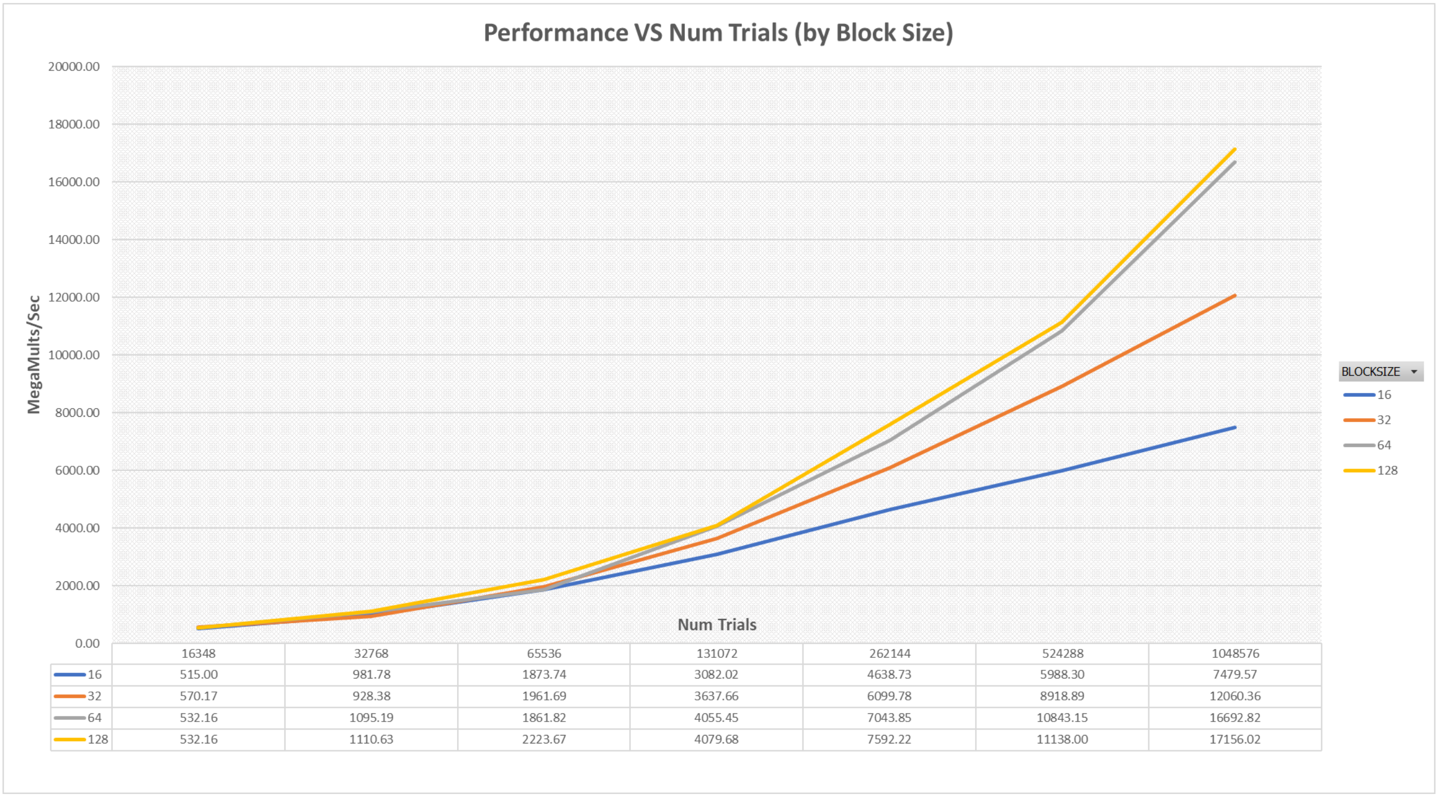
Clinton Hawkes
hawkesc@oregonstate.edu
CS475-400
Project #5

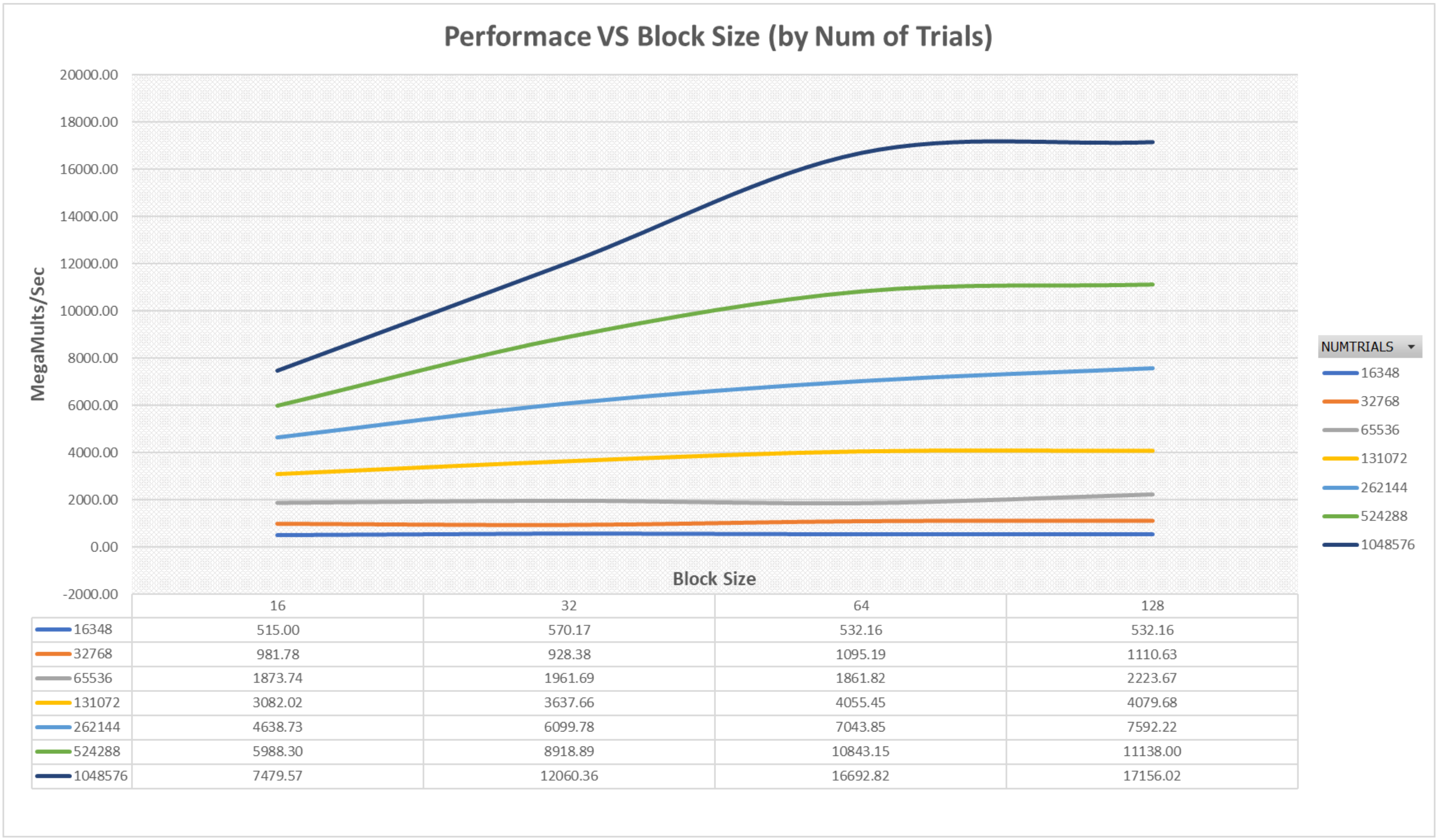
1. I ran this program on the DGX system at OSU. Your instructions/example made it simple to compile and get my results.

2. Here is the table of my results:

BLOCKSIZE	NUMTRIALS	PERFORMANCE	PROBABILITY
16	16348	514.995	42.43%
32	16348	570.173	41.53%
64	16348	532.1615	42.35%
128	16348	532.1615	41.23%
16	32768	981.7833	41.68%
32	32768	928.3771	42.10%
64	32768	1095.1871	41.78%
128	32768	1110.6291	42.41%
16	65536	1873.7419	42.33%
32	65536	1961.6858	41.69%
64	65536	1861.8182	41.95%
128	65536	2223.6699	41.96%
16	131072	3082.0166	41.99%
32	131072	3637.6555	41.99%
64	131072	4055.4455	42.29%
128	131072	4079.6814	41.93%
16	262144	4638.7317	42.03%
32	262144	6099.7767	42.04%
64	262144	7043.8521	41.78%
128	262144	7592.2154	42.13%
16	524288	5988.3038	41.90%
32	524288	8918.8894	42.09%
64	524288	10843.1504	41.97%
128	524288	11138.0012	42.18%
16	1048576	7479.5708	42.09%
32	1048576	12060.3608	42.00%
64	1048576	16692.8167	42.06%
128	1048576	17156.021	42.01%



	16348	32768	65536	131072	262144	524288	1048576
16	515.00	981.78	1873.74	3082.02	4638.73	5988.30	7479.57
32	570.17	928.38	1961.69	3637.66	6099.78	8918.89	12060.36
64	532.16	1095.19	1861.82	4055.45	7043.85	10843.15	16692.82
128	532.16	1110.63	2223.67	4079.68	7592.22	11138.00	17156.02



	16	32	64	128
16348	515.00	570.17	532.16	532.16
32768	981.78	928.38	1095.19	1110.63
65536	1873.74	1961.69	1861.82	2223.67
131072	3082.02	3637.66	4055.45	4079.68
262144	4638.73	6099.78	7043.85	7592.22
524288	5988.30	8918.89	10843.15	11138.00
1048576	7479.57	12060.36	16692.82	17156.02

3. The patterns I see when looking at the graphs above are:
    a. Performance increases as the number of trials or calculations increases. This is true for all the different block sizes.

    b. Performance increases as the block size increases. Keep in mind that the block sizes are multiples of 32 (except 16, more on that later). I do notice that the performance gained from using a block size of 128 rather than 64 is minimal. The curves in the first graph basically follow each other.

4. I will explain the two points made in question 3 with a corresponding bullet. (4a answers 3a)
    a. I think the reason the performance increases as the number of calculations increases is because there is a lot of setup that goes into performing calculations on the GPU rather than the CPU. This setup takes time, and if there are only a few thousand calculations to be performed, there will be a much larger percentage of time used on setup than there will be used on the calculations. This drives our megamults/sec down.
    When we have millions of calculations to be performed, the percentage of time spent on setup becomes much smaller. This causes the performance figure to increase.

    b. I think the performance increases as the block size increases from 16, because 16 is not a multiple of 32. Since it is not a multiple of 32, there is work that could get done, but the block size prevents the utilization. This was stated in the lecture.

    As the block size increases from 32 to 64, we see another big increase in performance. We were told that there is a sweet spot for block size, and that we needed to find it. I don't know of any calculation I can do to find this "sweet spot", other than running tests like we did in this project. It appears that a block size of 64 is right about there. You can see that there is little gained from increasing the block size further to 128, but it is a slight gain, so I would say the best block size would be 128.

5. We learned that the 32 threads in CUDA constitutes a WARP, that is, these 32 threads execute the same instruction. This is why we want our block size to be a multiple of 32. If we use a number that is not a multiple of 32 (like 16 in our problem), we have capacity that is not used. Since the block size of 16 is not a multiple of 32, there are 16 idle threads every time a block executes an instruction. This is why the performance is lower when block size equals 16.

6. These performance results are MUCH greater than my results for project 1. I was able to get 240 megatrials/sec in project 1 using my 6 core CPU. When using the DGX system, I was able to get over 17000 megatrials/sec. That is just insane! That is a 71x performance increase when switching to the V100 GPU. I looked up the specs on the V100, and it has 5120 CUDA cores (threads), so I can see why there is such a large increase in performance.

7. Using a GPU to handle the calculations for a problem can substantially increase your performance, assuming your problem has the types of calculations that the GPU can handle and you have enough calculations to make it worth the use of a GPU. CPUs are multipurpose and can handle a wide range of instructions. GPUs are more narrowly focused, and they excel at array/matrix type calculations that do not depend on one another and can be performed simultaneously. If your problem can take advantage of the way a GPU processes information, you can see huge benefits by using a GPU for your calculations.