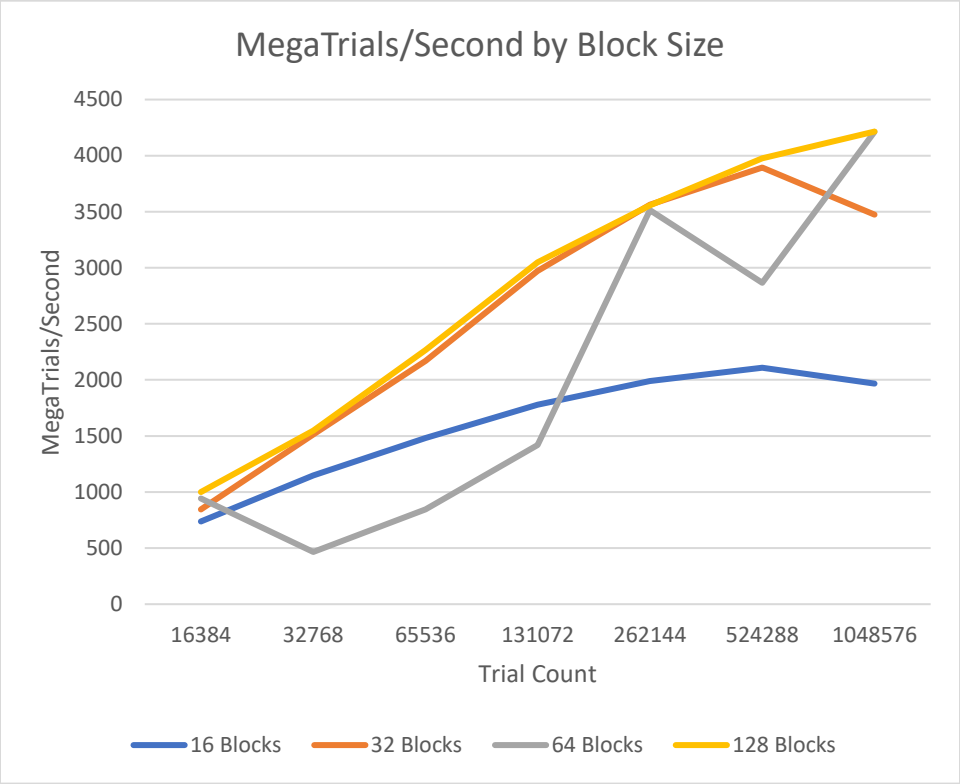
# Cuda Monte Carlo Simulation

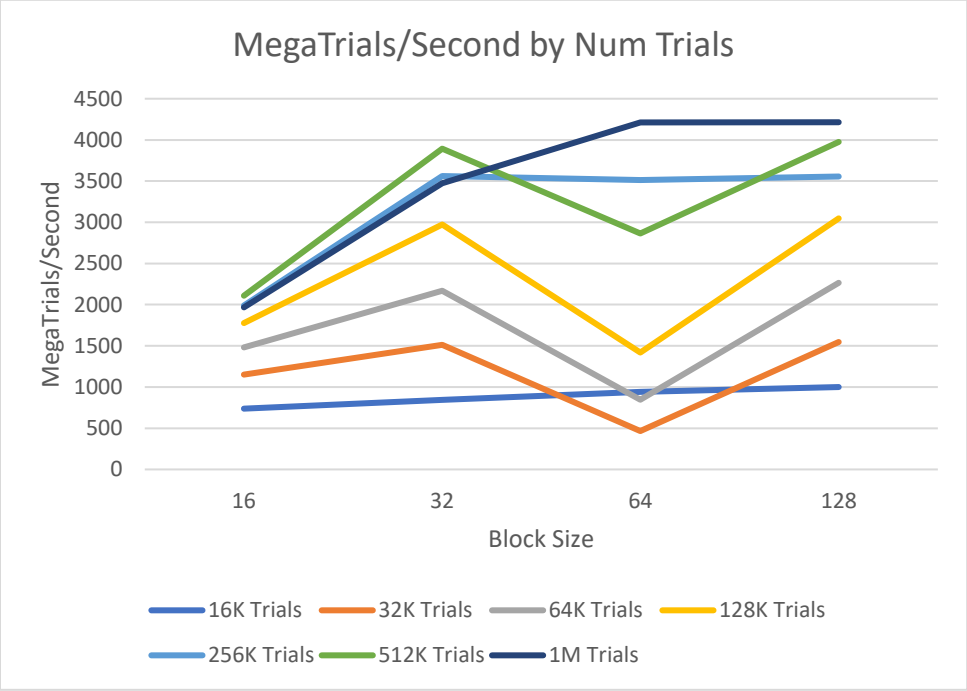
By Logan Saso

1. Machine Details
    a. Intel i7 8086K
        i. 4ghz base (5ghz single-core boost) clock
        ii. 6 cores, 12 threads
    b. 32 GB Corsair Dominator Platinum RAM
        i. Clocked at 2600MHZ
    c. NVIDIA Geforce GTX 1080 Ti EVGA FTW 3 Edition
        i. Clocked at 2ghz
        ii. Ran on Ubuntu 19.10 w/ CUDA installed
2. Table and Graphs

| BLOCKSIZE | TRIALS | MegaTrials per Second | Probability |
|---|---|---|---|
| 16 | 16384 | 737.752175 | 41.986084 |
| 16 | 32768 | 1149.270516 | 42.041016 |
| 16 | 65536 | 1480.838699 | 41.896057 |
| 16 | 131072 | 1777.777715 | 42.159271 |
| 16 | 262144 | 1989.798437 | 42.105865 |
| 16 | 524288 | 2108.622902 | 41.881561 |
| 16 | 1048576 | 1967.22094 | 41.982269 |
| 32 | 16384 | 844.884466 | 42.193604 |
| 32 | 32768 | 1512.555448 | 42.166138 |
| 32 | 65536 | 2169.49146 | 42.321777 |
| 32 | 131072 | 2972.423769 | 42.071533 |
| 32 | 262144 | 3561.739051 | 41.996765 |
| 32 | 524288 | 3893.536204 | 41.932678 |
| 32 | 1048576 | 3474.130579 | 42.044067 |
| 64 | 16384 | 942.909757 | 41.809082 |
| 64 | 32768 | 465.878077 | 42.044067 |
| 64 | 65536 | 846.28098 | 42.134094 |
| 64 | 131072 | 1418.773766 | 42.305756 |
| 64 | 262144 | 3512.864401 | 42.004776 |
| 64 | 524288 | 2865.83873 | 42.086029 |
| 64 | 1048576 | 4212.366703 | 42.063522 |
| 128 | 16384 | 1000.000003 | 42.175293 |
| 128 | 32768 | 1546.827849 | 42.129517 |
| 128 | 65536 | 2265.486688 | 41.667175 |
| 128 | 131072 | 3047.619072 | 41.773987 |
| 128 | 262144 | 3555.55543 | 42.16156 |
| 128 | 524288 | 3975.734096 | 42.039108 |

| | | | |
|---|---|---|---|
| 128 | 1048576 | 4214.533854 | 41.948605 |




3. The performance curves are mostly the same, with the exception that 16 and 64 have lower performance, on average, than 32 or 128 blocks. Unsurprisingly, when the GPU is given more trials it can get a better performance because it's over a longer period of time (less bad apples can ruin the batch). However it is interesting to see that 32 blocks has a slight dip at 1M trials.

4. I don't see a reason why the patterns *shouldn't* look this way. I'd expect that 16 blocks isn't enough to use the full power of the 32-thread CUDA cores. I am surprised that 64 blocks seems to have a lower performance than 32 blocks, and even 16 blocks in some cases. I'm wondering if that's because of the way the GPU divides up resources among different work groups. In that case, though, I'd expect to see 128 blocks also behave similarly but it doesn't.
5. Blocksize of 16 is worse because it only allows the GPU to use about half of it's performance, since a single CUDA core is 32 threads we're effectively not using half of each one.
6. These performance results are about 7 times faster than the CPU-bound performance of project 1. This is because this problem is particularly parallelizable, so it's easy for a GPU, which is designed for parallel processing, to finish the tasks faster. If this was a synchronous task I'm sure the CPU would be faster.
7. To use a GPU correctly, you need to use it with highly parallelizable tasks. It wouldn't be great for doing a list of things, but it's absolutely critical for very fast parallel tasks. Consider it's main purpose, video rendering (either live for games, or slowly for movies and media), as a perfect example of its purpose. Each pixel doesn't rely on others to render. Frankly, I'm surprised my GPU was only capable of 4k mega trials or so considering it's capable of rendering a full 3D scene over 250 times a second.