Adam Wright

wrighada

CS-475

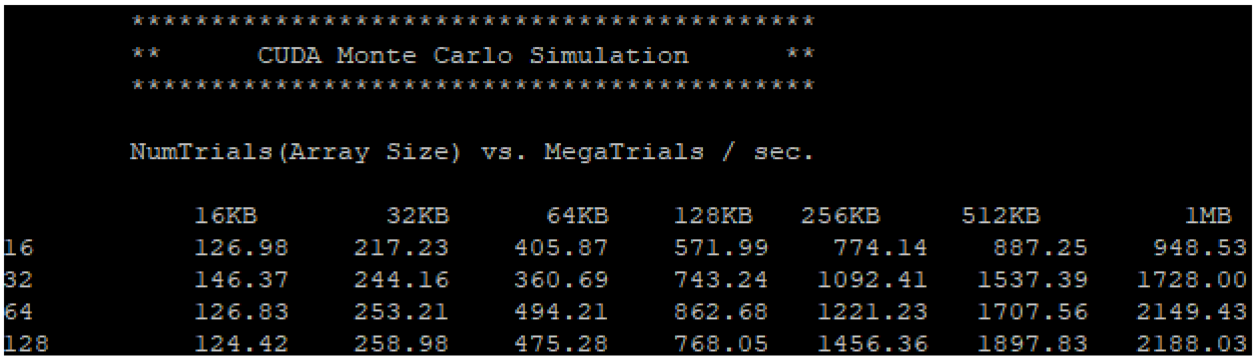
Project-5 – GPU Monte Carlo Simulation

1. What machine was the project run on?

The program was prepared on the OSU Rabbit server and the final results were run on OSU's DGX batch server.

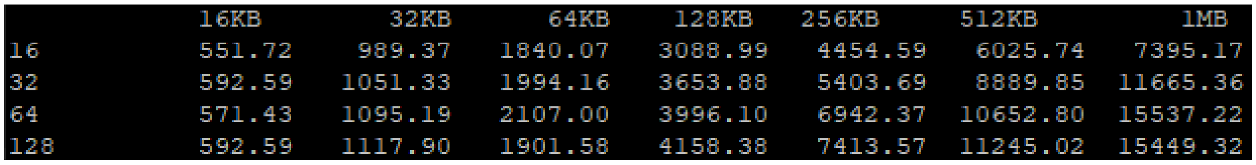
2. Show the table and the two graphs (Rabbit & DGX results tables both included)

Rabbit server results table



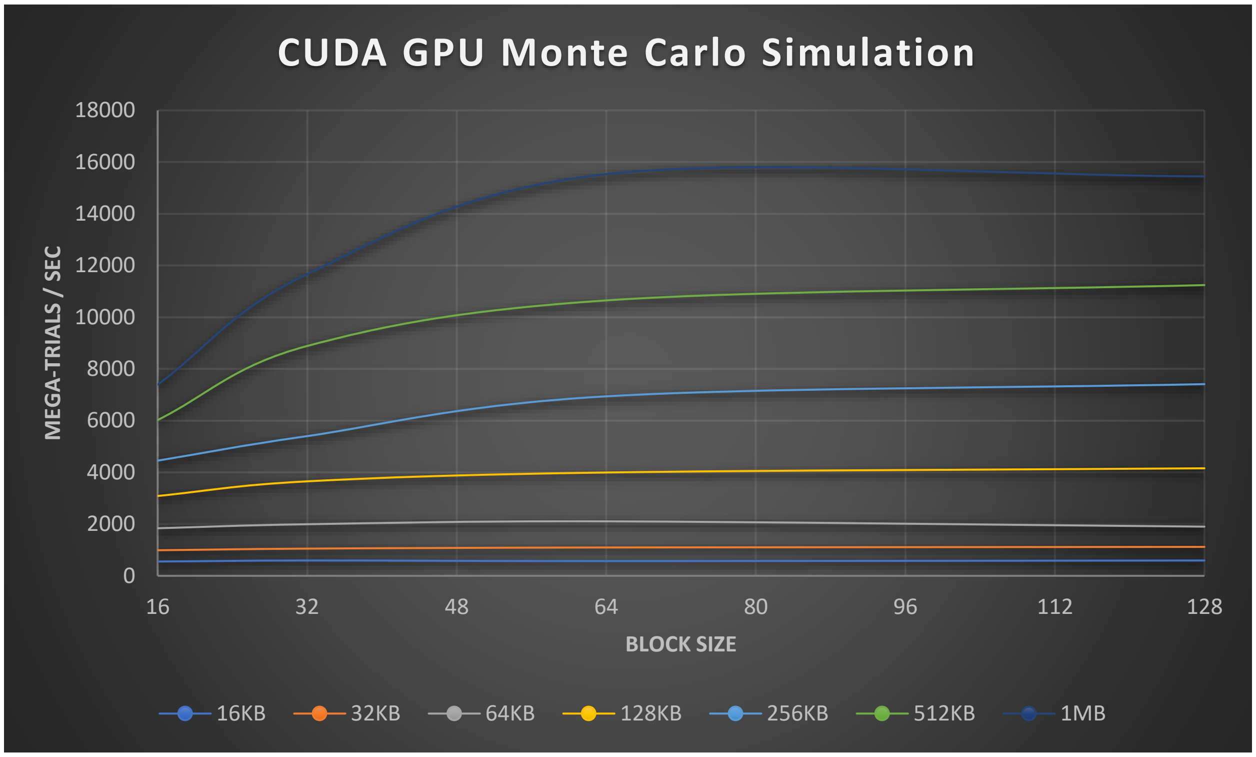
```
        ********************************************
        **      CUDA Monte Carlo Simulation       **
        ********************************************

        NumTrials(Array Size) vs. MegaTrials / sec.

            16KB        32KB       64KB     128KB   256KB      512KB       1MB
16          126.98    217.23     405.87    571.99    774.14    887.25    948.53
32          146.37    244.16     360.69    743.24   1092.41   1537.39   1728.00
64          126.83    253.21     494.21    862.68   1221.23   1707.56   2149.43
128         124.42    258.98     475.28    768.05   1456.36   1897.83   2188.03
```

DGX server results table

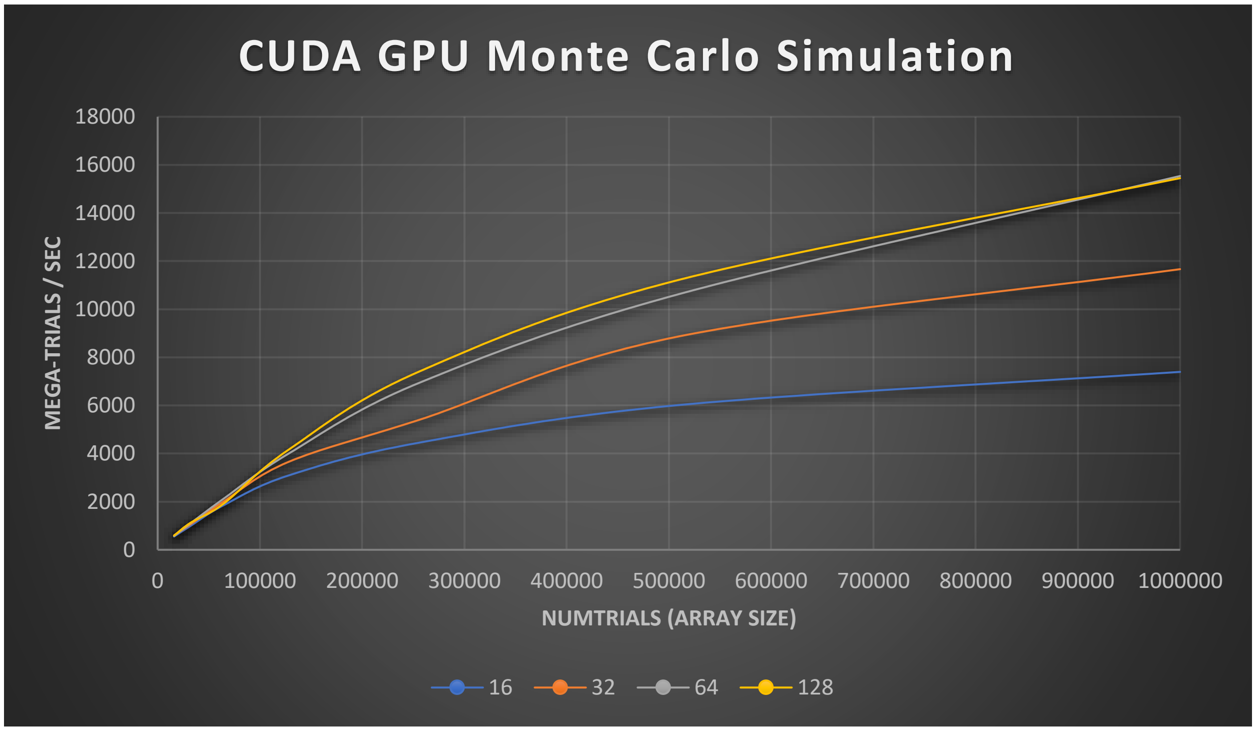


```
            16KB        32KB       64KB     128KB   256KB      512KB       1MB
16          551.72    989.37    1840.07   3088.99   4454.59   6025.74   7395.17
32          592.59   1051.33    1994.16   3653.88   5403.69   8889.85  11665.36
64          571.43   1095.19    2107.00   3996.10   6942.37  10652.80  15537.22
128         592.59   1117.90    1901.58   4158.38   7413.57  11245.02  15449.32
```

DGX - Performance vs. Block size



DGX - Performance vs. NUMTRIALS (array size)



3. What patterns are you seeing in the performance curves?

From the graph of block size versus performance it is clear that for array sizes larger than 256KB. and particularly at 1MB., that a larger block size of 64 or even better 128 is much more performant. The smaller block sizes of 16 and 32 are still about equal for array sizes of 64KB. and smaller.

From the graph of NUMTRIALS (array size) versus performance it is clear that as shown is the other graph, that the larger block size is the best performer in all cases. Since, there is no clear disadvantage to the larger block size, it seems that the block size of 128 is the best overall choice for this simulation.

4. Why do you think the patterns look this way?

Since all of the block sizes perform the same with array sizes of fewer than 64KB., then it seems that having filled warps is not necessary in the smallest array size. At above 100KB the different block sizes must be diverging because the larger block sizes are more efficient. It seems that with block sizes of 64 and 128 that there may be fewer times where the warps need to be swapped out as current calculations are threads are blocked awaiting data.

5. Why is a BLOCKSIZE of 16 so much worse than the others?

Since CUDA works with units of 32 threads called warps, the 16 thread blocksize leads to half-filled warps. This means that half of the possible computing capacity is being left unused, though the performance does not doulble with a 32 thread block, rather it doubles at the 64 thread block. To get the maximum performance it is important that the threads per block value is a multiple of 32 and 16 isn't.

6. How do these performance results compare with what you got in Project #1? Why?

The best performance from project #1 was with 6 threads and a 1MB array size, where the result was 250 Mega-Trials per second. The best equlvalent performance from the CUDA GPU implementation was 15,449 Mega-Trials per second with a block size of 128 and a 1MB. array. The CUDA implementation is nearly 62 times better in performance.

The CUDA implementation is much more efficient because many more operations are being done in parallel at the same time, as compared to the CPU implementation. On a CPU there were only 6 simultaneous operations in my best performing trial, because there is only one ALU per CPU on which to do the arithmetic. In the GPU implementation, there are thousands of floating point units so that each of the many threads has it's own. This means that many more arithmetic operations can be done simultaneously.

7. What does this mean for the proper use of GPU parallel computing?

This result shows that in cases where a simple arithmetic operation needs to be done many times on a set of values, that a graphics card is able to do many more of those operations in the same amount of time as a CPU can do. For large data sets with many small operations which can be done in parallel, the GPU is much more efficient. The main downside would be that a large rig of current top level GPUs is very expensive, but from the results found on the Rabbit GPU rig, the results were still almost 20 times greater than a 6 thread CPU implementation, but with less than the highest possible quality hardware. Just as RAID was originally intended to be a redundant array of inexpensive disks, there would be a middle ground where a certain spec of GPU in an array could be the best possible solution for data processing as part of a larger software system.