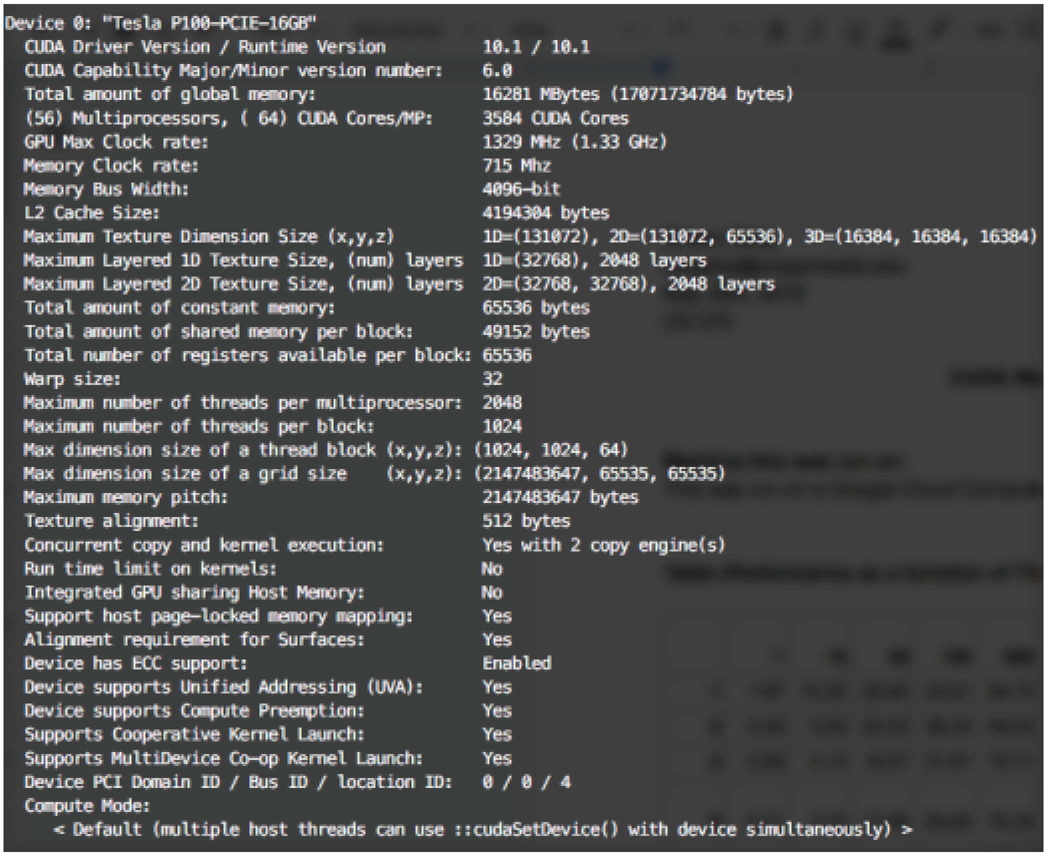
Shawn Berg
bergsha@oregonstate.edu
May 30th, 2019
CS 475

Project 6
CUDA Monte Carlo Simulation
Writeup

Machine this was run on:

This was run on a Google Cloud Compute Instance with the following GPU specifications:



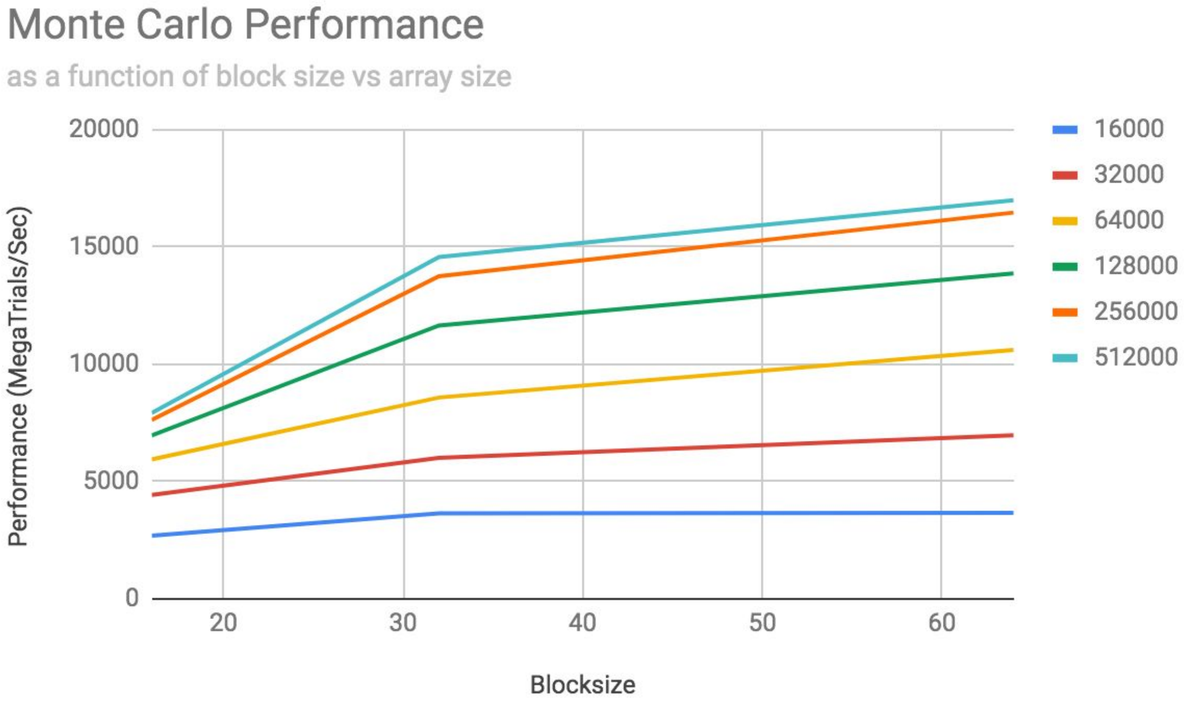
```
Device 0: "Tesla P100-PCIE-16GB"
  CUDA Driver Version / Runtime Version          10.1 / 10.1
  CUDA Capability Major/Minor version number:    6.0
  Total amount of global memory:                 16281 MBytes (17071734784 bytes)
  (56) Multiprocessors, ( 64) CUDA Cores/MP:     3584 CUDA Cores
  GPU Max Clock rate:                            1329 MHz (1.33 GHz)
  Memory Clock rate:                             715 Mhz
  Memory Bus Width:                              4096-bit
  L2 Cache Size:                                 4194304 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 4
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

Table (Performance as a function of Threads vs Number of Trials)

	16000	32000	64000	128000	256000	512000
16	2698.04	4426.93	5949.37	6964.52	7631.48	7926.13
32	3646.71	6009.25	8584.43	11654.67	13758.47	14576.86
64	3669.19	6975.45	10617.4	13873.47	16469.38	16990.55

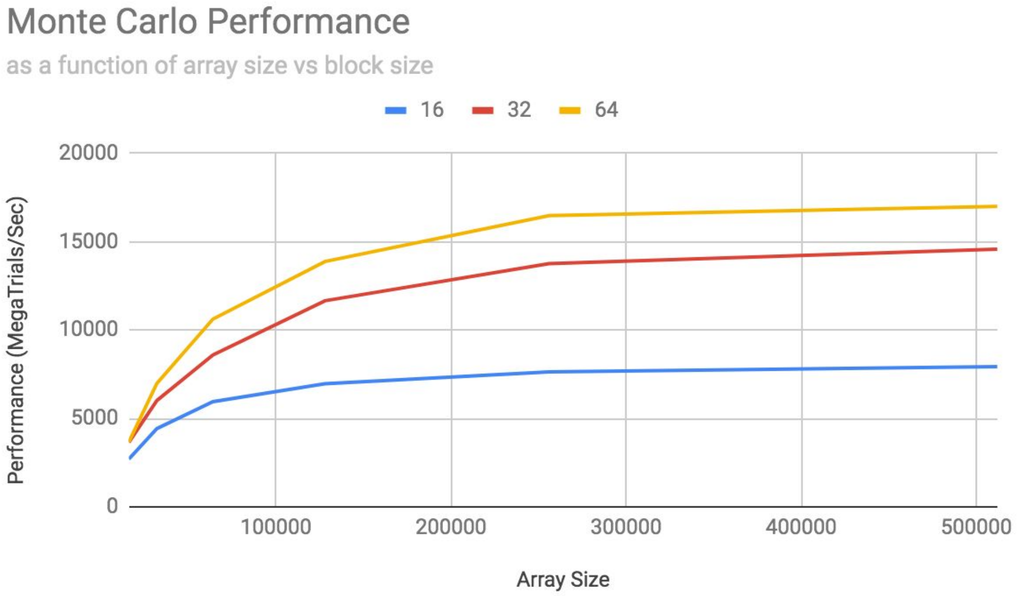
Units = MegaTrials/Sec

Graph 1 (MegaTrials per Second vs Blocksize)



Each line represents array size

Graph 2 (MegaTrials per Second vs Array Size)



Each line represents blocksize.

Additional Commentary

1. **What patterns are you seeing in the performance curves?**

   The performance curves for both graphs show a general increase in performance as array size and blocksize increases.

2. **Why do you think the patterns look this way?**

   The patterns appear this way because as array size and blocksize increase, more of the GPU is being utilized in the computations, thus an increase in performance is displayed.

3. **Why is a BLOCKSIZE of 16 so much worse than the other two?**

   The blocksize of 16 is worse than the other two because the warp size is 32. This means when using a block size of 16, 32 threads are still allocated to the problem, but only 16 are being used, leaving the other 16 unused.

4. **What does that mean for the proper use of GPU parallel computing?**

   To properly utilize the GPU in parallel computing, not only does the problem size need to be large enough, but the intended division of labor must also take into account the relevant sizes of the blocks on the target device to acquire all of the potential performance.