Evan Lang

Yang Lu

Lab 8 Report  
GPU USED: Tesla V100  
CPU USED: Intel Xeon Gold 6132 14 Core @ 2.6GHz

*All error checking is performed by comparing the produced cpu and gpu final matrixes. In all runs the results of each matrix matched exactly. In our initial versions of the .cu code we would experience minute errors of around .00001% on the most massive numbers. However, our most recent runs display 0 error when comparing the 2 final matrices.*

1. **mmm\_global.cu**  
   The data in all matrices used in this and future portions of the lab are filled by a random integer between 1 and 100.
   1. 2048x2048 Matrix
      1. Total Execution Time of GPU MMM: 0.160686 Seconds  
         MMM only execution time: 0.000043 Seconds  
         Standard Serial Execution time: 52.796551 Seconds  
           
         Overall the execution time for a 2048x2048 matrix is significantly worse for the serial code (by many orders of magnitiude). The overhead of the data transfers are also significantly larger than the total MMM runtime by 4 orders of magnitude.
      2. Largest difference between 2 values found: 0  
           
         Every entry in the CPU and GPU matrices matched after both completed their runs. There was 0 observed error.
   2. 1024x1024 Matrix
      1. Total Execution Time of GPU MMM: 0.174850 Seconds  
         MMM only execution time: 0.000041 Seconds  
         Standard Serial Execution time: 6.334954 Seconds  
           
         Once again there was a large difference in performance between the CPU serial and GPU MMM’s. However, the performance of the CPU is significantly better at this matrix size compared to 2048 as its cache is not completely saturated at all levels. Once again the data transfer and other procedure’s overhead was much greater than the MMM’s actual runtime (again around 4 orders of magnitude in difference).
      2. Largest difference between 2 values found: 0  
           
         Every entry in the CPU and GPU matrices matched after both completed their runs. There was 0 observed error.
2. **mmm\_shared.cu**  
   The data in all matrices used in this and future portions of the lab are filled by a random integer between 1 and 100.
   1. 2048x2048 Matrix
      1. Total Execution Time of GPU MMM: 0.152987 Seconds  
         MMM only total execution time: 0.000016 Seconds  
         Standard Serial Execution time: 50.720207 Seconds
      2. Overall the execution time for a 2048x2048 matrix is significantly worse for the serial code (by many orders of magnitiude). The overhead of the data transfers are also significantly larger than the total MMM runtime by 4 orders of magnitude. It should also be noted utilizing shared memory decreased the MMM runtime by ~50%, a significant improvement
      3. Largest difference between 2 values found: 0  
         Every entry in the CPU and GPU matrices matched after both completed their runs. There was 0 observed error.
   2. 1024x1024 Matrix
      1. Total Execution Time of GPU MMM: 0.137420 Seconds  
         MMM only execution time: 0.000014 Seconds  
         Standard Serial Execution time: 6.424279 Seconds  
           
         Once again there was a large difference in performance between the CPU serial and GPU MMM’s. However, the performance of the CPU is significantly better at this matrix size compared to 2048 as its cache is not completely saturated at all levels. Once again the data transfer and other procedure’s overhead was much greater than the MMM’s actual runtime (again around 4 orders of magnitude in difference).
      2. Largest difference between 2 values found: 0  
         Every entry in the CPU and GPU matrices matched after both completed their runs. There was 0 observed error.
3. **mmm\_sh\_unroll.cu** *based on mmm\_shared.cu, ran with a loop unrolling factor of 16*
   1. 2048x2048 Matrix
      1. Total Execution Time of GPU MMM: 0.153080 Seconds  
         MMM only total execution time: 0.000013 Seconds  
         Standard Serial Execution time: 6.424279 Seconds  
           
         Once again the execution time for a 2048x2048 matrix is significantly worse for the serial code (by many orders of magnitiude). The overhead of the data transfers are also significantly larger than the total MMM runtime by 4 orders of magnitude.The loop unrolling resulted in a small but consistent gain in performance, shrinking the dedicated MMM time for the gpu by 3 microseconds.
      2. Largest difference between 2 values found: 0  
         Every entry in the CPU and GPU matrices matched after both completed their runs. There was 0 observed error.
   2. 1024x1024 Matrix
      1. Total Execution Time of GPU MMM: 0.131514 Seconds  
         MMM only execution time: 0.000013 Seconds  
         Standard Serial Execution time: 6.274757 Seconds  
           
         This follows all the trends of previous runs, however it should be noted that the runtimes for both the 2048 and 1024 matrix are the same when using loop unrolling with a factor of 16. This goes against the previous trend in the non loop unrolling runs, where generally the 1024 had somewhat of a GPU MMM only advantage over 2048. Once again the serial fails to impress.
      2. Largest difference between 2 values found: 0  
         Every entry in the CPU and GPU matrices matched after both completed their runs. There was 0 observed error.
   3. Increasing the ThreadBlock size seems to have diminishing returns. The time does generally start trending downwards, however, it is extremely minute and is within an expected range of error. Also, it should be noted that the ThreadBlock can only be increased until the GPU’s shared memory’s max size is reached.
   4. Varying the matrix size seemed to have a minimal effect on the run time of the device code. However, the overall gpu overhead time seemed to go down dramatically as matrix sizes got smaller.
   5. Matrix sizes that aren’t multiples of the Thread block size seems to create issues
   6. By using warp tiling, MMM can experience an extreme speed up. This is done through allowing warp level parallelism. Source: <https://siboehm.com/articles/22/CUDA-MMM>
   7. Breaking the memory coalescing, makes the blocks perform in a serial manner. Hence, it will perform much much worse than it’s coalesced equivalent.   
        
        
      **The printed outputs of all of our runs are contained in the file data.txt. In each run you may see a line printed saying “true/false: 1”. This is the output of our error checking code. A 1 indicates the CPU MMM and GPU MMM output matrices match. A 0 indicates that they don’t match**