WITS
UNIVERSITY

School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS4040A & COMS7045A Assignment 2 – Report

Marc Marsden
1437889
BDA Hons

May 14, 2020

Introduction

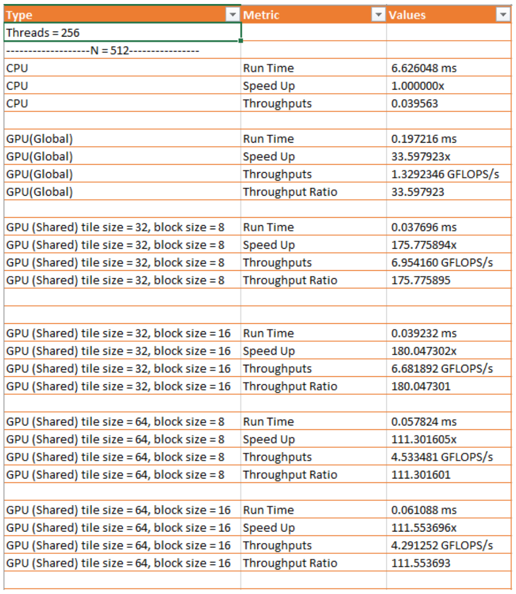
This report compares the performance of CUDA parallel computing and serial computing by applying them to three different mathematical operations. The first section will compare the performance of transposing a matrix. The second section looks at the performances of vector addition with reduction and the final section will analyse the performances on matrix multiplication. The results obtained in this report were achieved by making use of the CUDA Toolkit release 10.1, V10.1.243, running the code on Google Colab, which has the following specs:

- **CPU**: Intel Xeon @ 2.00GHz with 40MB L3 cache and 1 available core with 2 threads.
- **GPU**: Nvidia Tesla T4, with 2560 cores, 8.1 TFLOPS Single-Precision, 16GB GDDR6 VRAM.
- **RAM**: 13GB
- **Disk**: 34GB

Problem 1: Transposition

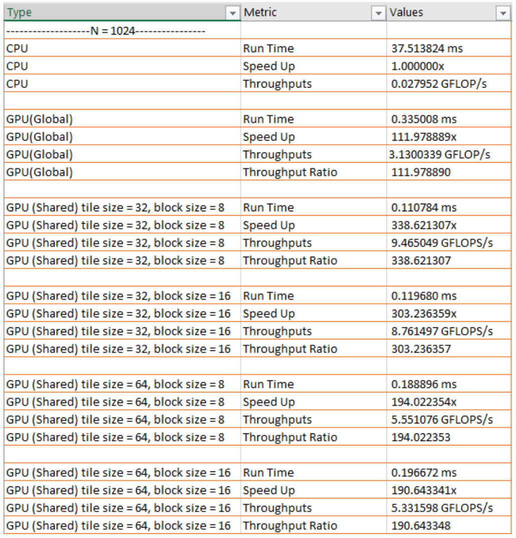
This section compares the of transposing a matrix serial version with the global memory version and shared memory version of CUDA parallel programming. In the global kernel, the block size was kept constant at 256 threads and the grid size was calculated as (N*N)\num_threads, where N is the size of the data and num_threads is the block size. For the shared memory kernel a two dimensional grid size and two dimensional block size were used, where the grid size was defined as (N\tile_size,N\tile_size) with tile_size defined as the tile size used. The block size was defined as (tile_size,block_size). Both tile_size and block_size were varied in the performance tests. These were tested using data sizes of 512, 1024, 2048, and 4096. The run-times and throughputs were measured and the speed-ups and throughput ratios, with the serial as the reference, were calculated. The results are tabulated below and the data was plotted for easier comparison. The code for the kernels can be found in the appendix.

Table: Results



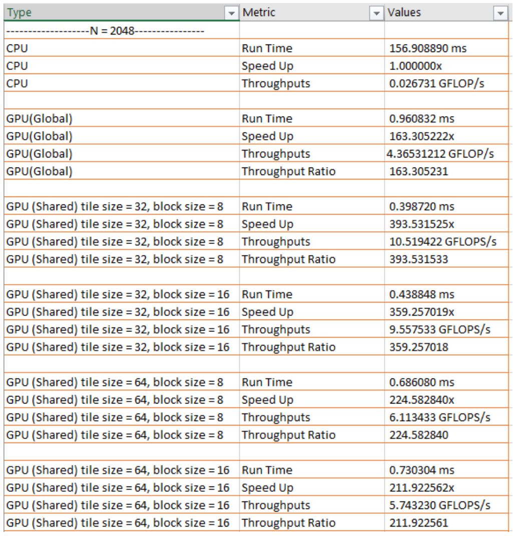
Type	Metric	Values
Threads = 256		
-------------------N = 512----------------		
CPU	Run Time	6.626048 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.039563
GPU(Global)	Run Time	0.197216 ms
GPU(Global)	Speed Up	33.597923x
GPU(Global)	Throughputs	1.3292346 GFLOPS/s
GPU(Global)	Throughput Ratio	33.597923
GPU (Shared) tile size = 32, block size = 8	Run Time	0.037696 ms
GPU (Shared) tile size = 32, block size = 8	Speed Up	175.775894x
GPU (Shared) tile size = 32, block size = 8	Throughputs	6.954160 GFLOPS/s
GPU (Shared) tile size = 32, block size = 8	Throughput Ratio	175.775895
GPU (Shared) tile size = 32, block size = 16	Run Time	0.039232 ms
GPU (Shared) tile size = 32, block size = 16	Speed Up	180.047302x
GPU (Shared) tile size = 32, block size = 16	Throughputs	6.681892 GFLOPS/s
GPU (Shared) tile size = 32, block size = 16	Throughput Ratio	180.047301
GPU (Shared) tile size = 64, block size = 8	Run Time	0.057824 ms
GPU (Shared) tile size = 64, block size = 8	Speed Up	111.301605x
GPU (Shared) tile size = 64, block size = 8	Throughputs	4.533481 GFLOPS/s
GPU (Shared) tile size = 64, block size = 8	Throughput Ratio	111.301601
GPU (Shared) tile size = 64, block size = 16	Run Time	0.061088 ms
GPU (Shared) tile size = 64, block size = 16	Speed Up	111.553696x
GPU (Shared) tile size = 64, block size = 16	Throughputs	4.291252 GFLOPS/s
GPU (Shared) tile size = 64, block size = 16	Throughput Ratio	111.553693

Results for Data Size: 512



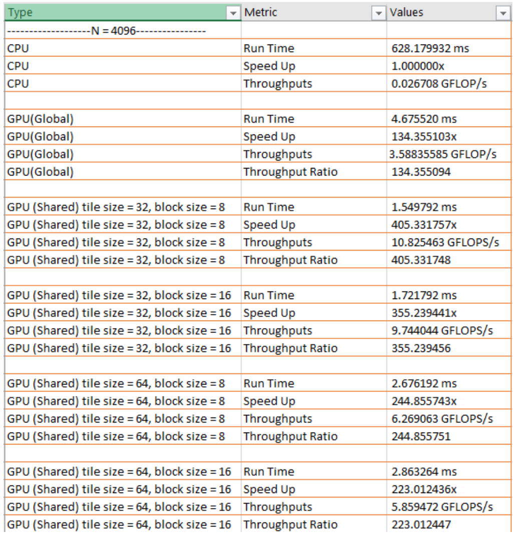
Type	Metric	Values
-------------------N = 1024----------------		
CPU	Run Time	37.513824 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.027952 GFLOP/s
GPU(Global)	Run Time	0.335008 ms
GPU(Global)	Speed Up	111.978889x
GPU(Global)	Throughputs	3.1300339 GFLOP/s
GPU(Global)	Throughput Ratio	111.978890
GPU (Shared) tile size = 32, block size = 8	Run Time	0.110784 ms
GPU (Shared) tile size = 32, block size = 8	Speed Up	338.621307x
GPU (Shared) tile size = 32, block size = 8	Throughputs	9.465049 GFLOPS/s
GPU (Shared) tile size = 32, block size = 8	Throughput Ratio	338.621307
GPU (Shared) tile size = 32, block size = 16	Run Time	0.119680 ms
GPU (Shared) tile size = 32, block size = 16	Speed Up	303.236359x
GPU (Shared) tile size = 32, block size = 16	Throughputs	8.761497 GFLOPS/s
GPU (Shared) tile size = 32, block size = 16	Throughput Ratio	303.236357
GPU (Shared) tile size = 64, block size = 8	Run Time	0.188896 ms
GPU (Shared) tile size = 64, block size = 8	Speed Up	194.022354x
GPU (Shared) tile size = 64, block size = 8	Throughputs	5.551076 GFLOPS/s
GPU (Shared) tile size = 64, block size = 8	Throughput Ratio	194.022353
GPU (Shared) tile size = 64, block size = 16	Run Time	0.196672 ms
GPU (Shared) tile size = 64, block size = 16	Speed Up	190.643341x
GPU (Shared) tile size = 64, block size = 16	Throughputs	5.331598 GFLOPS/s
GPU (Shared) tile size = 64, block size = 16	Throughput Ratio	190.643348

Results for Data Size: 2048



Type	Metric	Values
-------------------N = 2048----------------		
CPU	Run Time	156.908890 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.026731 GFLOP/s
GPU(Global)	Run Time	0.960832 ms
GPU(Global)	Speed Up	163.305222x
GPU(Global)	Throughputs	4.36531212 GFLOP/s
GPU(Global)	Throughput Ratio	163.305231
GPU (Shared) tile size = 32, block size = 8	Run Time	0.398720 ms
GPU (Shared) tile size = 32, block size = 8	Speed Up	393.531525x
GPU (Shared) tile size = 32, block size = 8	Throughputs	10.519422 GFLOPS/s
GPU (Shared) tile size = 32, block size = 8	Throughput Ratio	393.531533
GPU (Shared) tile size = 32, block size = 16	Run Time	0.438848 ms
GPU (Shared) tile size = 32, block size = 16	Speed Up	359.257019x
GPU (Shared) tile size = 32, block size = 16	Throughputs	9.557533 GFLOPS/s
GPU (Shared) tile size = 32, block size = 16	Throughput Ratio	359.257018
GPU (Shared) tile size = 64, block size = 8	Run Time	0.686080 ms
GPU (Shared) tile size = 64, block size = 8	Speed Up	224.582840x
GPU (Shared) tile size = 64, block size = 8	Throughputs	6.113433 GFLOPS/s
GPU (Shared) tile size = 64, block size = 8	Throughput Ratio	224.582840
GPU (Shared) tile size = 64, block size = 16	Run Time	0.730304 ms
GPU (Shared) tile size = 64, block size = 16	Speed Up	211.922562x
GPU (Shared) tile size = 64, block size = 16	Throughputs	5.743230 GFLOPS/s
GPU (Shared) tile size = 64, block size = 16	Throughput Ratio	211.922561

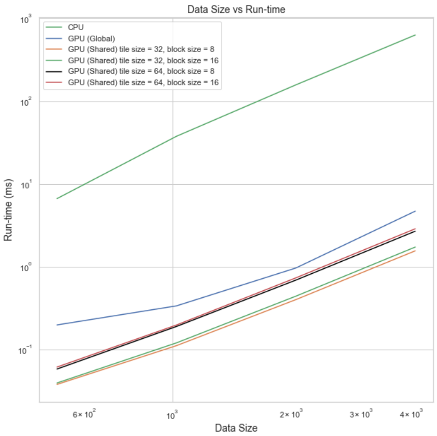
Results for Data Size: 512



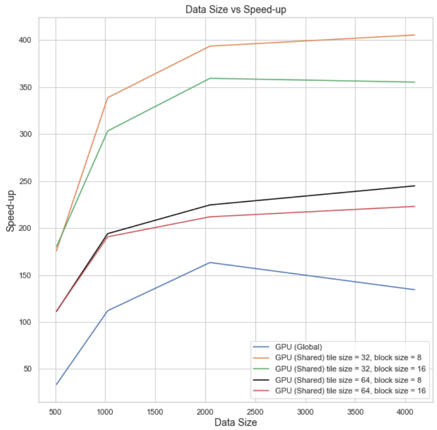
Type	Metric	Values
-------------------N = 4096----------------		
CPU	Run Time	628.179932 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.026708 GFLOP/s
GPU(Global)	Run Time	4.675520 ms
GPU(Global)	Speed Up	134.355103x
GPU(Global)	Throughputs	3.58835585 GFLOP/s
GPU(Global)	Throughput Ratio	134.355094
GPU (Shared) tile size = 32, block size = 8	Run Time	1.549792 ms
GPU (Shared) tile size = 32, block size = 8	Speed Up	405.331757x
GPU (Shared) tile size = 32, block size = 8	Throughputs	10.825463 GFLOPS/s
GPU (Shared) tile size = 32, block size = 8	Throughput Ratio	405.331748
GPU (Shared) tile size = 32, block size = 16	Run Time	1.721792 ms
GPU (Shared) tile size = 32, block size = 16	Speed Up	355.239441x
GPU (Shared) tile size = 32, block size = 16	Throughputs	9.744044 GFLOPS/s
GPU (Shared) tile size = 32, block size = 16	Throughput Ratio	355.239456
GPU (Shared) tile size = 64, block size = 8	Run Time	2.676192 ms
GPU (Shared) tile size = 64, block size = 8	Speed Up	244.855743x
GPU (Shared) tile size = 64, block size = 8	Throughputs	6.269063 GFLOPS/s
GPU (Shared) tile size = 64, block size = 8	Throughput Ratio	244.855751
GPU (Shared) tile size = 64, block size = 16	Run Time	2.863264 ms
GPU (Shared) tile size = 64, block size = 16	Speed Up	223.012436x
GPU (Shared) tile size = 64, block size = 16	Throughputs	5.859472 GFLOPS/s
GPU (Shared) tile size = 64, block size = 16	Throughput Ratio	223.012447

Results for Data Size: 4096

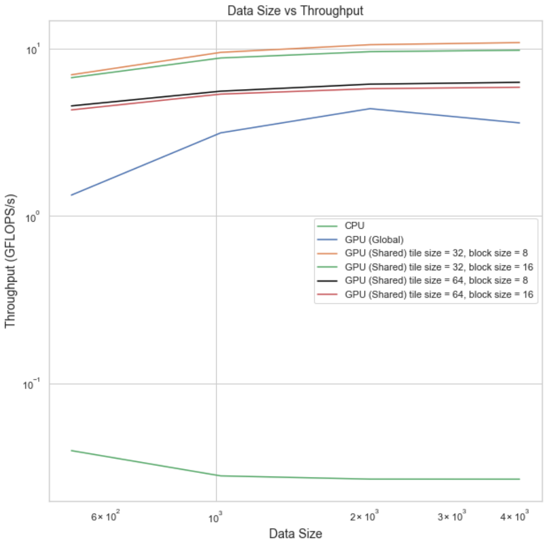
Graphs



Graph Showing Data Size vs Run-time



Graph Showing Data Size vs Speed-up

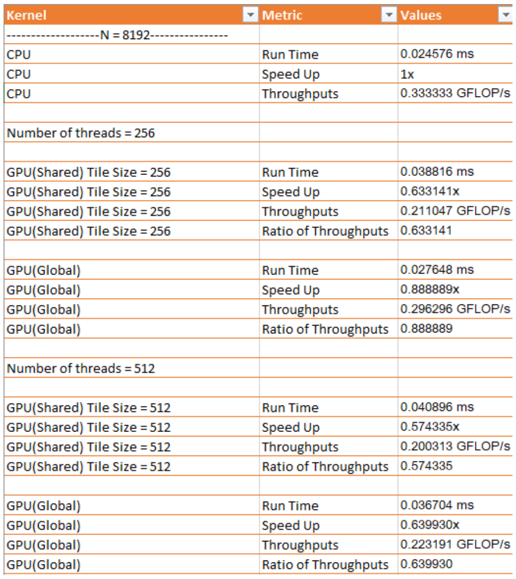


Graph Showing Data Size vs Throughputs

Problem 2: Vector Addition

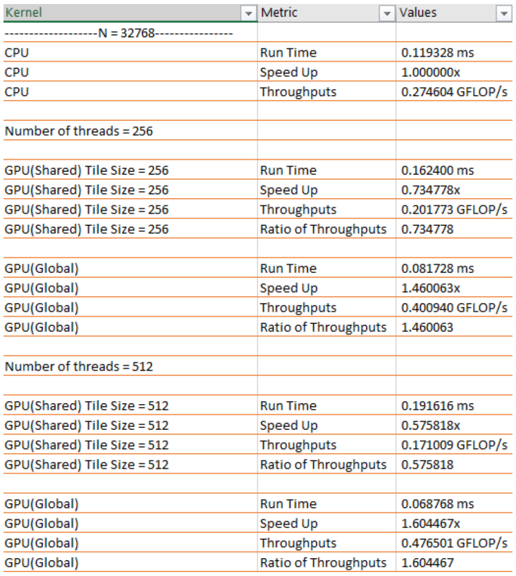
This section compares the serial version of vector addition with the shared memory version, and the global memory version of CUDA parallel programming. In the shared and global kernel, the block size was varied between 256 and 512 threads and used as the value for tile size for the shared memory. The grid size was calculated as N\num_threads, where N is the size of the data and num_threads is the block size. These were tested using data sizes of 8192, 32768, 65536, 1048596, and 4194304. The run-times and throughputs were measured and the speed-ups and throughput ratios, with the serial as the reference, were calculated. The results are tabulated below and the data was plotted for easier comparison. The code for the kernels can be found in the appendix.

Table: Results



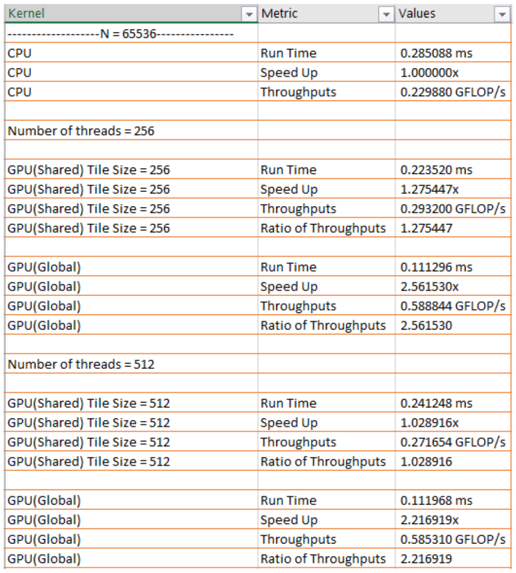
Kernel	Metric	Values
-------------------N = 8192----------------		
CPU	Run Time	0.024576 ms
CPU	Speed Up	1x
CPU	Throughputs	0.333333 GFLOP/s
Number of threads = 256		
GPU(Shared) Tile Size = 256	Run Time	0.038816 ms
GPU(Shared) Tile Size = 256	Speed Up	0.633141x
GPU(Shared) Tile Size = 256	Throughputs	0.211047 GFLOP/s
GPU(Shared) Tile Size = 256	Ratio of Throughputs	0.633141
GPU(Global)	Run Time	0.027648 ms
GPU(Global)	Speed Up	0.888889x
GPU(Global)	Throughputs	0.296296 GFLOP/s
GPU(Global)	Ratio of Throughputs	0.888889
Number of threads = 512		
GPU(Shared) Tile Size = 512	Run Time	0.040896 ms
GPU(Shared) Tile Size = 512	Speed Up	0.574335x
GPU(Shared) Tile Size = 512	Throughputs	0.200313 GFLOP/s
GPU(Shared) Tile Size = 512	Ratio of Throughputs	0.574335
GPU(Global)	Run Time	0.036704 ms
GPU(Global)	Speed Up	0.639930x
GPU(Global)	Throughputs	0.223191 GFLOP/s
GPU(Global)	Ratio of Throughputs	0.639930

Results for Data Size: 8192



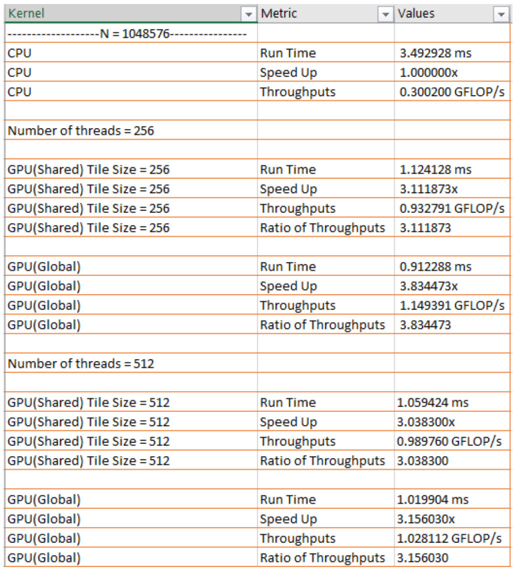
Kernel	Metric	Values
-------------------N = 32768----------------		
CPU	Run Time	0.119328 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.274604 GFLOP/s
Number of threads = 256		
GPU(Shared) Tile Size = 256	Run Time	0.162400 ms
GPU(Shared) Tile Size = 256	Speed Up	0.734778x
GPU(Shared) Tile Size = 256	Throughputs	0.201773 GFLOP/s
GPU(Shared) Tile Size = 256	Ratio of Throughputs	0.734778
GPU(Global)	Run Time	0.081728 ms
GPU(Global)	Speed Up	1.460063x
GPU(Global)	Throughputs	0.400940 GFLOP/s
GPU(Global)	Ratio of Throughputs	1.460063
Number of threads = 512		
GPU(Shared) Tile Size = 512	Run Time	0.191616 ms
GPU(Shared) Tile Size = 512	Speed Up	0.575818x
GPU(Shared) Tile Size = 512	Throughputs	0.171009 GFLOP/s
GPU(Shared) Tile Size = 512	Ratio of Throughputs	0.575818
GPU(Global)	Run Time	0.068768 ms
GPU(Global)	Speed Up	1.604467x
GPU(Global)	Throughputs	0.476501 GFLOP/s
GPU(Global)	Ratio of Throughputs	1.604467

Results for Data Size: 32768



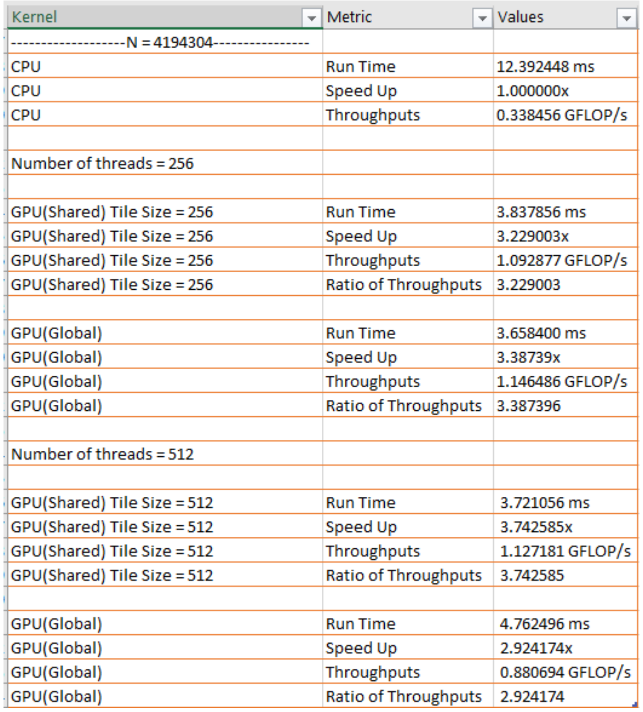
Kernel	Metric	Values
------------------N = 65536---------------		
CPU	Run Time	0.285088 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.229880 GFLOP/s
Number of threads = 256		
GPU(Shared) Tile Size = 256	Run Time	0.223520 ms
GPU(Shared) Tile Size = 256	Speed Up	1.275447x
GPU(Shared) Tile Size = 256	Throughputs	0.293200 GFLOP/s
GPU(Shared) Tile Size = 256	Ratio of Throughputs	1.275447
GPU(Global)	Run Time	0.111296 ms
GPU(Global)	Speed Up	2.561530x
GPU(Global)	Throughputs	0.588844 GFLOP/s
GPU(Global)	Ratio of Throughputs	2.561530
Number of threads = 512		
GPU(Shared) Tile Size = 512	Run Time	0.241248 ms
GPU(Shared) Tile Size = 512	Speed Up	1.028916x
GPU(Shared) Tile Size = 512	Throughputs	0.271654 GFLOP/s
GPU(Shared) Tile Size = 512	Ratio of Throughputs	1.028916
GPU(Global)	Run Time	0.111968 ms
GPU(Global)	Speed Up	2.216919x
GPU(Global)	Throughputs	0.585310 GFLOP/s
GPU(Global)	Ratio of Throughputs	2.216919

Results for Data Size: 65536



Kernel	Metric	Values
------------------N = 1048576---------------		
CPU	Run Time	3.492928 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.300200 GFLOP/s
Number of threads = 256		
GPU(Shared) Tile Size = 256	Run Time	1.124128 ms
GPU(Shared) Tile Size = 256	Speed Up	3.111873x
GPU(Shared) Tile Size = 256	Throughputs	0.932791 GFLOP/s
GPU(Shared) Tile Size = 256	Ratio of Throughputs	3.111873
GPU(Global)	Run Time	0.912288 ms
GPU(Global)	Speed Up	3.834473x
GPU(Global)	Throughputs	1.149391 GFLOP/s
GPU(Global)	Ratio of Throughputs	3.834473
Number of threads = 512		
GPU(Shared) Tile Size = 512	Run Time	1.059424 ms
GPU(Shared) Tile Size = 512	Speed Up	3.038300x
GPU(Shared) Tile Size = 512	Throughputs	0.989760 GFLOP/s
GPU(Shared) Tile Size = 512	Ratio of Throughputs	3.038300
GPU(Global)	Run Time	1.019904 ms
GPU(Global)	Speed Up	3.156030x
GPU(Global)	Throughputs	1.028112 GFLOP/s
GPU(Global)	Ratio of Throughputs	3.156030

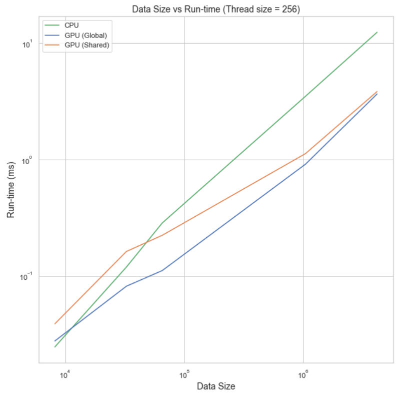
Results for Data Size: 1048576



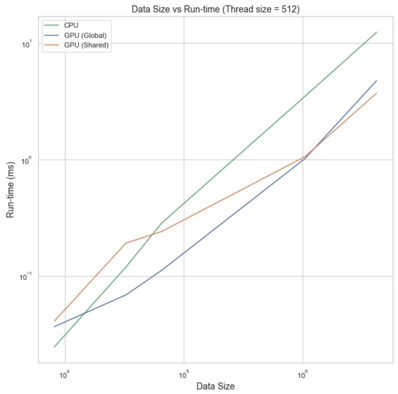
Kernel	Metric	Values
------------------N = 4194304---------------		
CPU	Run Time	12.392448 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.338456 GFLOP/s
Number of threads = 256		
GPU(Shared) Tile Size = 256	Run Time	3.837856 ms
GPU(Shared) Tile Size = 256	Speed Up	3.229003x
GPU(Shared) Tile Size = 256	Throughputs	1.092877 GFLOP/s
GPU(Shared) Tile Size = 256	Ratio of Throughputs	3.229003
GPU(Global)	Run Time	3.658400 ms
GPU(Global)	Speed Up	3.38739x
GPU(Global)	Throughputs	1.146486 GFLOP/s
GPU(Global)	Ratio of Throughputs	3.387396
Number of threads = 512		
GPU(Shared) Tile Size = 512	Run Time	3.721056 ms
GPU(Shared) Tile Size = 512	Speed Up	3.742585x
GPU(Shared) Tile Size = 512	Throughputs	1.127181 GFLOP/s
GPU(Shared) Tile Size = 512	Ratio of Throughputs	3.742585
GPU(Global)	Run Time	4.762496 ms
GPU(Global)	Speed Up	2.924174x
GPU(Global)	Throughputs	0.880694 GFLOP/s
GPU(Global)	Ratio of Throughputs	2.924174

Results for Data Size: 4194304

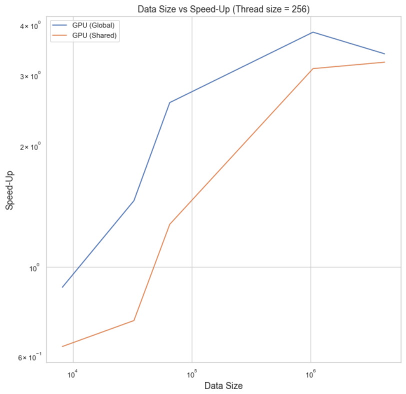
Graphs



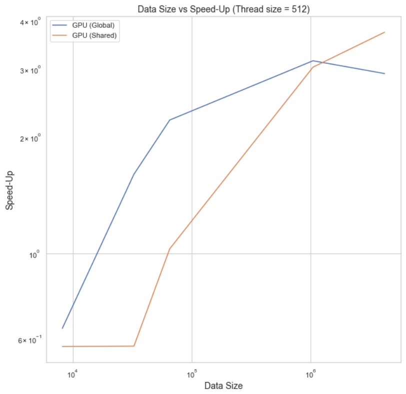
Graph Showing Data Size vs Run-time (256 Threads)



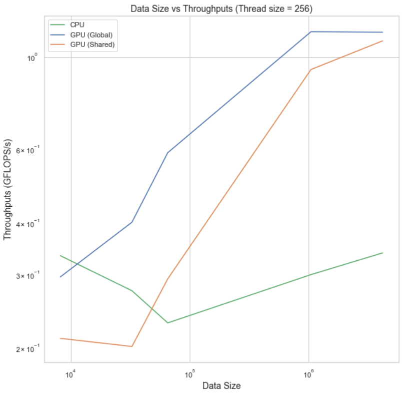
Graph Showing Data Size vs Run-time (512 Threads)



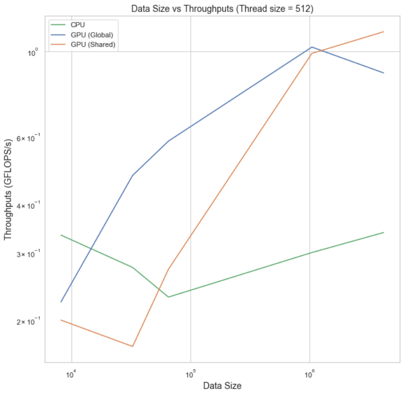
Graph Showing Data Size vs Speed-up (256 Threads)



Graph Showing Data Size vs Speed-up (512 Threads)



Graph Showing Data Size vs Throughputs (256 Threads)

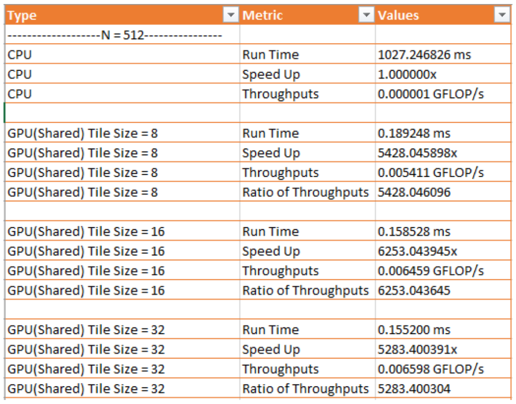


Graph Showing Data Size vs Throughputs (512 Threads)

Problem 3: Matrix Multiplication

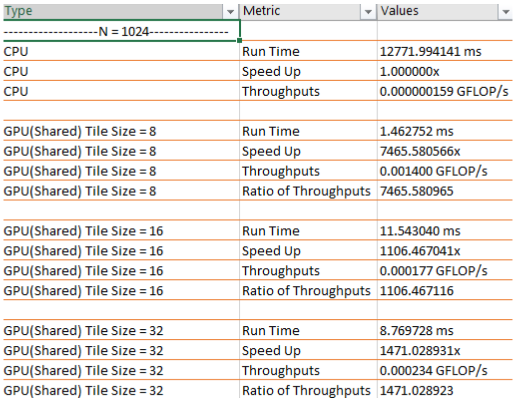
This section compares the serial version of matrix multiplication with the shared memory implementation of CUDA parallel programming. For the shared memory kernel a two dimensional grid size and two dimensional block size were used, where the grid size was defined as (N\tile_size,N\tile_size) with tile_size defined as the tile size used and N as the data size. The block size was defined as (tile_size,block_size). Tile_size was varied in the performance tests with the values of 8, 16, and 32. These were tested using data sizes of 512, 1024, 2048, and 4096. The run-times and throughputs were measured and the speed-ups and throughput ratios, with the serial as the reference, were calculated. The results are tabulated below and the data was plotted for easier comparison. The code for the kernels can be found in the appendix.

Table: Results



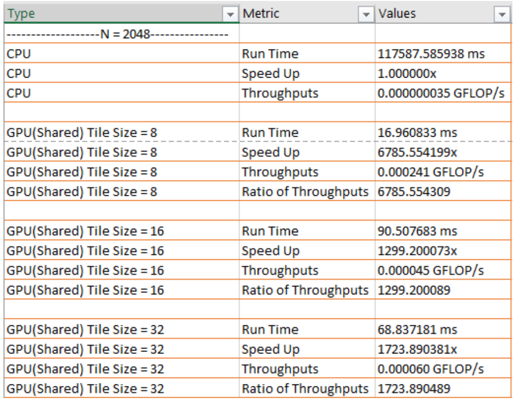
Type	Metric	Values
-------------------N = 512----------------		
CPU	Run Time	1027.246826 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.000001 GFLOP/s
GPU(Shared) Tile Size = 8	Run Time	0.189248 ms
GPU(Shared) Tile Size = 8	Speed Up	5428.045898x
GPU(Shared) Tile Size = 8	Throughputs	0.005411 GFLOP/s
GPU(Shared) Tile Size = 8	Ratio of Throughputs	5428.046096
GPU(Shared) Tile Size = 16	Run Time	0.158528 ms
GPU(Shared) Tile Size = 16	Speed Up	6253.043945x
GPU(Shared) Tile Size = 16	Throughputs	0.006459 GFLOP/s
GPU(Shared) Tile Size = 16	Ratio of Throughputs	6253.043645
GPU(Shared) Tile Size = 32	Run Time	0.155200 ms
GPU(Shared) Tile Size = 32	Speed Up	5283.400391x
GPU(Shared) Tile Size = 32	Throughputs	0.006598 GFLOP/s
GPU(Shared) Tile Size = 32	Ratio of Throughputs	5283.400304

Results for Data Size: 512



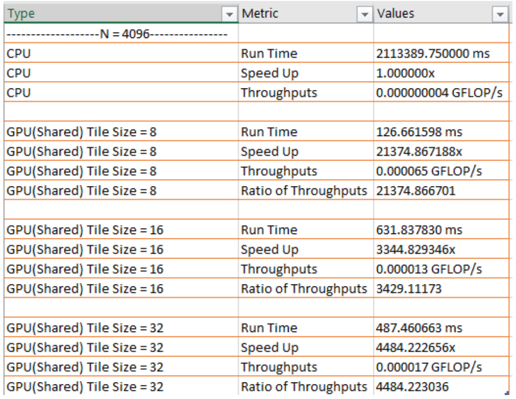
Type	Metric	Values
-------------------N = 1024----------------		
CPU	Run Time	12771.994141 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.000000159 GFLOP/s
GPU(Shared) Tile Size = 8	Run Time	1.462752 ms
GPU(Shared) Tile Size = 8	Speed Up	7465.580566x
GPU(Shared) Tile Size = 8	Throughputs	0.001400 GFLOP/s
GPU(Shared) Tile Size = 8	Ratio of Throughputs	7465.580965
GPU(Shared) Tile Size = 16	Run Time	11.543040 ms
GPU(Shared) Tile Size = 16	Speed Up	1106.467041x
GPU(Shared) Tile Size = 16	Throughputs	0.000177 GFLOP/s
GPU(Shared) Tile Size = 16	Ratio of Throughputs	1106.467116
GPU(Shared) Tile Size = 32	Run Time	8.769728 ms
GPU(Shared) Tile Size = 32	Speed Up	1471.028931x
GPU(Shared) Tile Size = 32	Throughputs	0.000234 GFLOP/s
GPU(Shared) Tile Size = 32	Ratio of Throughputs	1471.028923

Results for Data Size: 2048



Type	Metric	Values
-------------------N = 2048----------------		
CPU	Run Time	117587.585938 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.000000035 GFLOP/s
GPU(Shared) Tile Size = 8	Run Time	16.960833 ms
GPU(Shared) Tile Size = 8	Speed Up	6785.554199x
GPU(Shared) Tile Size = 8	Throughputs	0.000241 GFLOP/s
GPU(Shared) Tile Size = 8	Ratio of Throughputs	6785.554309
GPU(Shared) Tile Size = 16	Run Time	90.507683 ms
GPU(Shared) Tile Size = 16	Speed Up	1299.200073x
GPU(Shared) Tile Size = 16	Throughputs	0.000045 GFLOP/s
GPU(Shared) Tile Size = 16	Ratio of Throughputs	1299.200089
GPU(Shared) Tile Size = 32	Run Time	68.837181 ms
GPU(Shared) Tile Size = 32	Speed Up	1723.890381x
GPU(Shared) Tile Size = 32	Throughputs	0.000060 GFLOP/s
GPU(Shared) Tile Size = 32	Ratio of Throughputs	1723.890489

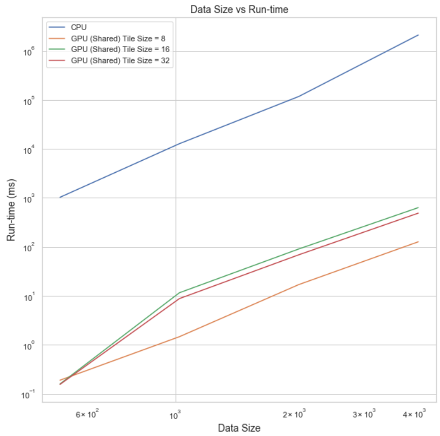
Results for Data Size: 512



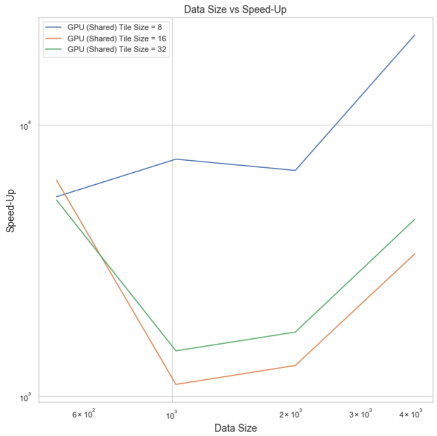
Type	Metric	Values
-------------------N = 4096----------------		
CPU	Run Time	2113389.750000 ms
CPU	Speed Up	1.000000x
CPU	Throughputs	0.000000004 GFLOP/s
GPU(Shared) Tile Size = 8	Run Time	126.661598 ms
GPU(Shared) Tile Size = 8	Speed Up	21374.867188x
GPU(Shared) Tile Size = 8	Throughputs	0.000065 GFLOP/s
GPU(Shared) Tile Size = 8	Ratio of Throughputs	21374.866701
GPU(Shared) Tile Size = 16	Run Time	631.837830 ms
GPU(Shared) Tile Size = 16	Speed Up	3344.829346x
GPU(Shared) Tile Size = 16	Throughputs	0.000013 GFLOP/s
GPU(Shared) Tile Size = 16	Ratio of Throughputs	3429.11173
GPU(Shared) Tile Size = 32	Run Time	487.460663 ms
GPU(Shared) Tile Size = 32	Speed Up	4484.222656x
GPU(Shared) Tile Size = 32	Throughputs	0.000017 GFLOP/s
GPU(Shared) Tile Size = 32	Ratio of Throughputs	4484.223036

Results for Data Size: 4096

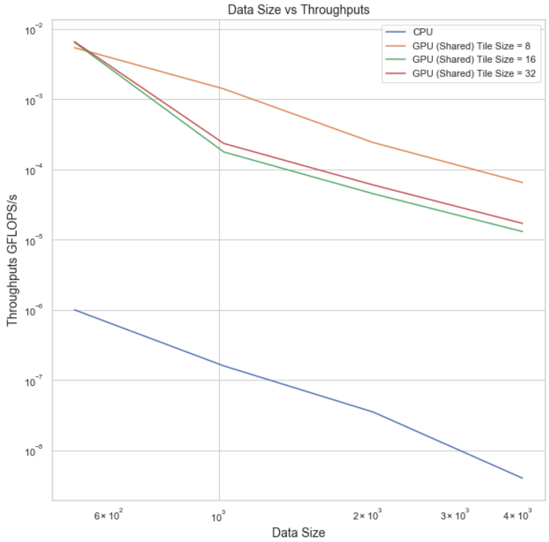
Graphs



Graph Showing Data Size vs Run-time



Graph Showing Data Size vs Speed-up



Graph Showing Data Size vs Throughputs

Appendix

Kernels for Transposition

```
__global__ void glob_gpuTranspose(float* arr, float* out)
{
    int row, col, t;
    unsigned int i = threadIdx.x + blockIdx.x*blockDim.x;
    if (i < (N*N))
    {
        row = i/N;
        col = i % N;
        t = col*N + row;
        out[t] = arr[i];
    }
}
```

Listing 1: Global Memory Kernel

```
__global__ void share_gpuTranspose(float* arr, float* out)
{
    __shared__ float tile[tile_size][tile_size];

    unsigned int row = threadIdx.x + blockIdx.x*tile_size;
    unsigned int col = threadIdx.y + blockIdx.y*tile_size;

    for (int i = 0; i < tile_size; i += block_size)
    {
        if ((row < N*N) && (col < N*N))
        tile[threadIdx.y+i][threadIdx.x] = arr[(col+i)*N + row];
    }
    __syncthreads();

    row = blockIdx.y*tile_size + threadIdx.x;
    col = blockIdx.x*tile_size + threadIdx.y;


    for (int i = 0; i < tile_size; i += block_size)
    {
        if ((row < N*N) && (col < N*N))
        out[(col +i)*N + row] = tile[threadIdx.x][threadIdx.y + i];
    }
}
```

Listing 2: Shared Memory Kernel

Kernels for Vector Addition

```
__global__ void share_VecAdd(float* arr_in, float* arr_out)
{
    __shared__ float sVec[num_threads];

    unsigned int i = threadIdx.x + blockIdx.x*blockDim.x;

    //Loading shared memory
    if (i < N)
    {
    sVec[threadIdx.x] = arr_in[i];
    }
    __syncthreads();

    //reduction in shared mem
    for(unsigned int j = blockDim.x/2; j > 0; j >>= 1)
    {
        if (threadIdx.x < j)
        {
            sVec[threadIdx.x] += sVec[threadIdx.x + j];
        }
        __syncthreads();
    }
    if (threadIdx.x == 0)
    {
        atomicAdd(arr_out,sVec[threadIdx.x]);
    }

}
```

Listing 3: Shared Memory Kernel

```
__global__ void glob_VecAdd(float* arr_in, float* arr_out)
{

    unsigned int i = threadIdx.x + blockIdx.x*blockDim.x;

    for(unsigned int j = blockDim.x/2; j > 0; j = j/2)
    {
        if (threadIdx.x < j)
        {
            arr_in[i] += arr_in[i + j];
        }
        __syncthreads();
    }
    if (threadIdx.x == 0)
    {
        atomicAdd(arr_out,arr_in[i]);
    }

}
```

Listing 4: Global Memory Kernel

Kernels for Matrix Multiplication

```
__global__ void matrixMul(float* a, float* b, float* c)
{
    __shared__ float a_s[tile_size][tile_size];
    __shared__ float b_s[tile_size][tile_size];

    int row = threadIdx.y + blockIdx.y*tile_size;
    int col = threadIdx.x + blockIdx.x*tile_size;

    float sum_value = 0;

        for (int i = 0; i < N/tile_size; i++)
        {
            a_s[threadIdx.y][threadIdx.x] = a[row*N + (i*tile_size + threadIdx.x)];
            b_s[threadIdx.y][threadIdx.x] = b[col + (i*tile_size + threadIdx.y)*N];
            __syncthreads();

            for (int j = 0; j < tile_size; j++)
            {
                sum_value += a_s[threadIdx.y][j]*b_s[j][threadIdx.x];
            }
         __syncthreads();
        }
    c[row*N+col] = sum_value;
}

void checkCUDAError(const char *msg)
{
    cudaError_t err = cudaGetLastError();
    if( cudaSuccess != err) {
        fprintf(stderr, "Cuda error: %s: %s.\n", msg, cudaGetErrorString( err) );
        exit(EXIT_FAILURE);
    }
}
```

Listing 5: Shared Memory Kernel