Matrix transpose optimization with Cuda

- ## Algorithm

  storing index (i,j) of input matrix in the index(j,i) of output matrix.

- ## Implementations

  - ### Serial transpose with CPU

    The input matrix with size m*n stores as one-dimension list that contains all its elements in sequence.
    For calculating the transpose, we change elements places in the follow manner:

    ```
    for(int n = 0; n<dim1*dim2; n++) {
            int i = n/dim1;
            int j = n%dim1;
            out[n] = in_[dim2*j + i];
        }
    ```

    For storing index `(j,i)` of output matrix with index `(i,j)` of input matrix we first find `(i,j)` and then calculate the destination index by `(dim2*j + i)` which means index `(j,i)` in output matrix.

  - ### Parallel, global memory transpose

    Every GPU thread transpose one element of matrix.
    Row and column of element for every thread calculate as follow:

    ```
    int column = tile_size * blockIdx.x + threadIdx.x;
    int row = tile_size * blockIdx.y + threadIdx.y;
    out[column*dim2 + row] = in_[column + row*dim2];
    ```

  - ### Parallel, shared memory transpose

    Shared memory is a memory that is shared among block threads and help performance.
    We calculate row and column as previous manner but first we transfer input matrix element into shared memory and then place it from shared memory into output matrix.

    ```
    M_Shared[threadIdx.y][threadIdx.x] = in_[index_in];
    __syncthreads();
    ```

```
out[index_out] = M_Shared[threadIdx.y][threadIdx.x];
```

- commands:
  installing dependencies as follow:

```
!pip install git+git://github.com/andreinechaev/nvcc4jupyter.git

%load_ext nvcc_plugin
```

for creating .cu file we place `%%writefile transpose.cu`
before including libraries and then running the Google Colab cell.

Then compiling .cu file by:

```
!nvcc transpose.cu -o out_int_10 -Wno-deprecated-gpu-targets
```

At last we profile output using nvprof as bellow:

```
!nvprof ./out_dataType_TileWidth  dim1 dim2
```

Like:

```
!nvprof ./out_int_10  100 100
```

- Results
  - Checking result correctness
    As the log shows the result calculated correctly in serial and parallel way:

```
1 !./out_double_5 10 10

Matrix data type : double
 dimentions = (10 , 10) ,Tile width = 5
*********matrix**********
 0.00  10.00  20.00  30.00  40.00  50.00  60.00  70.00  80.00  90.00
 1.00  11.00  21.00  31.00  41.00  51.00  61.00  71.00  81.00  91.00
 2.00  12.00  22.00  32.00  42.00  52.00  62.00  72.00  82.00  92.00
 3.00  13.00  23.00  33.00  43.00  53.00  63.00  73.00  83.00  93.00
 4.00  14.00  24.00  34.00  44.00  54.00  64.00  74.00  84.00  94.00
 5.00  15.00  25.00  35.00  45.00  55.00  65.00  75.00  85.00  95.00
 6.00  16.00  26.00  36.00  46.00  56.00  66.00  76.00  86.00  96.00
 7.00  17.00  27.00  37.00  47.00  57.00  67.00  77.00  87.00  97.00
 8.00  18.00  28.00  38.00  48.00  58.00  68.00  78.00  88.00  98.00
 9.00  19.00  29.00  39.00  49.00  59.00  69.00  79.00  89.00  99.00

*********serial**********
 0.00  1.00  2.00  3.00  4.00  5.00  6.00  7.00  8.00  9.00
 10.00  11.00  12.00  13.00  14.00  15.00  16.00  17.00  18.00  19.00
 20.00  21.00  22.00  23.00  24.00  25.00  26.00  27.00  28.00  29.00
 30.00  31.00  32.00  33.00  34.00  35.00  36.00  37.00  38.00  39.00
 40.00  41.00  42.00  43.00  44.00  45.00  46.00  47.00  48.00  49.00
 50.00  51.00  52.00  53.00  54.00  55.00  56.00  57.00  58.00  59.00
 60.00  61.00  62.00  63.00  64.00  65.00  66.00  67.00  68.00  69.00
 70.00  71.00  72.00  73.00  74.00  75.00  76.00  77.00  78.00  79.00
 80.00  81.00  82.00  83.00  84.00  85.00  86.00  87.00  88.00  89.00
 90.00  91.00  92.00  93.00  94.00  95.00  96.00  97.00  98.00  99.00

*********parallel**********
 0.00  1.00  2.00  3.00  4.00  5.00  6.00  7.00  8.00  9.00
 10.00  11.00  12.00  13.00  14.00  15.00  16.00  17.00  18.00  19.00
 20.00  21.00  22.00  23.00  24.00  25.00  26.00  27.00  28.00  29.00
 30.00  31.00  32.00  33.00  34.00  35.00  36.00  37.00  38.00  39.00
 40.00  41.00  42.00  43.00  44.00  45.00  46.00  47.00  48.00  49.00
 50.00  51.00  52.00  53.00  54.00  55.00  56.00  57.00  58.00  59.00
 60.00  61.00  62.00  63.00  64.00  65.00  66.00  67.00  68.00  69.00
 70.00  71.00  72.00  73.00  74.00  75.00  76.00  77.00  78.00  79.00
 80.00  81.00  82.00  83.00  84.00  85.00  86.00  87.00  88.00  89.00
 90.00  91.00  92.00  93.00  94.00  95.00  96.00  97.00  98.00  99.00

Time for the serial:     3 ms
Time for the NAIVE:      0.142080 ms and speedup: 21.114866
Time for the shared:     0.021088 ms and speedup: 142.261002
```

- Algorithm analysis

  Initialing the matrix is the same for serial and parallel manner but the way that algorithm process the input matrix is different and as image shows the serial way is significantly slower.

Code analysis

As the profiling shows 77% of execution time is for copy data from device to host and 22.70% for copy data from host to device.

```
1 !nvprof ./out_int_50 1000 1000
```

```
Matrix data type : integer
==2684== NVPROF is profiling process 2684, command: ./out_int_50 1000 1000
 dimentions = (1000 , 1000) ,Tile width = 50
Time for the serial:     11169 ms
Time for the NAIVE:      0.003872 ms and speedup: 2884555.750000
Time for the shared:     0.002752 ms and speedup: 4058503.000000

==2684== Profiling application: ./out_int_50 1000 1000
==2684== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   77.22%  1.8921ms         1  1.8921ms  1.8921ms  1.8921ms  [CUDA memcpy DtoH]
                   22.70%  556.32us         1  556.32us  556.32us  556.32us  [CUDA memcpy HtoD]
                    0.07%  1.8240us         1  1.8240us  1.8240us  1.8240us  [CUDA memset]
```

So for improving the performance we use `cudaMemcpyAsync()` instead of `cudaMemcpy()` And performance improve by factor 1.272 (1.8921 ms/1.4875ms) for copy from device to host and 1.051 (556.32 ms/528.83ms) improvement for copy data from host to device.

```
1 !nvprof ./out_int_50 1000 1000
```

```
Matrix data type : integer
==2750== NVPROF is profiling process 2750, command: ./out_int_50 1000 1000
 dimentions = (1000 , 1000) ,Tile width = 50
Time for the serial:     13927 ms
Time for the NAIVE:      0.005376 ms and speedup: 2590587.750000
Time for the shared:     0.002752 ms and speedup: 5060683.000000

==2750== Profiling application: ./out_int_50 1000 1000
==2750== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   73.70%  1.4875ms         1  1.4875ms  1.4875ms  1.4875ms  [CUDA memcpy DtoH]
                   26.20%  528.83us         1  528.83us  528.83us  528.83us  [CUDA memcpy HtoD]
                    0.09%  1.8560us         1  1.8560us  1.8560us  1.8560us  [CUDA memset]
```

- # Profiling the parallel execution

  - ## Integer type:

    - ### 10 * 10

```
!nvprof ./out_int_5 10 10
```

```
Matrix data type : integer
==2989== NVPROF is profiling process 2989, command: ./out_int_5 10 10
 dimentions = (10 , 10) ,Tile width = 5
Time for the serial:     3 ms
Time for the NAIVE:      0.171456 ms and speedup: 17.497202
Time for the shared:     0.019328 ms and speedup: 155.215225

==2989== Profiling application: ./out_int_5 10 10
==2989== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   23.53%  3.5840us         1  3.5840us  3.5840us  3.5840us  transpose_GPU(int*, int*, int, int)
                   22.69%  3.4560us         1  3.4560us  3.4560us  3.4560us  [CUDA memset]
                   19.54%  2.9760us         1  2.9760us  2.9760us  2.9760us  transpose_GPU_shared(int*, int*, int, int)
                   19.12%  2.9120us         1  2.9120us  2.9120us  2.9120us  [CUDA memcpy DtoH]
                   15.13%  2.3040us         1  2.3040us  2.3040us  2.3040us  [CUDA memcpy HtoD]
```

    - ### 100 * 100

```
!nvprof ./out_int_10 100 100
```

```
Matrix data type : integer
==3100== NVPROF is profiling process 3100, command: ./out_int_10 100 100
 dimentions = (100 , 100) ,Tile width = 10
Time for the serial:     110 ms
Time for the NAIVE:      0.138592 ms and speedup: 793.696594
Time for the shared:     0.020448 ms and speedup: 5379.499512

==3100== Profiling application: ./out_int_10 100 100
==3100== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   30.01%  10.592us         1  10.592us  10.592us  10.592us  [CUDA memcpy HtoD]
                   23.84%  8.4160us         1  8.4160us  8.4160us  8.4160us  [CUDA memset]
                   21.85%  7.7120us         1  7.7120us  7.7120us  7.7120us  [CUDA memcpy DtoH]
                   12.69%  4.4800us         1  4.4800us  4.4800us  4.4800us  transpose_GPU_shared(int*, int*, int, int)
                   11.60%  4.0960us         1  4.0960us  4.0960us  4.0960us  transpose_GPU(int*, int*, int, int)
```

    - ### 1000 * 1000

```
!nvprof ./out_int_50 1000 1000
```

```
Matrix data type : integer
==3193== NVPROF is profiling process 3193, command: ./out_int_50 1000 1000
 dimentions = (1000 , 1000) ,Tile width = 50
Time for the serial:     10610 ms
Time for the NAIVE:      0.005728 ms and speedup: 1852304.500000
Time for the shared:     0.002784 ms and speedup: 3811063.250000

==3193== Profiling application: ./out_int_50 1000 1000
==3193== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   73.38%  1.6129ms         1  1.6129ms  1.6129ms  1.6129ms  [CUDA memcpy DtoH]
                   26.54%  583.26us         1  583.26us  583.26us  583.26us  [CUDA memcpy HtoD]
                    0.09%  1.8880us         1  1.8880us  1.8880us  1.8880us  [CUDA memset]
```

- Double type:
  - 10 * 10

```
1 !nvprof ./out_double_5 10 10
```

```
Matrix data type : double
==3426== NVPROF is profiling process 3426, command: ./out_double_5 10 10
 dimentions = (10 , 10) ,Tile width = 5
Time for the serial:     3 ms
Time for the NAIVE:      0.101440 ms and speedup: 29.574133
Time for the shared:     0.018880 ms and speedup: 158.898300

==3426== Profiling application: ./out_double_5 10 10
==3426== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   23.50%  3.5200us         1  3.5200us  3.5200us  3.5200us  [CUDA memset]
                   23.08%  3.4560us         1  3.4560us  3.4560us  3.4560us  transpose_GPU_shared(double*, double*, int, int)
                   20.73%  3.1040us         1  3.1040us  3.1040us  3.1040us  transpose_GPU(double*, double*, int, int)
                   16.88%  2.5280us         1  2.5280us  2.5280us  2.5280us  [CUDA memcpy DtoH]
                   15.81%  2.3680us         1  2.3680us  2.3680us  2.3680us  [CUDA memcpy HtoD]
```

  - 100 * 100

```
1 !nvprof ./out_double_10 100 100
```

```
Matrix data type : double
==3621== NVPROF is profiling process 3621, command: ./out_double_10 100 100
 dimentions = (100 , 100) ,Tile width = 10
Time for the serial:     117 ms
Time for the NAIVE:      0.088064 ms and speedup: 1328.579224
Time for the shared:     0.018912 ms and speedup: 6186.547852

==3621== Profiling application: ./out_double_10 100 100
==3621== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   39.88%  15.008us         1  15.008us  15.008us  15.008us  [CUDA memcpy HtoD]
                   33.76%  12.704us         1  12.704us  12.704us  12.704us  [CUDA memcpy DtoH]
                   11.82%  4.4480us         1  4.4480us  4.4480us  4.4480us  transpose_GPU_shared(double*, double*, int, int)
                   10.88%  4.0960us         1  4.0960us  4.0960us  4.0960us  transpose_GPU(double*, double*, int, int)
                    3.66%  1.3760us         1  1.3760us  1.3760us  1.3760us  [CUDA memset]
```

  - 1000 * 1000

```
1 !nvprof ./out_double_50 1000 1000
```

```
Matrix data type : double
==3671== NVPROF is profiling process 3671, command: ./out_double_50 1000 1000
 dimentions = (1000 , 1000) ,Tile width = 50
Time for the serial:     14022 ms
Time for the NAIVE:      0.006336 ms and speedup: 2213068.250000
Time for the shared:     0.003456 ms and speedup: 4057291.750000

==3671== Profiling application: ./out_double_50 1000 1000
==3671== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   81.89%  5.0466ms         2  2.5233ms  1.0460ms  4.0007ms  [CUDA memcpy DtoH]
                   18.09%  1.1147ms         1  1.1147ms  1.1147ms  1.1147ms  [CUDA memcpy HtoD]
                    0.03%  1.6960us         1  1.6960us  1.6960us  1.6960us  [CUDA memset]
```

As result shows by increasing the matrix dimension data transferring overhead overcome the data process overhead and this shows that data transfer is still the bottle neck of the process.

GPU Shared memory size is another limitation of the process as bellow shows its dimensions for storing double can’t be 100*100:

```
1 !nvcc transpose_double_100.cu -o out_double_100 -Wno-deprecated-gpu-targets

ptxas error   : Entry function '_Z20transpose_GPU_sharedPdS_ii' uses too much shared data
```

- System configuration

GPU:

Model:Tesla K80
Architecture: Kepler 2.0
Base clock: 562 MHz
Memory clock: 1253 MHz ,5012 MHz effective

RAM:

Memory Type: GDDR5
Memory Size :12 GB

Cache:

L1 cache:16 KB (per SMX)
L2 cache: 1536 KB