Julia set

1D indexing

1d_1.cu : <<< DIM * DIM, 1 >>>

```
Time to generate using CPU:  927.4 ms
Time to generate using GPU:  84.8 ms
==4472== Profiling application: ./julia
==4472== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   98.59%  21.877ms         1  21.877ms  21.877ms  21.877ms  kernel(unsigned char*)
                    0.74%  165.09us         1  165.09us  165.09us  165.09us  [CUDA memcpy HtoD]
                    0.66%  146.78us         1  146.78us  146.78us  146.78us  [CUDA memcpy DtoH]
      API calls:   63.38%  158.06ms         4  39.515ms     700ns  158.05ms  cudaEventCreate
                   26.44%  65.928ms         1  65.928ms  65.928ms  65.928ms  cudaDeviceReset
                    9.27%  23.118ms         2  11.559ms  285.70us  22.833ms  cudaMemcpy
                    0.62%  1.5509ms        96  16.155us     100ns  1.4293ms  cuDeviceGetAttribute
                    0.10%  241.50us         1  241.50us  241.50us  241.50us  cudaMalloc
                    0.07%  179.90us         1  179.90us  179.90us  179.90us  cudaFree
                    0.06%  158.50us         1  158.50us  158.50us  158.50us  cuDeviceTotalMem
                    0.02%  43.799us         4  10.949us  3.0000us  25.100us  cudaEventRecord
                    0.01%  36.699us         1  36.699us  36.699us  36.699us  cudaLaunchKernel
                    0.01%  24.200us         2  12.100us  10.500us  13.700us  cudaEventSynchronize
                    0.01%  17.900us         1  17.900us  17.900us  17.900us  cuDeviceGetName
                    0.00%  5.4000us         2  2.7000us  2.6000us  2.8000us  cudaEventElapsedTime
                    0.00%  2.7000us         1  2.7000us  2.7000us  2.7000us  cuDeviceGetPCIBusId
                    0.00%  1.8000us         3     600ns     100ns  1.1000us  cuDeviceGetCount
                    0.00%  1.4000us         2     700ns     300ns  1.1000us  cuDeviceGet
```

Number of warps = 1000*1000 = 1000000

1d_2.cu : <<< 977, 1024 >>>

```
Time to generate using CPU:  947.9 ms
Time to generate using GPU:  64.9 ms
==5072== Profiling application: ./julia
==5072== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   72.21%  807.90us         1  807.90us  807.90us  807.90us  kernel(unsigned char*)
                   14.75%  164.99us         1  164.99us  164.99us  164.99us  [CUDA memcpy HtoD]
                   13.05%  145.98us         1  145.98us  145.98us  145.98us  [CUDA memcpy DtoH]
      API calls:   62.61%  116.48ms         4  29.121ms     800ns  116.48ms  cudaEventCreate
                   35.05%  65.211ms         1  65.211ms  65.211ms  65.211ms  cudaDeviceReset
                    1.09%  2.0299ms         2  1.0149ms  287.70us  1.7422ms  cudaMemcpy
                    0.84%  1.5623ms        96  16.273us     100ns  1.4291ms  cuDeviceGetAttribute
                    0.13%  237.10us         1  237.10us  237.10us  237.10us  cudaMalloc
                    0.11%  213.20us         1  213.20us  213.20us  213.20us  cudaFree
                    0.09%  168.90us         1  168.90us  168.90us  168.90us  cuDeviceTotalMem
                    0.02%  41.600us         4  10.400us  3.2000us  22.900us  cudaEventRecord
                    0.02%  41.299us         1  41.299us  41.299us  41.299us  cudaLaunchKernel
                    0.01%  24.699us         2  12.349us  10.300us  14.399us  cudaEventSynchronize
                    0.01%  20.200us         1  20.200us  20.200us  20.200us  cuDeviceGetName
                    0.00%  5.8000us         2  2.9000us  2.9000us  2.9000us  cudaEventElapsedTime
                    0.00%  3.2000us         1  3.2000us  3.2000us  3.2000us  cuDeviceGetPCIBusId
                    0.00%  2.1000us         3     700ns     300ns  1.3000us  cuDeviceGetCount
                    0.00%  1.1000us         2     550ns     300ns     800ns  cuDeviceGet
```

Number of warps = 977*1024/32 = 31264

1d_3.cu : <<< 333334, 3 >>>

```
Time to generate using CPU:  956.2 ms
Time to generate using GPU:  71.8 ms
==5549== Profiling application: ./julia
==5549== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   96.54%  8.6850ms         1  8.6850ms  8.6850ms  8.6850ms  kernel(unsigned char*)
                    1.84%  165.18us         1  165.18us  165.18us  165.18us  [CUDA memcpy HtoD]
                    1.63%  146.46us         1  146.46us  146.46us  146.46us  [CUDA memcpy DtoH]
```

```
      API calls:   61.29%  119.66ms         4  29.916ms     700ns  119.65ms  cudaEventCreate
                   32.43%  63.319ms         1  63.319ms  63.319ms  63.319ms  cudaDeviceReset
                    5.08%  9.9247ms         2  4.9624ms  274.10us  9.6506ms  cudaMemcpy
                    0.83%  1.6273ms        96  16.950us     100ns  1.4812ms  cuDeviceGetAttribute
                    0.12%  234.70us         1  234.70us  234.70us  234.70us  cudaMalloc
                    0.09%  175.30us         1  175.30us  175.30us  175.30us  cudaFree
                    0.09%  166.50us         1  166.50us  166.50us  166.50us  cuDeviceTotalMem
                    0.02%  43.698us         4  10.924us  3.8000us  25.199us  cudaEventRecord
                    0.02%  38.199us         1  38.199us  38.199us  38.199us  cudaLaunchKernel
                    0.01%  23.900us         2  11.950us  9.6000us  14.300us  cudaEventSynchronize
                    0.01%  20.400us         1  20.400us  20.400us  20.400us  cuDeviceGetName
                    0.00%  5.5000us         2  2.7500us  2.5000us  3.0000us  cudaEventElapsedTime
                    0.00%  2.4000us         1  2.4000us  2.4000us  2.4000us  cuDeviceGetPCIBusId
                    0.00%  1.9000us         3     633ns     300ns  1.0000us  cuDeviceGetCount
                    0.00%  1.1000us         2     550ns     200ns     900ns  cuDeviceGet
```

Number of warps = 333334

1d_4.cu : <<< 1954, 512 >>>

```
Time to generate using CPU:  924.3 ms
Time to generate using GPU:  63.1 ms
==5963== Profiling application: ./julia
==5963== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   64.51%  564.09us         1  564.09us  564.09us  564.09us  kernel(unsigned char*)
                   18.80%  164.38us         1  164.38us  164.38us  164.38us  [CUDA memcpy HtoD]
                   16.70%  146.01us         1  146.01us  146.01us  146.01us  [CUDA memcpy DtoH]
      API calls:   62.00%  114.61ms         4  28.652ms     800ns  114.60ms  cudaEventCreate
                   35.92%  66.403ms         1  66.403ms  66.403ms  66.403ms  cudaDeviceReset
                    0.90%  1.6730ms         2  836.49us  268.60us  1.4044ms  cudaMemcpy
                    0.79%  1.4573ms        96  15.180us     100ns  1.3343ms  cuDeviceGetAttribute
                    0.12%  216.90us         1  216.90us  216.90us  216.90us  cudaFree
                    0.12%  213.70us         1  213.70us  213.70us  213.70us  cudaMalloc
                    0.09%  161.10us         1  161.10us  161.10us  161.10us  cuDeviceTotalMem
                    0.02%  39.399us         4  9.8490us  3.1000us  21.600us  cudaEventRecord
                    0.02%  38.700us         1  38.700us  38.700us  38.700us  cudaLaunchKernel
                    0.01%  24.000us         2  12.000us  10.400us  13.600us  cudaEventSynchronize
                    0.01%  18.100us         1  18.100us  18.100us  18.100us  cuDeviceGetName
                    0.00%  6.0000us         2  3.0000us  2.9000us  3.1000us  cudaEventElapsedTime
                    0.00%  2.9000us         1  2.9000us  2.9000us  2.9000us  cuDeviceGetPCIBusId
                    0.00%  1.7000us         3     566ns     200ns  1.0000us  cuDeviceGetCount
                    0.00%  1.1000us         2     550ns     200ns     900ns  cuDeviceGet
```

Number of warps = 1954*512/32 = 31264

1d_5.cu : <<< 7813, 128 >>>

```
Time to generate using CPU:  961.4 ms
Time to generate using GPU:  64.9 ms
==6353== Profiling application: ./julia
==6353== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   62.90%  538.23us         1  538.23us  538.23us  538.23us  kernel(unsigned char*)
                   20.04%  171.45us         1  171.45us  171.45us  171.45us  [CUDA memcpy HtoD]
                   17.06%  146.02us         1  146.02us  146.02us  146.02us  [CUDA memcpy DtoH]
      API calls:   66.63%  140.22ms         4  35.055ms     600ns  140.21ms  cudaEventCreate
                   31.29%  65.839ms         1  65.839ms  65.839ms  65.839ms  cudaDeviceReset
                    0.86%  1.8055ms        96  18.807us     100ns  1.6704ms  cuDeviceGetAttribute
                    0.85%  1.7897ms         2  894.84us  296.50us  1.4932ms  cudaMemcpy
                    0.11%  241.50us         1  241.50us  241.50us  241.50us  cudaMalloc
                    0.11%  223.00us         1  223.00us  223.00us  223.00us  cudaFree
                    0.08%  174.60us         1  174.60us  174.60us  174.60us  cuDeviceTotalMem
                    0.02%  43.300us         1  43.300us  43.300us  43.300us  cudaLaunchKernel
                    0.02%  43.200us         4  10.800us  3.4000us  24.000us  cudaEventRecord
                    0.01%  24.999us         1  24.999us  24.999us  24.999us  cuDeviceGetName
                    0.01%  23.998us         2  11.999us  9.8990us  14.099us  cudaEventSynchronize
                    0.00%  5.7000us         2  2.8500us  2.8000us  2.9000us  cudaEventElapsedTime
                    0.00%  3.1000us         1  3.1000us  3.1000us  3.1000us  cuDeviceGetPCIBusId
                    0.00%  2.4000us         3     800ns     200ns  1.7000us  cuDeviceGetCount
                    0.00%  1.1000us         2     550ns     200ns     900ns  cuDeviceGet
```

Number of warps = 7813*128/32 = 31252

2D indexing

2d_1.cu : <<< (DIM, DIM), (1, 1) >>>

```
Time to generate using CPU:  919.2 ms
Time to generate using GPU:  83.0 ms
==6679== Profiling application: ./julia
==6679== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   98.59%  21.688ms         1  21.688ms  21.688ms  21.688ms  kernel(unsigned char*)
                    0.75%  164.58us         1  164.58us  164.58us  164.58us  [CUDA memcpy HtoD]
                    0.67%  146.37us         1  146.37us  146.37us  146.37us  [CUDA memcpy DtoH]
      API calls:   58.84%  122.93ms         4  30.731ms     600ns  122.92ms  cudaEventCreate
                   29.13%  60.862ms         1  60.862ms  60.862ms  60.862ms  cudaDeviceReset
                   10.93%  22.827ms         2  11.413ms  264.80us  22.562ms  cudaMemcpy
                    0.77%  1.6174ms        96  16.847us     100ns  1.4935ms  cuDeviceGetAttribute
                    0.11%  220.10us         1  220.10us  220.10us  220.10us  cudaMalloc
                    0.08%  169.20us         1  169.20us  169.20us  169.20us  cudaFree
                    0.08%  162.10us         1  162.10us  162.10us  162.10us  cuDeviceTotalMem
                    0.02%  37.400us         4  9.3500us  3.1000us  21.700us  cudaEventRecord
                    0.02%  37.100us         1  37.100us  37.100us  37.100us  cudaLaunchKernel
                    0.01%  22.099us         2  11.049us  9.2000us  12.899us  cudaEventSynchronize
                    0.01%  17.399us         1  17.399us  17.399us  17.399us  cuDeviceGetName
                    0.00%  5.7000us         2  2.8500us  2.8000us  2.9000us  cudaEventElapsedTime
                    0.00%  2.7000us         1  2.7000us  2.7000us  2.7000us  cuDeviceGetPCIBusId
                    0.00%  1.8000us         3     600ns     200ns  1.1000us  cuDeviceGetCount
                    0.00%  1.0000us         2     500ns     200ns     800ns  cuDeviceGet
```

Number of warps = 1000*1000 = 1000000

2d_2.cu : <<< (32, 32), (32, 32) >>>

```
Time to generate using CPU:  947.5 ms
Time to generate using GPU:  65.0 ms
==7960== Profiling application: ./julia
==7960== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   66.41%  614.94us         1  614.94us  614.94us  614.94us  kernel(unsigned char*)
                   17.83%  165.09us         1  165.09us  165.09us  165.09us  [CUDA memcpy HtoD]
                   15.76%  145.98us         1  145.98us  145.98us  145.98us  [CUDA memcpy DtoH]
      API calls:   61.68%  121.37ms         4  30.343ms     700ns  121.36ms  cudaEventCreate
                   36.18%  71.192ms         1  71.192ms  71.192ms  71.192ms  cudaDeviceReset
                    0.92%  1.8183ms         2  909.14us  272.40us  1.5459ms  cudaMemcpy
                    0.81%  1.5980ms        96  16.645us     100ns  1.4648ms  cuDeviceGetAttribute
                    0.13%  256.20us         1  256.20us  256.20us  256.20us  cudaMalloc
                    0.11%  224.60us         1  224.60us  224.60us  224.60us  cudaFree
                    0.09%  173.90us         1  173.90us  173.90us  173.90us  cuDeviceTotalMem
                    0.02%  43.800us         4  10.950us  3.5000us  24.300us  cudaEventRecord
                    0.02%  39.699us         1  39.699us  39.699us  39.699us  cudaLaunchKernel
                    0.01%  23.100us         2  11.550us  10.400us  12.700us  cudaEventSynchronize
                    0.01%  20.199us         1  20.199us  20.199us  20.199us  cuDeviceGetName
                    0.00%  6.5000us         2  3.2500us  3.0000us  3.5000us  cudaEventElapsedTime
                    0.00%  3.1000us         1  3.1000us  3.1000us  3.1000us  cuDeviceGetPCIBusId
                    0.00%  2.5000us         3     833ns     200ns  1.8000us  cuDeviceGetCount
                    0.00%     900ns         2     450ns     200ns     700ns  cuDeviceGet
```

Number of warps = 32*32*32*32/32 = 32768

2d_3.cu : <<< (334, 334), (3, 3) >>>

```
Time to generate using CPU:  956.4 ms
Time to generate using GPU:  65.6 ms
==8374== Profiling application: ./julia
==8374== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   91.39%  3.3016ms         1  3.3016ms  3.3016ms  3.3016ms  kernel(unsigned char*)
                    4.57%  165.12us         1  165.12us  165.12us  165.12us  [CUDA memcpy HtoD]
                    4.04%  145.98us         1  145.98us  145.98us  145.98us  [CUDA memcpy DtoH]
      API calls:   64.85%  122.27ms         4  30.566ms     700ns  122.26ms  cudaEventCreate
                   31.60%  59.580ms         1  59.580ms  59.580ms  59.580ms  cudaDeviceReset
                    2.35%  4.4353ms         2  2.2177ms  304.10us  4.1312ms  cudaMemcpy
                    0.83%  1.5619ms        96  16.269us     100ns  1.4370ms  cuDeviceGetAttribute
                    0.13%  238.30us         1  238.30us  238.30us  238.30us  cudaMalloc
                    0.09%  162.70us         1  162.70us  162.70us  162.70us  cuDeviceTotalMem
```

```
                    0.08%  151.70us         1  151.70us  151.70us  151.70us  cudaFree
                    0.02%  38.500us         1  38.500us  38.500us  38.500us  cudaLaunchKernel
                    0.02%  37.500us         4  9.3750us  2.9000us  23.300us  cudaEventRecord
                    0.01%  23.400us         2  11.700us  10.000us  13.400us  cudaEventSynchronize
                    0.01%  18.000us         1  18.000us  18.000us  18.000us  cuDeviceGetName
                    0.00%  6.0000us         2  3.0000us  2.9000us  3.1000us  cudaEventElapsedTime
                    0.00%  3.0000us         1  3.0000us  3.0000us  3.0000us  cuDeviceGetPCIBusId
                    0.00%  1.7000us         3     566ns     200ns     900ns  cuDeviceGetCount
                    0.00%     899ns         2     449ns     200ns     699ns  cuDeviceGet
```

Number of warps = 334*334 = 111556

2d_4.cu : <<< (63, 63), (16, 16) >>>

```
Time to generate using CPU:  948.8 ms
Time to generate using GPU:  63.5 ms
==8813== Profiling application: ./julia
==8813== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   61.29%  491.93us         1  491.93us  491.93us  491.93us  kernel(unsigned char*)
                   20.52%  164.70us         1  164.70us  164.70us  164.70us  [CUDA memcpy HtoD]
                   18.19%  146.01us         1  146.01us  146.01us  146.01us  [CUDA memcpy DtoH]
      API calls:   62.47%  116.26ms         4  29.065ms     800ns  116.25ms  cudaEventCreate
                   35.26%  65.609ms         1  65.609ms  65.609ms  65.609ms  cudaDeviceReset
                    0.94%  1.7567ms        96  18.298us      99ns  1.6292ms  cuDeviceGetAttribute
                    0.93%  1.7381ms         2  869.04us  284.30us  1.4538ms  cudaMemcpy
                    0.13%  250.20us         1  250.20us  250.20us  250.20us  cudaMalloc
                    0.10%  178.60us         1  178.60us  178.60us  178.60us  cudaFree
                    0.09%  164.00us         1  164.00us  164.00us  164.00us  cuDeviceTotalMem
                    0.02%  42.600us         4  10.650us  3.6000us  25.400us  cudaEventRecord
                    0.02%  38.700us         1  38.700us  38.700us  38.700us  cudaLaunchKernel
                    0.01%  22.900us         2  11.450us  9.9000us  13.000us  cudaEventSynchronize
                    0.01%  18.800us         1  18.800us  18.800us  18.800us  cuDeviceGetName
                    0.00%  6.4000us         2  3.2000us  3.1000us  3.3000us  cudaEventElapsedTime
                    0.00%  2.9000us         1  2.9000us  2.9000us  2.9000us  cuDeviceGetPCIBusId
                    0.00%  2.2000us         3     733ns     200ns  1.2000us  cuDeviceGetCount
                    0.00%  1.3000us         2     650ns     300ns  1.0000us  cuDeviceGet
```

Number of warps = 63*63*16*16/32 = 31752

2d_5.cu : <<< (125, 125), (8, 8) >>>

```
Time to generate using CPU:  949.4 ms
Time to generate using GPU:  63.7 ms
==9176== Profiling application: ./julia
==9176== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   66.92%  628.44us         1  628.44us  628.44us  628.44us  kernel(unsigned char*)
                   17.54%  164.70us         1  164.70us  164.70us  164.70us  [CUDA memcpy HtoD]
                   15.55%  146.02us         1  146.02us  146.02us  146.02us  [CUDA memcpy DtoH]
      API calls:   63.89%  123.00ms         4  30.750ms     700ns  122.99ms  cudaEventCreate
                   33.87%  65.211ms         1  65.211ms  65.211ms  65.211ms  cudaDeviceReset
                    0.95%  1.8282ms         2  914.09us  265.40us  1.5628ms  cudaMemcpy
                    0.89%  1.7218ms        96  17.935us     100ns  1.5934ms  cuDeviceGetAttribute
                    0.13%  241.60us         1  241.60us  241.60us  241.60us  cudaMalloc
                    0.11%  208.10us         1  208.10us  208.10us  208.10us  cudaFree
                    0.09%  165.40us         1  165.40us  165.40us  165.40us  cuDeviceTotalMem
                    0.02%  40.700us         4  10.175us  3.0000us  21.100us  cudaEventRecord
                    0.02%  37.199us         1  37.199us  37.199us  37.199us  cudaLaunchKernel
                    0.01%  23.200us         2  11.600us  10.400us  12.800us  cudaEventSynchronize
                    0.01%  19.400us         1  19.400us  19.400us  19.400us  cuDeviceGetName
                    0.00%  6.2000us         2  3.1000us  3.0000us  3.2000us  cudaEventElapsedTime
                    0.00%  3.2000us         1  3.2000us  3.2000us  3.2000us  cuDeviceGetPCIBusId
                    0.00%  1.9000us         3     633ns     200ns  1.2000us  cuDeviceGetCount
                    0.00%  1.1000us         2     550ns     300ns     800ns  cuDeviceGet
```

Number of warps = 125*125*8*8/32 = 31250

Report

By comparing the results, it is observed that the execution time of the kernel(unsigned char*) varies a lot. There are four combinations that used longer than 1ms : <<< DIM * DIM, 1 >>>, <<< 333334, 3 >>>, <<< (DIM, DIM), (1, 1) >>>, <<< (334, 334), (3, 3) >>>, both of them have lower number of threads per block and larger number of warps. In fact, each block is divided into warps (32 threads) and warps are scheduled to the functional units for execution. For example, there is only 1 thread per warp using <<< DIM * DIM, 1 >>>, the rest are inactively "masked out". To obtain optimal efficiency, the number of threads per block should be in multiples of 32. Otherwise, lots of computing power will be wasted, as the hardware will not coalesce threads from different warps. The larger the number of warps, the longer the execution time of kernel(unsigned char*). Comparing <<< DIM * DIM, 1 >>> with <<< (32, 32), (32, 32) >>>, 1000000/32768 ≈ 30.5 and 21.877ms/614.94us ≈ 35.6. Also, block size should be large enough, so there will be enough number of warps to hide memory access latency. After all, using 1D indexing or 2D indexing does not make a noticeable difference in execution time in this case, as long as the number of threads per block is in multiples of 32 and large enough.

1D array is actually just a concatenation of items of a two-dimensional array. To use 1D indexing to generate Julia set, people would imagine a 1D array as points distributed along a line segment and move on to distribute regularly on a 2D plane, e.g. `ptr[tid] = 255 * julia(tid % DIM, tid / DIM)`. However, to use 2D indexing, we just need to compute row index and column index of pixels, which intuitively matches up with the image and is much easier to understand, e.g. `ptr[tid] = 255 * julia(y, x)`. Therefore, we should use 1D indexing when we need to keep track of data in linear order, and use 2D indexing in certain situations to visualise the data, such as digital image processing.

In addition, the execution time on GPU is much short than the execution time on CPU. Using 2d_2.cu (`<<< (32, 32), (32, 32) >>>`) as an example, the time to generate on CPU is 947.5ms and 65.0ms on GPU. The speedup on GPU is about 14.6 times faster than on CPU. It is because GPUs focus on data processing rather than data caching and flow control. A GPU has multiple SMs (13 on K80 in Azure virtual machine), each with multiple execution units (e.g. DP cores, CUDA cores, special function units). Although the clock rate is lower

than CPU, there are thousands of CUDA Cores on GPU to perform operations in parallel with high performance.

Gaussian blur

1.cu : filter as localVar, input image as sharedVar

Time spent on executing your_gaussian_blur_func() (in ms)										Fastest
5.9	6.1	5.9	6.0	6.0	5.9	6.1	6.0	6.0	5.9	5.9

```
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z11blur_kernelP8PPMImageP8PPMPixel' for 'sm_35'
ptxas info    : Function properties for _Z11blur_kernelP8PPMImageP8PPMPixel
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 25 registers, 432 bytes smem, 336 bytes cmem[0], 12 bytes cmem[2]

Time to generate:  5.9 ms
==7522== Profiling application: ./blur
==7522== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   36.57%  1.2454ms         1  1.2454ms  1.2454ms  1.2454ms  blur_kernel(PPMImage*, PPMPixel*)
                   33.00%  1.1238ms         2  561.90us  2.1440us  1.1217ms  [CUDA memcpy HtoD]
                   30.43%  1.0361ms         1  1.0361ms  1.0361ms  1.0361ms  [CUDA memcpy DtoH]
      API calls:   93.53%  109.19ms         2  54.595ms  1.3000us  109.19ms  cudaEventCreate
                    2.96%  3.4603ms         3  1.1534ms  14.800us  2.3762ms  cudaMemcpy
                    1.67%  1.9538ms         3  651.25us  129.10us  917.58us  cudaFree
                    1.22%  1.4297ms        96  14.892us      99ns  1.3096ms  cuDeviceGetAttribute
                    0.40%  471.89us         3  157.30us  104.10us  230.20us  cudaMalloc
                    0.14%  159.40us         1  159.40us  159.40us  159.40us  cuDeviceTotalMem
                    0.03%  33.399us         1  33.399us  33.399us  33.399us  cudaLaunchKernel
                    0.01%  15.600us         1  15.600us  15.600us  15.600us  cuDeviceGetName
                    0.01%  10.300us         2  5.1500us  4.6000us  5.7000us  cudaEventRecord
                    0.01%  9.2000us         1  9.2000us  9.2000us  9.2000us  cudaEventSynchronize
                    0.00%  2.6000us         1  2.6000us  2.6000us  2.6000us  cuDeviceGetPCIBusId
                    0.00%  2.6000us         1  2.6000us  2.6000us  2.6000us  cudaEventElapsedTime
                    0.00%  1.6000us         3     533ns     200ns     900ns  cuDeviceGetCount
                    0.00%  1.0000us         2     500ns     200ns     800ns  cuDeviceGet
```

2.cu : filter as localVar, input image as GlobalVar

Time spent on executing your_gaussian_blur_func() (in ms)										Fastest
7.8	7.8	8.0	7.9	8.3	7.8	7.8	7.8	7.8	7.8	7.8

```
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z11blur_kernelP8PPMImageP8PPMPixel' for 'sm_35'
ptxas info    : Function properties for _Z11blur_kernelP8PPMImageP8PPMPixel
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 21 registers, 336 bytes cmem[0], 12 bytes cmem[2]

Time to generate:  7.8 ms
==3427== Profiling application: ./blur
==3427== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   58.85%  3.0849ms         1  3.0849ms  3.0849ms  3.0849ms  blur_kernel(PPMImage*, PPMPixel*)
                   21.44%  1.1237ms         2  561.87us  2.1440us  1.1216ms  [CUDA memcpy HtoD]
                   19.72%  1.0336ms         1  1.0336ms  1.0336ms  1.0336ms  [CUDA memcpy DtoH]
      API calls:   92.07%  109.97ms         2  54.986ms  1.3000us  109.97ms  cudaEventCreate
                    4.43%  5.2934ms         3  1.7645ms  14.400us  4.2097ms  cudaMemcpy
                    1.64%  1.9594ms         3  653.12us  116.80us  924.99us  cudaFree
                    1.27%  1.5167ms        96  15.798us      100ns  1.3914ms  cuDeviceGetAttribute
                    0.39%  467.09us         3  155.70us  105.30us  229.40us  cudaMalloc
                    0.13%  160.30us         1  160.30us  160.30us  160.30us  cuDeviceTotalMem
                    0.03%  32.300us         1  32.300us  32.300us  32.300us  cudaLaunchKernel
                    0.02%  18.400us         1  18.400us  18.400us  18.400us  cuDeviceGetName
                    0.01%  10.500us         2  5.2500us  5.0000us  5.5000us  cudaEventRecord
                    0.01%  9.1000us         1  9.1000us  9.1000us  9.1000us  cudaEventSynchronize
                    0.00%  2.5000us         1  2.5000us  2.5000us  2.5000us  cuDeviceGetPCIBusId
                    0.00%  2.4000us         1  2.4000us  2.4000us  2.4000us  cudaEventElapsedTime
                    0.00%  1.8000us         3     600ns     100ns  1.0000us  cuDeviceGetCount
                    0.00%  1.0000us         2     500ns     200ns     800ns  cuDeviceGet
```

3.cu : filter as ConstantVar, input image as sharedVar

Time spent on executing your_gaussian_blur_func() (in ms)										Fastest
6.3	6.2	6.3	6.3	6.3	6.6	6.4	6.3	6.3	6.3	6.2

```
ptxas info    : 0 bytes gmem, 36 bytes cmem[3]
ptxas info    : Compiling entry function '_Z11blur_kernelP8PPMImageP8PPMPixel' for 'sm_35'
ptxas info    : Function properties for _Z11blur_kernelP8PPMImageP8PPMPixel
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 25 registers, 432 bytes smem, 336 bytes cmem[0]

Time to generate:  6.2 ms
==8970== Profiling application: ./blur
==8970== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   41.98%  1.5630ms         1  1.5630ms  1.5630ms  1.5630ms  blur_kernel(PPMImage*, PPMPixel*)
                   30.20%  1.1246ms         2  562.29us  2.1440us  1.1224ms  [CUDA memcpy HtoD]
                   27.82%  1.0358ms         1  1.0358ms  1.0358ms  1.0358ms  [CUDA memcpy DtoH]
      API calls:   93.41%  114.99ms         2  57.493ms  4.0000us  114.98ms  cudaEventCreate
                    3.07%  3.7750ms         3  1.2583ms  14.500us  2.6948ms  cudaMemcpy
                    1.59%  1.9555ms         3  651.82us  112.50us  923.49us  cudaFree
                    1.36%  1.6782ms        96  17.480us     100ns  1.5540ms  cuDeviceGetAttribute
                    0.38%  464.49us         3  154.83us  112.70us  226.70us  cudaMalloc
                    0.13%  163.10us         1  163.10us  163.10us  163.10us  cuDeviceTotalMem
                    0.02%  28.099us         1  28.099us  28.099us  28.099us  cudaLaunchKernel
                    0.01%  18.400us         1  18.400us  18.400us  18.400us  cuDeviceGetName
                    0.01%  10.100us         2  5.0500us  4.8000us  5.3000us  cudaEventRecord
                    0.01%  10.000us         1  10.000us  10.000us  10.000us  cudaEventSynchronize
                    0.00%  2.5000us         1  2.5000us  2.5000us  2.5000us  cuDeviceGetPCIBusId
                    0.00%  2.5000us         1  2.5000us  2.5000us  2.5000us  cudaEventElapsedTime
                    0.00%  1.7000us         3     566ns     100ns  1.0000us  cuDeviceGetCount
                    0.00%  1.0000us         2     500ns     200ns     800ns  cuDeviceGet
```

4.cu : filter as ConstantVar, input image as GlobalVar

Time spent on executing your_gaussian_blur_func() (in ms)										Fastest
7.8	7.7	7.8	7.8	8.0	7.9	8.5	7.8	7.7	7.8	7.7

```
ptxas info    : 0 bytes gmem, 36 bytes cmem[3]
ptxas info    : Compiling entry function '_Z11blur_kernelP8PPMImageP8PPMPixel' for 'sm_35'
ptxas info    : Function properties for _Z11blur_kernelP8PPMImageP8PPMPixel
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 21 registers, 336 bytes cmem[0]

Time to generate:  7.7 ms
==4615== Profiling application: ./blur
==4615== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   58.66%  3.0588ms         1  3.0588ms  3.0588ms  3.0588ms  blur_kernel(PPMImage*, PPMPixel*)
                   21.55%  1.1236ms         2  561.80us  2.1440us  1.1215ms  [CUDA memcpy HtoD]
                   19.79%  1.0322ms         1  1.0322ms  1.0322ms  1.0322ms  [CUDA memcpy DtoH]
      API calls:   92.16%  110.69ms         2  55.345ms  1.7000us  110.69ms  cudaEventCreate
                    4.39%  5.2780ms         3  1.7593ms  14.900us  4.1879ms  cudaMemcpy
                    1.63%  1.9566ms         3  652.19us  114.60us  923.58us  cudaFree
                    1.24%  1.4873ms        96  15.492us     100ns  1.3666ms  cuDeviceGetAttribute
                    0.38%  456.49us         3  152.16us  110.10us  221.50us  cudaMalloc
                    0.13%  161.00us         1  161.00us  161.00us  161.00us  cuDeviceTotalMem
                    0.02%  29.600us         1  29.600us  29.600us  29.600us  cudaLaunchKernel
                    0.01%  16.300us         1  16.300us  16.300us  16.300us  cuDeviceGetName
                    0.01%  10.499us         2  5.2490us  5.1990us  5.3000us  cudaEventRecord
                    0.01%  9.6000us         1  9.6000us  9.6000us  9.6000us  cudaEventSynchronize
                    0.00%  2.6000us         1  2.6000us  2.6000us  2.6000us  cudaEventElapsedTime
                    0.00%  2.4000us         1  2.4000us  2.4000us  2.4000us  cuDeviceGetPCIBusId
                    0.00%  1.7000us         3     566ns     100ns  1.0000us  cuDeviceGetCount
                    0.00%  1.0000us         2     500ns     200ns     800ns  cuDeviceGet
```

5.cu : filter as GlobalVar, input image as sharedVar

Time spent on executing your_gaussian_blur_func() (in ms)										Fastest
7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.6	7.5

```
ptxas info    : 36 bytes gmem
ptxas info    : Compiling entry function '_Z11blur_kernelP8PPMImageP8PPMPixel' for 'sm_35'
ptxas info    : Function properties for _Z11blur_kernelP8PPMImageP8PPMPixel
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 25 registers, 432 bytes smem, 336 bytes cmem[0]

Time to generate:  7.5 ms
==9866== Profiling application: ./blur
==9866== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   56.44%  2.7955ms         1  2.7955ms  2.7955ms  2.7955ms  blur_kernel(PPMImage*, PPMPixel*)
                   22.69%  1.1237ms         2  561.87us  2.1440us  1.1216ms  [CUDA memcpy HtoD]
                   20.87%  1.0336ms         1  1.0336ms  1.0336ms  1.0336ms  [CUDA memcpy DtoH]
      API calls:   92.27%  112.47ms         2  56.237ms  1.5000us  112.47ms  cudaEventCreate
                    4.13%  5.0295ms         3  1.6765ms  14.200us  3.9462ms  cudaMemcpy
                    1.61%  1.9624ms         3  654.12us  116.30us  923.29us  cudaFree
                    1.41%  1.7165ms        96  17.879us      99ns  1.5813ms  cuDeviceGetAttribute
                    0.38%  468.59us         3  156.20us  111.20us  226.10us  cudaMalloc
                    0.13%  161.50us         1  161.50us  161.50us  161.50us  cuDeviceTotalMem
                    0.03%  31.100us         1  31.100us  31.100us  31.100us  cudaLaunchKernel
                    0.02%  20.299us         1  20.299us  20.299us  20.299us  cuDeviceGetName
                    0.01%  11.000us         2  5.5000us  5.3000us  5.7000us  cudaEventRecord
                    0.01%  9.7000us         1  9.7000us  9.7000us  9.7000us  cudaEventSynchronize
                    0.00%  2.7000us         1  2.7000us  2.7000us  2.7000us  cudaEventElapsedTime
                    0.00%  2.6000us         1  2.6000us  2.6000us  2.6000us  cuDeviceGetPCIBusId
                    0.00%  2.2000us         3     733ns     200ns  1.2000us  cuDeviceGetCount
                    0.00%  1.0990us         2     549ns     300ns     799ns  cuDeviceGet
```

6.cu : filter as GlobalVar, input image as GlobalVar

Time spent on executing your_gaussian_blur_func() (in ms)										Fastest
8.3	8.3	8.4	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3

```
ptxas info    : 36 bytes gmem
ptxas info    : Compiling entry function '_Z11blur_kernelP8PPMImageP8PPMPixel' for 'sm_35'
ptxas info    : Function properties for _Z11blur_kernelP8PPMImageP8PPMPixel
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 31 registers, 336 bytes cmem[0]

Time to generate:  8.3 ms
==5411== Profiling application: ./blur
==5411== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   62.44%  3.5918ms         1  3.5918ms  3.5918ms  3.5918ms  blur_kernel(PPMImage*, PPMPixel*)
                   19.54%  1.1240ms         2  562.00us  2.1440us  1.1219ms  [CUDA memcpy HtoD]
                   18.02%  1.0367ms         1  1.0367ms  1.0367ms  1.0367ms  [CUDA memcpy DtoH]
      API calls:   91.75%  111.39ms         2  55.694ms  1.3000us  111.39ms  cudaEventCreate
                    4.81%  5.8350ms         3  1.9450ms  14.200us  4.7219ms  cudaMemcpy
                    1.61%  1.9555ms         3  651.82us  111.80us  922.99us  cudaFree
                    1.25%  1.5189ms        96  15.821us     100ns  1.3952ms  cuDeviceGetAttribute
                    0.38%  460.49us         3  153.50us  103.60us  232.80us  cudaMalloc
                    0.14%  171.40us         1  171.40us  171.40us  171.40us  cuDeviceTotalMem
                    0.02%  28.100us         1  28.100us  28.100us  28.100us  cudaLaunchKernel
                    0.01%  16.699us         1  16.699us  16.699us  16.699us  cuDeviceGetName
                    0.01%  10.500us         1  10.500us  10.500us  10.500us  cudaEventSynchronize
                    0.01%  10.098us         2  5.0490us  4.5990us  5.4990us  cudaEventRecord
                    0.00%  2.7000us         1  2.7000us  2.7000us  2.7000us  cuDeviceGetPCIBusId
                    0.00%  2.5000us         1  2.5000us  2.5000us  2.5000us  cudaEventElapsedTime
                    0.00%  1.8000us         3     600ns     200ns  1.1000us  cuDeviceGetCount
                    0.00%  1.1000us         2     550ns     300ns     800ns  cuDeviceGet
```

Results

Time spent on executing your_gaussian_blur_func()		Filter		
		localVar	ConstantVar	GlobalVar
Input Image	sharedVar	5.9ms	6.2ms	7.5ms
	GlobalVar	7.8ms	7.7ms	8.3ms

Time: 1 < 3 < 5 < 4 < 2 < 6

Report

To start with, programs declaring the input image as sharedVar apparently consume less time than declaring as GlobalVar (1 < 2, 3 < 4, 5 < 6). As shared memory is on-chip, it serves as a cache for all threads in a block and normally uses 1-3 cycles for memory accesses. However, global memory is off-chip, and it normally uses 300-800 cycles for memory accesses. Although on newer GPUs, all global memory accesses are cached to L2 cache (~50-100 cycles), it is still far slower than shared memory. Shared memory has shorter latency and higher bandwidth than global memory. Also, an input image contains lots of data (e.g. RGB values of 1900x1200 pixels), so the speed difference between two kinds of memory become more obvious. For example, program 1 used ~1.2ms to execute blur_kernel while program 2 used ~3.1ms, the difference is ~1.9ms and is sufficient for program 1 to execute blur_kernel once again.

In addition, programs declaring the filter as localVar and ConstantVar consume similar time and both less than declaring as GlobalVar. Registers allow fast access of data in a single thread (~1 cycle), and constant memory is optimised for broadcast (~5 cycles), which used to cache values shared by all functional units. Both of them minimize global memory accesses. In most cases, assessing registers is the fastest and assessing global memory is the slowest. But program 4 (7.7ms) used less time than program 2 (7.8ms) when executing your_gaussian_blur_func. Actually, there are many more factors affecting the speed of memory, like bank conflicts and data dependencies. It will be discussed in the following.

One may afraid that the execution time of kernels and functions are so small, hence this will introduce uncertainty into the measurement. Sure, a result is often not at all accurate. We have to run a program N times, then the standard deviation will become smaller and the result will be more accurate. In fact, the results are quite similar, just vary within ~0.3ms. A question comes into mind: why we choose the fast one rather than the average? Of course, choosing the average one can better represent the actual execution time in

daily situations. But what we want is to determine the fastest approach to finish the tasks. There are many situations that will affect the execution time, such as memory latency divergence, memory bank divergence, and branch divergence. Also, the profiler adds latency since it involves host-device transfers. We have to test the programs several times to figure out the best approach, so as to maximise the use of hardware and use as little slow-access memory as possible.

In a nutshell, declaring the filter as localVar and the input image as sharedVar is the fastest way. The speed comparison is as follows: registers > shared memory > constant memory > global memory. We should reduce direct accesses to global memory. However, the amount of registers, shared memory, and constant memory are quite limited. For instance, total shared memory per block in the CUDA device (K80) is 49152. It is impossible to store a very large array in the shared memory, therefore we can’t store the whole image (1900x1200 RGB values) in it. In each shared memory, I just store the pixels according to the indexes of threads and the surrounding pixels. It is noteworthy that the lines of code of program 2, 4, 6 are longer, the comparison may not be that accurate.