Parallel Computing

Lab 2 - Report

Name	Sec	BN	ID
بموا عريان عياد	1	17	9202391
مارك ياسر نبيل	2	14	9203106

Input of size 1000 x 1000 (Rows = Columns)

Kernel 1

```
!nvprof ./k1.out "q1_1000_1000.txt" "k1_1000_1000_output.txt"
```

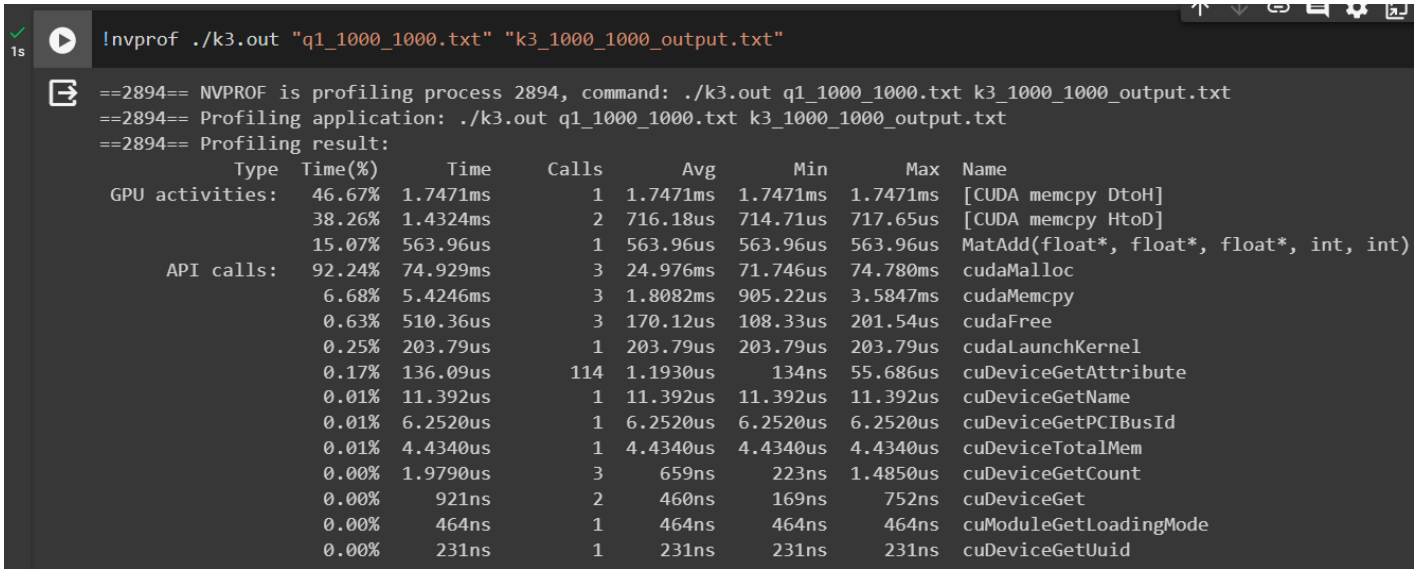
```
==2858== NVPROF is profiling process 2858, command: ./k1.out q1_1000_1000.txt k1_1000_1000_output.txt
==2858== Profiling application: ./k1.out q1_1000_1000.txt k1_1000_1000_output.txt
==2858== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   53.47%  1.7045ms         1  1.7045ms  1.7045ms  1.7045ms  [CUDA memcpy DtoH]
                   44.91%  1.4316ms         2  715.81us  710.87us  720.75us  [CUDA memcpy HtoD]
                    1.61%  51.455us         1  51.455us  51.455us  51.455us  MatAdd(float*, float*, float*, int, int)
      API calls:   94.76%  104.90ms         3  34.966ms  78.251us  104.72ms  cudaMalloc
                    4.43%  4.9060ms         3  1.6353ms  896.20us  3.0780ms  cudaMemcpy
                    0.50%  554.23us         3  184.74us  114.76us  231.82us  cudaFree
                    0.17%  185.62us         1  185.62us  185.62us  185.62us  cudaLaunchKernel
                    0.12%  135.19us       114  1.1850us     137ns  53.943us  cuDeviceGetAttribute
                    0.01%  11.965us         1  11.965us  11.965us  11.965us  cuDeviceGetName
                    0.00%  5.2680us         1  5.2680us  5.2680us  5.2680us  cuDeviceGetPCIBusId
                    0.00%  4.1340us         1  4.1340us  4.1340us  4.1340us  cuDeviceTotalMem
                    0.00%  1.7140us         3     571ns     209ns  1.2210us  cuDeviceGetCount
                    0.00%     865ns         2     432ns     177ns     688ns  cuDeviceGet
                    0.00%     541ns         1     541ns     541ns     541ns  cuModuleGetLoadingMode
                    0.00%     196ns         1     196ns     196ns     196ns  cuDeviceGetUuid
```

Kernel 2

```
!nvprof ./k2.out "q1_1000_1000.txt" "k2_1000_1000_output.txt"
```

```
==2879== NVPROF is profiling process 2879, command: ./k2.out q1_1000_1000.txt k2_1000_1000_output.txt
==2879== Profiling application: ./k2.out q1_1000_1000.txt k2_1000_1000_output.txt
==2879== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   48.50%  1.6813ms         1  1.6813ms  1.6813ms  1.6813ms  [CUDA memcpy DtoH]
                   42.96%  1.4890ms         2  744.51us  716.82us  772.21us  [CUDA memcpy HtoD]
                    8.54%  295.96us         1  295.96us  295.96us  295.96us  MatAdd(float*, float*, float*, int, int)
      API calls:   93.95%  96.381ms         3  32.127ms  116.86us  96.116ms  cudaMalloc
                    5.18%  5.3153ms         3  1.7718ms  958.00us  3.3521ms  cudaMemcpy
                    0.51%  519.73us         3  173.24us  110.39us  208.36us  cudaFree
                    0.20%  204.75us         1  204.75us  204.75us  204.75us  cudaLaunchKernel
                    0.14%  140.45us       114  1.2320us     148ns  56.015us  cuDeviceGetAttribute
                    0.01%  11.681us         1  11.681us  11.681us  11.681us  cuDeviceGetName
                    0.01%  5.2780us         1  5.2780us  5.2780us  5.2780us  cuDeviceGetPCIBusId
                    0.00%  4.5480us         1  4.5480us  4.5480us  4.5480us  cuDeviceTotalMem
                    0.00%  1.8740us         3     624ns     220ns  1.2850us  cuDeviceGetCount
                    0.00%  1.1060us         2     553ns     197ns     909ns  cuDeviceGet
                    0.00%     538ns         1     538ns     538ns     538ns  cuModuleGetLoadingMode
                    0.00%     241ns         1     241ns     241ns     241ns  cuDeviceGetUuid
```

Kernel 3



```
!nvprof ./k3.out "q1_1000_1000.txt" "k3_1000_1000_output.txt"

==2894== NVPROF is profiling process 2894, command: ./k3.out q1_1000_1000.txt k3_1000_1000_output.txt
==2894== Profiling application: ./k3.out q1_1000_1000.txt k3_1000_1000_output.txt
==2894== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   46.67%  1.7471ms         1  1.7471ms  1.7471ms  1.7471ms  [CUDA memcpy DtoH]
                   38.26%  1.4324ms         2  716.18us  714.71us  717.65us  [CUDA memcpy HtoD]
                   15.07%  563.96us         1  563.96us  563.96us  563.96us  MatAdd(float*, float*, float*, int, int)
      API calls:   92.24%  74.929ms         3  24.976ms  71.746us  74.780ms  cudaMalloc
                    6.68%  5.4246ms         3  1.8082ms  905.22us  3.5847ms  cudaMemcpy
                    0.63%  510.36us         3  170.12us  108.33us  201.54us  cudaFree
                    0.25%  203.79us         1  203.79us  203.79us  203.79us  cudaLaunchKernel
                    0.17%  136.09us       114  1.1930us     134ns  55.686us  cuDeviceGetAttribute
                    0.01%  11.392us         1  11.392us  11.392us  11.392us  cuDeviceGetName
                    0.01%  6.2520us         1  6.2520us  6.2520us  6.2520us  cuDeviceGetPCIBusId
                    0.01%  4.4340us         1  4.4340us  4.4340us  4.4340us  cuDeviceTotalMem
                    0.00%  1.9790us         3     659ns     223ns  1.4850us  cuDeviceGetCount
                    0.00%     921ns         2     460ns     169ns     752ns  cuDeviceGet
                    0.00%     464ns         1     464ns     464ns     464ns  cuModuleGetLoadingMode
                    0.00%     231ns         1     231ns     231ns     231ns  cuDeviceGetUuid
```

Observation 1

Kernel 1 achieves the highest speed (51.455 microseconds) for MatAdd.
Kernel 2 comes 2nd with (259.96 microseconds) for MatAdd.
Kernel 3 comes 3rd with (563.96 microseconds) for MatAdd.

Conclusion 1

Kernel 1 is the fastest due to its utilization of maximum parallelism, with each thread dedicated to computing a single element
Kernel 2 is faster than **Kernel 3** as the operation of fetching 1 row per thread is faster than fetching 1 column as the rows are stored next to each other in memory but columns require additional jumps in memory which makes the retrieval slower, given that both kernels have the same number of blocks and threads because the input matrix is symmetric.

Input of size 500 x 1000 (Rows<Columns)

Kernel 1

```
[18] !nvprof ./k1.out "q1_500_1000.txt" "k1_500_1000_output.txt"

==7096== NVPROF is profiling process 7096, command: ./k1.out q1_500_1000.txt k1_500_1000_output.txt
==7096== Profiling application: ./k1.out q1_500_1000.txt k1_500_1000_output.txt
==7096== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   74.71%  560.72us         2  280.36us  275.16us  285.56us  [CUDA memcpy HtoD]
                   22.03%  165.37us         1  165.37us  165.37us  165.37us  [CUDA memcpy DtoH]
                    3.26%  24.447us         1  24.447us  24.447us  24.447us  MatAdd(float*, float*, float*, int, int)
      API calls:   96.57%  95.538ms         3  31.846ms  70.108us  95.396ms  cudaMalloc
                    2.69%  2.6610ms         3  887.00us  479.46us  1.6884ms  cudaMemcpy
                    0.35%  346.00us         3  115.33us  109.09us  122.52us  cudaFree
                    0.20%  197.41us         1  197.41us  197.41us  197.41us  cudaLaunchKernel
                    0.17%  164.87us       114  1.4460us     139ns  68.728us  cuDeviceGetAttribute
                    0.01%  10.640us         1  10.640us  10.640us  10.640us  cuDeviceGetName
                    0.01%  6.2190us         1  6.2190us  6.2190us  6.2190us  cuDeviceGetPCIBusId
                    0.00%  4.2970us         1  4.2970us  4.2970us  4.2970us  cuDeviceTotalMem
                    0.00%  2.0610us         3     687ns     210ns  1.5580us  cuDeviceGetCount
                    0.00%     944ns         2     472ns     169ns     775ns  cuDeviceGet
                    0.00%     608ns         1     608ns     608ns     608ns  cuModuleGetLoadingMode
                    0.00%     268ns         1     268ns     268ns     268ns  cuDeviceGetUuid
```

Kernel 2

```
[19] !nvprof ./k2.out "q1_500_1000.txt" "k2_500_1000_output.txt"

==7111== NVPROF is profiling process 7111, command: ./k2.out q1_500_1000.txt k2_500_1000_output.txt
==7111== Profiling application: ./k2.out q1_500_1000.txt k2_500_1000_output.txt
==7111== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   59.36%  573.08us         2  286.54us  283.10us  289.98us  [CUDA memcpy HtoD]
                   22.66%  218.75us         1  218.75us  218.75us  218.75us  MatAdd(float*, float*, float*, int, int)
                   17.98%  173.63us         1  173.63us  173.63us  173.63us  [CUDA memcpy DtoH]
      API calls:   95.13%  70.792ms         3  23.597ms  89.689us  70.608ms  cudaMalloc
                    3.82%  2.8398ms         3  946.59us  490.60us  1.8382ms  cudaMemcpy
                    0.47%  351.51us         3  117.17us  114.35us  120.21us  cudaFree
                    0.37%  272.77us         1  272.77us  272.77us  272.77us  cudaLaunchKernel
                    0.18%  136.59us       114  1.1980us     139ns  55.111us  cuDeviceGetAttribute
                    0.02%  12.311us         1  12.311us  12.311us  12.311us  cuDeviceGetName
                    0.01%  5.9110us         1  5.9110us  5.9110us  5.9110us  cuDeviceGetPCIBusId
                    0.01%  4.4250us         1  4.4250us  4.4250us  4.4250us  cuDeviceTotalMem
                    0.00%  1.8220us         3     607ns     215ns  1.3300us  cuDeviceGetCount
                    0.00%  1.0000us         2     500ns     167ns     833ns  cuDeviceGet
                    0.00%     523ns         1     523ns     523ns     523ns  cuModuleGetLoadingMode
                    0.00%     218ns         1     218ns     218ns     218ns  cuDeviceGetUuid
```

Kernel 3

```
!nvprof ./k3.out "q1_500_1000.txt" "k3_500_1000_output.txt"
```

```
==7126== NVPROF is profiling process 7126, command: ./k3.out q1_500_1000.txt k3_500_1000_output.txt
==7126== Profiling application: ./k3.out q1_500_1000.txt k3_500_1000_output.txt
==7126== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   57.13%  578.90us         2  289.45us  278.78us  300.12us  [CUDA memcpy HtoD]
                   26.31%  266.59us         1  266.59us  266.59us  266.59us  MatAdd(float*, float*, float*, int, int)
                   16.56%  167.77us         1  167.77us  167.77us  167.77us  [CUDA memcpy DtoH]
      API calls:   95.48%  73.249ms         3  24.416ms  79.675us  73.081ms  cudaMalloc
                    3.60%  2.7634ms         3  921.12us  445.70us  1.7992ms  cudaMemcpy
                    0.44%  337.29us         3  112.43us  106.60us  115.37us  cudaFree
                    0.27%  203.45us         1  203.45us  203.45us  203.45us  cudaLaunchKernel
                    0.17%  134.22us       114  1.1770us     145ns  53.054us  cuDeviceGetAttribute
                    0.02%  11.804us         1  11.804us  11.804us  11.804us  cuDeviceGetName
                    0.01%  5.5510us         1  5.5510us  5.5510us  5.5510us  cuDeviceGetPCIBusId
                    0.01%  4.5120us         1  4.5120us  4.5120us  4.5120us  cuDeviceTotalMem
                    0.00%  2.0850us         3     695ns     232ns  1.5580us  cuDeviceGetCount
                    0.00%     838ns         2     419ns     185ns     653ns  cuDeviceGet
                    0.00%     578ns         1     578ns     578ns     578ns  cuModuleGetLoadingMode
                    0.00%     223ns         1     223ns     223ns     223ns  cuDeviceGetUuid
```

Observation 2

Kernel 1 achieves the highest speed (24.477 microseconds) for MatAdd.
Kernel 2 comes 2nd with (218.75 microseconds) for MatAdd.
Kernel 3 comes 3rd with (266.59 microseconds) for MatAdd.

Conclusion 2

Kernel 1 is the fastest due to its utilization of maximum parallelism, with each thread dedicated to computing a single element
Kernel 2 is faster than **Kernel 3** as the operation of fetching 1 row per thread is faster than fetching 1 column as explained in conclusion 1, but the difference in time is not significant, since having more columns leads to allocation more blocks hence more total threads for kernel 3, each having less loop iterations than kernel 2, which compensates the speed difference by a small margin.

Input of size 100 x 150 (Rows<Columns)

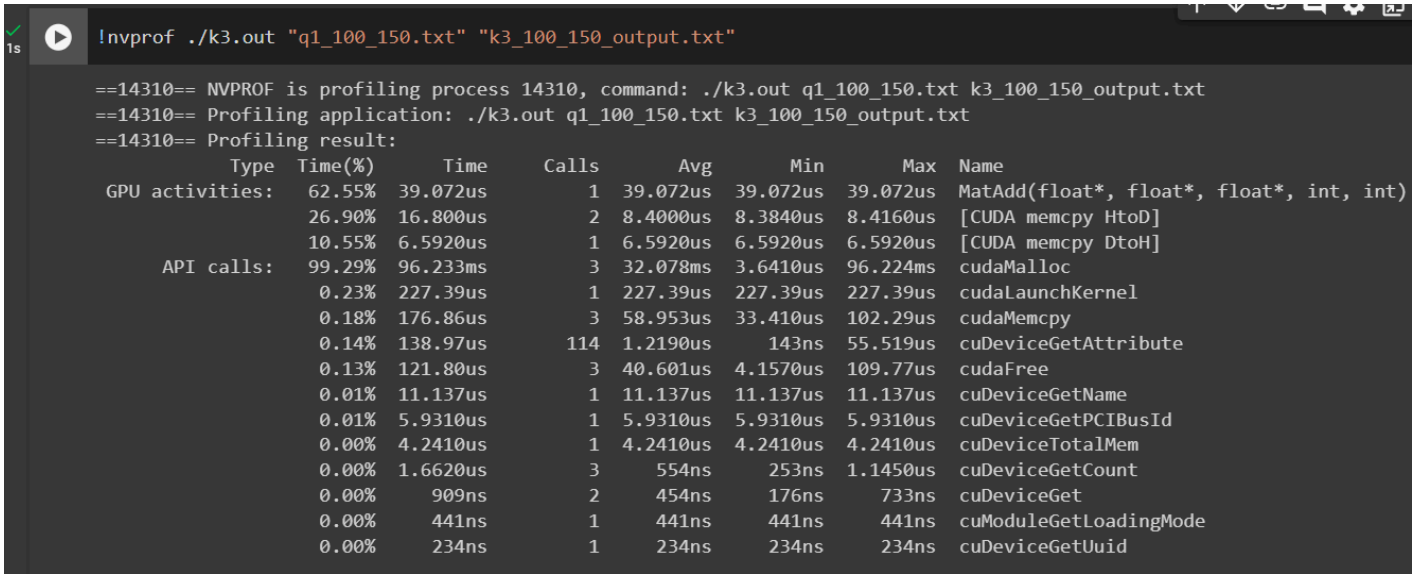
Kernel 1

```
[27] !nvprof ./k1.out "q1_100_150.txt" "k1_100_150_output.txt"

==14260== NVPROF is profiling process 14260, command: ./k1.out q1_100_150.txt k1_100_150_output.txt
==14260== Profiling application: ./k1.out q1_100_150.txt k1_100_150_output.txt
==14260== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   62.66%  17.344us         2  8.6720us  8.4160us  8.9280us  [CUDA memcpy HtoD]
                   23.82%  6.5920us         1  6.5920us  6.5920us  6.5920us  [CUDA memcpy DtoH]
                   13.53%  3.7440us         1  3.7440us  3.7440us  3.7440us  MatAdd(float*, float*, float*, int, int)
      API calls:   99.39%  96.405ms         3  32.135ms  2.9220us  96.397ms  cudaMalloc
                    0.18%  177.50us         1  177.50us  177.50us  177.50us  cudaLaunchKernel
                    0.16%  155.49us         3  51.831us  39.923us  69.637us  cudaMemcpy
                    0.14%  137.04us       114  1.2020us     143ns  54.240us  cuDeviceGetAttribute
                    0.10%  101.23us         3  33.742us  3.5350us  89.812us  cudaFree
                    0.01%  10.775us         1  10.775us  10.775us  10.775us  cuDeviceGetName
                    0.01%  5.3640us         1  5.3640us  5.3640us  5.3640us  cuDeviceGetPCIBusId
                    0.00%  4.4170us         1  4.4170us  4.4170us  4.4170us  cuDeviceTotalMem
                    0.00%  1.8700us         3     623ns     205ns  1.4260us  cuDeviceGetCount
                    0.00%     908ns         2     454ns     182ns     726ns  cuDeviceGet
                    0.00%     497ns         1     497ns     497ns     497ns  cuModuleGetLoadingMode
                    0.00%     225ns         1     225ns     225ns     225ns  cuDeviceGetUuid
```

Kernel 2

```
[28] !nvprof ./k2.out "q1_100_150.txt" "k2_100_150_output.txt"

==14285== NVPROF is profiling process 14285, command: ./k2.out q1_100_150.txt k2_100_150_output.txt
==14285== Profiling application: ./k2.out q1_100_150.txt k2_100_150_output.txt
==14285== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   60.90%  42.207us         1  42.207us  42.207us  42.207us  MatAdd(float*, float*, float*, int, int)
                   29.64%  20.544us         2  10.272us  10.080us  10.464us  [CUDA memcpy HtoD]
                    9.46%  6.5600us         1  6.5600us  6.5600us  6.5600us  [CUDA memcpy DtoH]
      API calls:   99.28%  96.083ms         3  32.028ms  3.5650us  96.074ms  cudaMalloc
                    0.24%  233.43us         1  233.43us  233.43us  233.43us  cudaLaunchKernel
                    0.20%  198.28us         3  66.093us  41.828us  106.14us  cudaMemcpy
                    0.14%  138.28us       114  1.2130us     148ns  54.232us  cuDeviceGetAttribute
                    0.11%  103.32us         3  34.440us  3.4900us  92.662us  cudaFree
                    0.01%  11.735us         1  11.735us  11.735us  11.735us  cuDeviceGetName
                    0.01%  5.3140us         1  5.3140us  5.3140us  5.3140us  cuDeviceGetPCIBusId
                    0.00%  4.5950us         1  4.5950us  4.5950us  4.5950us  cuDeviceTotalMem
                    0.00%  1.9130us         3     637ns     226ns  1.3800us  cuDeviceGetCount
                    0.00%     741ns         2     370ns     179ns     562ns  cuDeviceGet
                    0.00%     560ns         1     560ns     560ns     560ns  cuModuleGetLoadingMode
                    0.00%     207ns         1     207ns     207ns     207ns  cuDeviceGetUuid
```

Kernel 3



```
!nvprof ./k3.out "q1_100_150.txt" "k3_100_150_output.txt"

==14310== NVPROF is profiling process 14310, command: ./k3.out q1_100_150.txt k3_100_150_output.txt
==14310== Profiling application: ./k3.out q1_100_150.txt k3_100_150_output.txt
==14310== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   62.55%  39.072us         1  39.072us  39.072us  39.072us  MatAdd(float*, float*, float*, int, int)
                   26.90%  16.800us         2  8.4000us  8.3840us  8.4160us  [CUDA memcpy HtoD]
                   10.55%  6.5920us         1  6.5920us  6.5920us  6.5920us  [CUDA memcpy DtoH]
      API calls:   99.29%  96.233ms         3  32.078ms  3.6410us  96.224ms  cudaMalloc
                    0.23%  227.39us         1  227.39us  227.39us  227.39us  cudaLaunchKernel
                    0.18%  176.86us         3  58.953us  33.410us  102.29us  cudaMemcpy
                    0.14%  138.97us       114  1.2190us     143ns  55.519us  cuDeviceGetAttribute
                    0.13%  121.80us         3  40.601us  4.1570us  109.77us  cudaFree
                    0.01%  11.137us         1  11.137us  11.137us  11.137us  cuDeviceGetName
                    0.01%  5.9310us         1  5.9310us  5.9310us  5.9310us  cuDeviceGetPCIBusId
                    0.00%  4.2410us         1  4.2410us  4.2410us  4.2410us  cuDeviceTotalMem
                    0.00%  1.6620us         3     554ns     253ns  1.1450us  cuDeviceGetCount
                    0.00%     909ns         2     454ns     176ns     733ns  cuDeviceGet
                    0.00%     441ns         1     441ns     441ns     441ns  cuModuleGetLoadingMode
                    0.00%     234ns         1     234ns     234ns     234ns  cuDeviceGetUuid
```

Observation 3

Kernel 1 achieves the highest speed (3.744 microseconds) for MatAdd.
Kernel 3 comes 2nd with (39.072 microseconds) for MatAdd
Kernel 2 comes 3rd with (42.207 microseconds) for MatAdd.

Conclusion 3

Kernel 1 is the fastest due to its utilization of maximum parallelism, with each thread dedicated to computing a single element
Kernel 3 is faster than **Kernel 2** since having more columns leads to allocation more blocks hence more total threads for kernel 3, each having less loop iterations than kernel 2, which compensates for the speed difference between kernel 2 and 3 for this input size.

Input of size 1000 x 500 (Rows>Columns)

Kernel 1

```
!nvprof ./k1.out "q1_1000_500.txt" "k1_1000_500_output.txt"
```

```
==7185== NVPROF is profiling process 7185, command: ./k1.out q1_1000_500.txt k1_1000_500_output.txt
==7185== Profiling application: ./k1.out q1_1000_500.txt k1_1000_500_output.txt
==7185== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   74.81%  564.05us         2  282.03us  275.13us  288.92us  [CUDA memcpy HtoD]
                   21.85%  164.70us         1  164.70us  164.70us  164.70us  [CUDA memcpy DtoH]
                    3.34%  25.183us         1  25.183us  25.183us  25.183us  MatAdd(float*, float*, float*, int, int)
      API calls:   95.72%  73.626ms         3  24.542ms  74.087us  73.476ms  cudaMalloc
                    3.31%  2.5425ms         3  847.48us  453.50us  1.5974ms  cudaMemcpy
                    0.46%  354.36us         3  118.12us  104.10us  133.29us  cudaFree
                    0.24%  186.91us       114  1.6390us     175ns  72.997us  cuDeviceGetAttribute
                    0.24%  182.06us         1  182.06us  182.06us  182.06us  cudaLaunchKernel
                    0.02%  13.475us         1  13.475us  13.475us  13.475us  cuDeviceGetName
                    0.01%  6.0580us         1  6.0580us  6.0580us  6.0580us  cuDeviceTotalMem
                    0.01%  5.7530us         1  5.7530us  5.7530us  5.7530us  cuDeviceGetPCIBusId
                    0.00%  2.8390us         3     946ns     315ns  2.1720us  cuDeviceGetCount
                    0.00%     905ns         2     452ns     256ns     649ns  cuDeviceGet
                    0.00%     679ns         1     679ns     679ns     679ns  cuModuleGetLoadingMode
                    0.00%     527ns         1     527ns     527ns     527ns  cuDeviceGetUuid
```

Kernel 2

```
[21] !nvprof ./k2.out "q1_1000_500.txt" "k2_1000_500_output.txt"
```

```
==7155== NVPROF is profiling process 7155, command: ./k2.out q1_1000_500.txt k2_1000_500_output.txt
==7155== Profiling application: ./k2.out q1_1000_500.txt k2_1000_500_output.txt
==7155== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   63.85%  569.78us         2  284.89us  283.64us  286.14us  [CUDA memcpy HtoD]
                   18.53%  165.40us         1  165.40us  165.40us  165.40us  MatAdd(float*, float*, float*, int, int)
                   17.62%  157.24us         1  157.24us  157.24us  157.24us  [CUDA memcpy DtoH]
      API calls:   96.45%  92.332ms         3  30.777ms  71.268us  92.186ms  cudaMalloc
                    2.76%  2.6446ms         3  881.54us  465.15us  1.6726ms  cudaMemcpy
                    0.37%  355.91us         3  118.64us  113.24us  127.40us  cudaFree
                    0.22%  214.45us         1  214.45us  214.45us  214.45us  cudaLaunchKernel
                    0.16%  155.85us       114  1.3670us     163ns  62.311us  cuDeviceGetAttribute
                    0.01%  12.661us         1  12.661us  12.661us  12.661us  cuDeviceGetName
                    0.01%  6.0090us         1  6.0090us  6.0090us  6.0090us  cuDeviceGetPCIBusId
                    0.00%  4.6430us         1  4.6430us  4.6430us  4.6430us  cuDeviceTotalMem
                    0.00%  2.8460us         2  1.4230us     212ns  2.6340us  cuDeviceGet
                    0.00%  2.1940us         3     731ns     268ns  1.5750us  cuDeviceGetCount
                    0.00%     537ns         1     537ns     537ns     537ns  cuModuleGetLoadingMode
                    0.00%     266ns         1     266ns     266ns     266ns  cuDeviceGetUuid
```

Kernel 3

```
[22] !nvprof ./k3.out "q1_1000_500.txt" "k3_1000_500_output.txt"

==7170== NVPROF is profiling process 7170, command: ./k3.out q1_1000_500.txt k3_1000_500_output.txt
==7170== Profiling application: ./k3.out q1_1000_500.txt k3_1000_500_output.txt
==7170== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   45.15%  567.60us         2  283.80us  277.63us  289.98us  [CUDA memcpy HtoD]
                   41.61%  523.06us         1  523.06us  523.06us  523.06us  MatAdd(float*, float*, float*, int, int)
                   13.24%  166.40us         1  166.40us  166.40us  166.40us  [CUDA memcpy DtoH]
      API calls:   95.04%  71.581ms         3  23.860ms  81.352us  71.409ms  cudaMalloc
                    4.00%  3.0094ms         3  1.0031ms  461.36us  2.0363ms  cudaMemcpy
                    0.47%  353.44us         3  117.81us  115.88us  120.66us  cudaFree
                    0.28%  208.98us         1  208.98us  208.98us  208.98us  cudaLaunchKernel
                    0.18%  137.02us       114  1.2010us     144ns  55.261us  cuDeviceGetAttribute
                    0.02%  11.780us         1  11.780us  11.780us  11.780us  cuDeviceGetName
                    0.01%  5.3590us         1  5.3590us  5.3590us  5.3590us  cuDeviceGetPCIBusId
                    0.01%  4.7190us         1  4.7190us  4.7190us  4.7190us  cuDeviceTotalMem
                    0.00%  1.7140us         3     571ns     189ns  1.2330us  cuDeviceGetCount
                    0.00%  1.2460us         2     623ns     196ns  1.0500us  cuDeviceGet
                    0.00%     513ns         1     513ns     513ns     513ns  cuModuleGetLoadingMode
                    0.00%     241ns         1     241ns     241ns     241ns  cuDeviceGetUuid
```

Observation 4

Kernel 1 achieves the highest speed (25.183 microseconds) for MatAdd.
Kernel 2 comes 2nd with (165.40 microseconds) for MatAdd.
Kernel 3 comes 3rd with (523.06 microseconds) for MatAdd.

Conclusion 4

Kernel 1 is the fastest due to its utilization of maximum parallelism, with each thread dedicated to computing a single element
Kernel 2 is faster than **Kernel 3** as the operation of fetching 1 row per thread is faster than fetching 1 column as explained in conclusion 1, also the difference in time is significant, since having more rows leads to allocation more blocks hence more total threads for kernel 2, each having less loop iterations than kernel 3, which makes kernel 3 even slower.