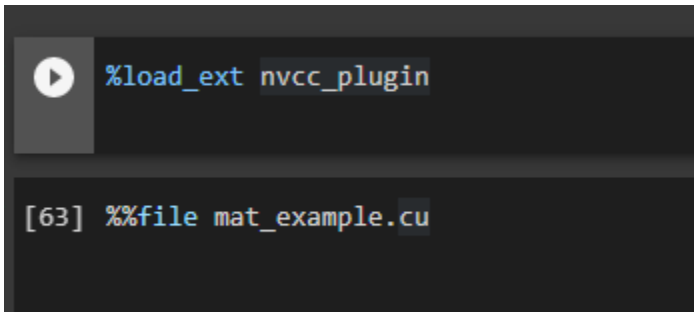
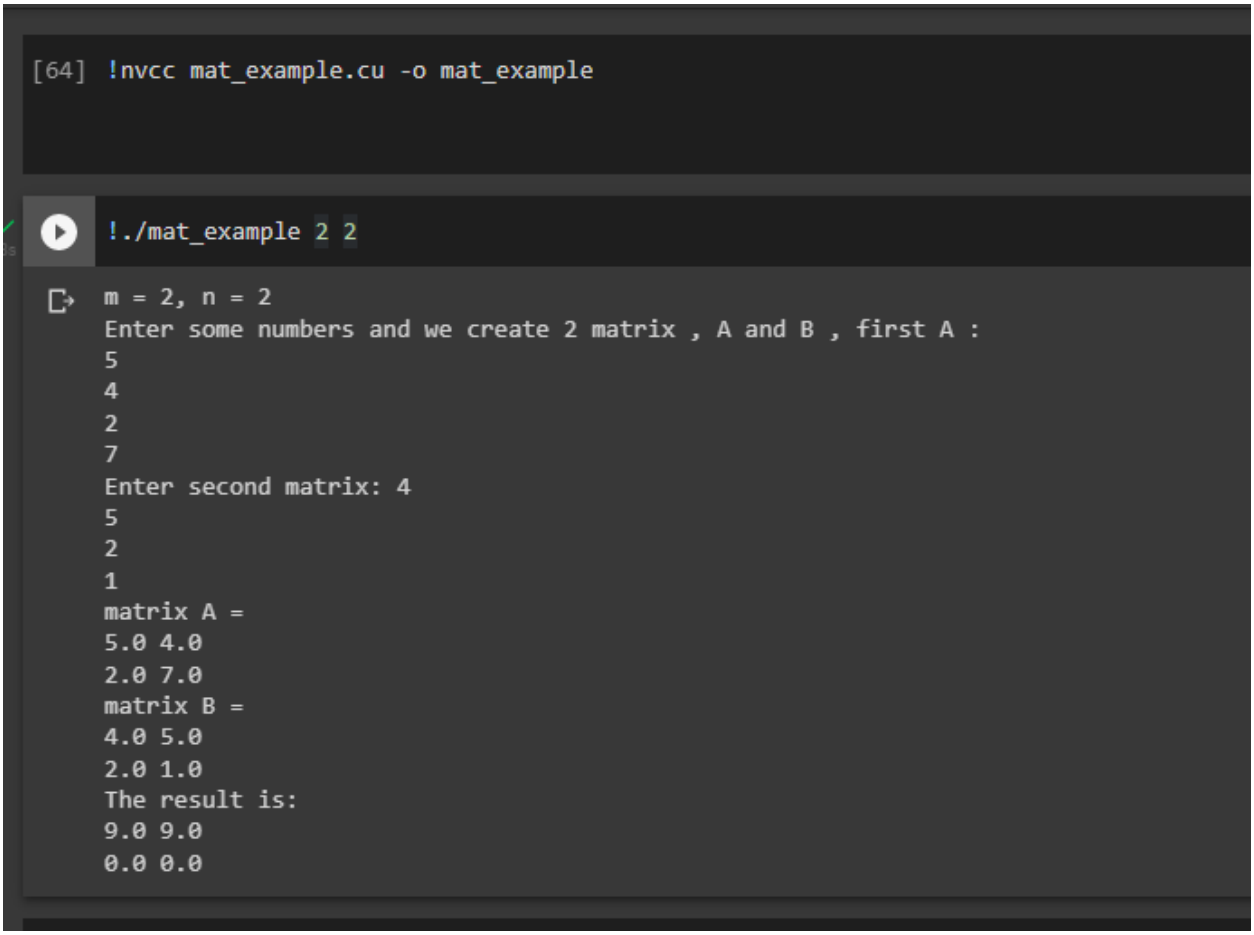
علی عسگری
40032223

به کمک google research colab پروفایلینگ صورت گرفت و مراحل آن پس از نصب کودا و کانفیگ های ابتدایی به شکل زیر بود.



با استفاده از دو خط بالا میتوانیم یک فایل کودا تحت عنوان mat_example ایجاد کنیم ، این فایل معادل تلاش اول و همان matrix_sum میباشد.

حال با استفاده از nvcc فایل را کامپایل و یک فایل اجرایی برای آن ایجاد میکنیم. خروجی به شکل زیر است.



پروفایلینگ برای تلاش اول: کدی که حاوی قطعه کد زیر برای ایجاد بلاک های چند گانه است.

```
/* Invoke kernel using m thread blocks, each of    */
/* which contains n threads                        */
dim3 block_size( 16, 16 );
dim3 num_blocks( ( n - 1 + block_size.x ) / block_size.x,
                 ( m - 1 + block_size.y ) / block_size.y );

matrix_sum<<<block_size, num_blocks>>>(d_A, d_B, d_C, m, n);
```

پروفایلینگ تلاش اول ، matrix_sum که در اینجا `mat_example` تعریف شده است.
یک ماتریس 2*2 ایجاد میشود.

```
de  + Text
6
matrix A =
3.0 3.0
4.0 3.0
matrix B =
2.0 4.0
5.0 6.0
==18345== NVPROF is profiling process 18345, command: ./mat_example 2 2
The result is:
5.0 7.0
0.0 0.0
==18345== Profiling application: ./mat_example 2 2
==18345== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   38.96%  3.8400us         2  1.9200us  1.5680us  2.2720us  [CUDA memcpy HtoD]
                   35.06%  3.4560us         1  3.4560us  3.4560us  3.4560us  matrix_sum(float*, float*, float*, int, int)
                   25.97%  2.5600us         1  2.5600us  2.5600us  2.5600us  [CUDA memcpy DtoH]
      API calls:   99.56%  278.19ms         3  92.730ms  2.7840us  278.18ms  cudaMalloc
                    0.19%  531.52us         1  531.52us  531.52us  531.52us  cuDeviceTotalMem
                    0.08%  233.51us        96  2.4320us     120ns  94.774us  cuDeviceGetAttribute
                    0.05%  149.00us         1  149.00us  149.00us  149.00us  cudaLaunchKernel
                    0.05%  139.07us         3  46.355us  6.2560us  116.55us  cudaFree
                    0.04%  104.43us         3  34.810us  13.150us  67.318us  cudaMemcpy
                    0.02%  46.203us         1  46.203us  46.203us  46.203us  cuDeviceGetName
                    0.00%  8.9010us         1  8.9010us  8.9010us  8.9010us  cudaThreadSynchronize
                    0.00%  6.7680us         1  6.7680us  6.7680us  6.7680us  cuDeviceGetPCIBusId
                    0.00%  1.9550us         3     651ns     151ns     968ns  cuDeviceGetCount
                    0.00%  1.8480us         2     924ns     329ns  1.5190us  cuDeviceGet
```

در ادامه به سراغ تلاش دوم ، قطعه کد با نام mat_sum که در اینجا mat_2 نام گذاری میشود.
قطعه کد زیر شیوه فراخوانی کرنل GPU در این فایل را نشان میدهد که خبری از ایجاد بلاک های چندگانه و محاسبه تعدادشان نیست و از ابعاد ماتریس ها استفاده شده است.

```
/* Invoke kernel using m thread blocks, each of    */
/* which contains n threads                        */
Mat_sum<<<m, n>>>(d_A, d_B, d_C, m, n);
```

پروفایلینگ برای تلاش دوم :

+ Code + Text

```
matrix A =
2.0 3.0
3.0 2.0
matrix B =
3.0 2.0
3.0 2.0
==18457== NVPROF is profiling process 18457, command: ./mat_2 2 2
The sum is:
5.0 5.0
6.0 4.0
==18457== Profiling application: ./mat_2 2 2
==18457== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   43.12%  4.4160us         2  2.2080us  2.0800us  2.3360us  [CUDA memcpy HtoD]
                   30.62%  3.1360us         1  3.1360us  3.1360us  3.1360us  Mat_sum(float*, float*, float*, int, int)
                   26.25%  2.6880us         1  2.6880us  2.6880us  2.6880us  [CUDA memcpy DtoH]
      API calls:   99.41%  195.95ms         3  65.317ms  2.6650us  195.94ms  cudaMalloc
                    0.24%  465.29us         1  465.29us  465.29us  465.29us  cuDeviceTotalMem
                    0.12%  235.30us         1  235.30us  235.30us  235.30us  cudaLaunchKernel
                    0.10%  194.65us        96  2.0270us     124ns  88.250us  cuDeviceGetAttribute
                    0.08%  150.67us         3  50.224us  6.3800us  121.06us  cudaFree
                    0.03%  64.330us         3  21.443us  12.545us  26.229us  cudaMemcpy
                    0.01%  26.477us         1  26.477us  26.477us  26.477us  cuDeviceGetName
                    0.00%  7.5460us         1  7.5460us  7.5460us  7.5460us  cudaThreadSynchronize
                    0.00%  6.6470us         1  6.6470us  6.6470us  6.6470us  cuDeviceGetPCIBusId
                    0.00%  1.8700us         3     623ns     139ns     908ns  cuDeviceGetCount
                    0.00%  1.5840us         2     792ns     312ns  1.2720us  cuDeviceGet
```

لازم به ذکر است که برای پروفایلینگ از دستوری مشابه دستور زیر استفاده شد.

```
nvprof ./mat_example 2 2
```