# M2-BIG DATA
# GPGPU - Chapter 4

## Exercice 2


Réalisé par:
Gaby Maroun

Encadré par:
Dr. Etancelin JM

December 21, 2020

## Objectives

Implement a basic dense matrix multiplication kernel with a tiled algorithm using device's shared memory.

## Instructions

Create a second version of your program from exercise 1 and adjust to perform a tiled algorithm with shared memory. In this exercice, we use a tile size of 16x16.

The code can be found in the attached file under the name "2-tiledMatMul2.cu"

1. **How many floating operations are being performed in your matrix multiply kernel ? explain.**

   There are

$$2 * nACols * \frac{nACols}{TILE\_WIDTH}$$
$$=> 2 * \frac{nACols^2}{TILE\_WIDTH}$$

   floating operations being performed as the for loop breaks out when reaching the number of A columns.
   *And it represents the number of times the kernel multiply ds_A and ds_B and add the resulted product to the pValue.*

2. **How many global memory reads are being performed by your kernel ? explain.**
   *There are*

$$\frac{nACols}{TILE\_WIDTH} * (nARows * nACols + nBRows * nBCols + 2 * nACols)$$
$$=> \frac{nACols}{TILE\_WIDTH} * (nARows * nACols + nACols * nBCols + 2 * nACols)$$
$$=> \frac{nACols^2}{TILE\_WIDTH} * (nARows + nBCols + 2)$$

   *reads from A and B then from ds_A and ds_B are being performed by the kernel*

3. **How many global memory writes are being performed by your kernel ? EXPLAIN.**
   *There are*

$$\frac{nACols}{TILE\_WIDTH} * (nARows * nACols + nBRows * nBCols) + nARows * nBCols$$

   *writes in ds_A, ds_B and C that are being performed by the kernel.*

4. **Compute the arithmetic intensity of your kernel. The arithmetic intensity is a FLOP/Byte number standing for the number of floating point operations performed per byte of global memory access**

$$\frac{1}{4} * \frac{2 * \frac{nACols^2}{TILE\_WIDTH}}{\frac{nACols^2}{TILE\_WIDTH} * (nARows + nBCols + 2) + \frac{nACols}{TILE\_WIDTH} * (nARows * nACols + nBRows * nBCols) + nARows * nBCols}$$

$$=> \frac{1}{4} * \frac{2 * nACols}{nACols * (nARows + nBCols + 2) + (nARows * nACols + nBRows * nBCols) + nARows * nBCols}$$

$$=> \frac{1}{4} * \frac{2 * nACols}{nACols * nARows + nBRows * nBCols + 2 * nACols + nARows * nACols + nBRows * nBCols + nARows * nBCols}$$

$$=> \frac{1}{4} * \frac{2 * nACols}{2 * nACols * nARows + 2 * nBRows * nBCols + 2 * nACols + nARows * nBCols}$$

$$=> \frac{1}{4} * \frac{nACols}{nACols * nARows + nBRows * nBCols + nACols + \frac{nARows * nBCols}{2}}$$

The 1/4 is used to convert the denominator to Bytes.

5. **Compare with the arithmetic intensity of the basicMatMul kernel. Explain why, at same matrices dimensions, the tiled version is better than the basic one. Get the kernels execution time using NVIDIA profiler nvprof.**

The arithmetic intensity of the basicMatMul kernel was equal to :

$$\frac{1}{4} * \frac{2 * nACols}{2 * nACols + 1} = \frac{1}{4} * \frac{nACols}{nACols * nARows + nBRows * nBCols + nACols + \frac{nARows * nBCols}{2}}$$

$$=> \frac{nACols}{2 * nACols + 1} = \frac{nACols}{2 * (nACols * nARows + nBRows * nBCols + nACols + \frac{nARows * nBCols}{2})}$$

Having, $2 * nACols + 1 < 2 * (nACols * nARows + nBRows * nBCols + nACols + \frac{nARows * nBCols}{2})$

which assures us that basicMatMul is more intense than the tiledMatMul kernel *The kernel execution time of the basic matrix multiplication exercise is as follows,*

```
Matrix multiplication dimensions: [300;500] = [300;400] x [400;500]
==3602== NVPROF is profiling process 3602, command: ./1-basicMatMul 300 400 500
Ok
==3602== Profiling application: ./1-basicMatMul 300 400 500
==3602== Profiling result:
   Start  Duration            Grid Size      Block Size     Regs*    SSMem*    DSMem*      Size  Throughput  SrcMemType  DstMemType           Device   Context    Stream
  Name
324.19ms  40.063us                    -               -         -         -         -  468.75KB  11.158GB/s    Pageable      Device  Quadro RTX 4000         1         7
  [CUDA memcpy HtoD]
324.31ms  70.687us                    -               -         -         -         -  781.25KB  10.540GB/s    Pageable      Device  Quadro RTX 4000         1         7
  [CUDA memcpy HtoD]
324.40ms  1.3120us                    -               -         -         -         -  585.94KB  425.91GB/s      Device           -  Quadro RTX 4000         1         7
  [CUDA memset]
326.22ms  257.18us           (16 10 1)       (32 32 1)        48        0B        0B         -           -           -           -  Quadro RTX 4000         1         7
  dgemm(float*, float*, float*, int, int, int, int) [113]
326.48ms  48.287us                    -               -         -         -         -  585.94KB  11.572GB/s      Device    Pageable  Quadro RTX 4000         1         7
  [CUDA memcpy DtoH]

Regs: Number of registers used per CUDA thread. This number includes registers used internally by the CUDA driver and/or tools and can be more than what the compiler sh
ows.
SSMem: Static shared memory allocated per CUDA block.
DSMem: Dynamic shared memory allocated per CUDA block.
SrcMemType: The type of source memory accessed by memory operation/copy
```

*Meanwhile, the amount of time for the tiled matrix multiplication is,*

```
gmaroun@scinfe058:/import/etud/3/gmaroun/Bureau/stockage/Semestre 3/GPGPU/Chap4/Ex2$ nvprof --print-gpu-trace ./2-tiledMatMul2 300 400 500
Matrix multiplication dimensions: [300;500] = [300;400] x [400;500]
==4707== NVPROF is profiling process 4707, command: ./2-tiledMatMul2 300 400 500
Ok
==4707== Profiling application: ./2-tiledMatMul2 300 400 500
==4707== Profiling result:
   Start  Duration            Grid Size      Block Size     Regs*    SSMem*    DSMem*      Size  Throughput  SrcMemType  DstMemType           Device   Context    Stream
  Name
278.78ms  39.968us                    -               -         -         -         -  468.75KB  11.185GB/s    Pageable      Device  Quadro RTX 4000         1         7
  [CUDA memcpy HtoD]
278.89ms  65.087us                    -               -         -         -         -  781.25KB  11.447GB/s    Pageable      Device  Quadro RTX 4000         1         7
  [CUDA memcpy HtoD]
278.97ms  1.3120us                    -               -         -         -         -  585.94KB  425.91GB/s      Device           -  Quadro RTX 4000         1         7
  [CUDA memset]
281.07ms  281.56us           (32 19 1)       (16 16 1)        20  2.0000KB        0B         -           -           -           -  Quadro RTX 4000         1         7
  dgemm(float*, float*, float*, int, int, int, int) [113]
281.36ms  48.127us                    -               -         -         -         -  585.94KB  11.611GB/s      Device    Pageable  Quadro RTX 4000         1         7
  [CUDA memcpy DtoH]

Regs: Number of registers used per CUDA thread. This number includes registers used internally by the CUDA driver and/or tools and can be more than what the compiler sh
ows.
SSMem: Static shared memory allocated per CUDA block.
DSMem: Dynamic shared memory allocated per CUDA block.
SrcMemType: The type of source memory accessed by memory operation/copy
DstMemType: The type of destination memory accessed by memory operation/copy
```

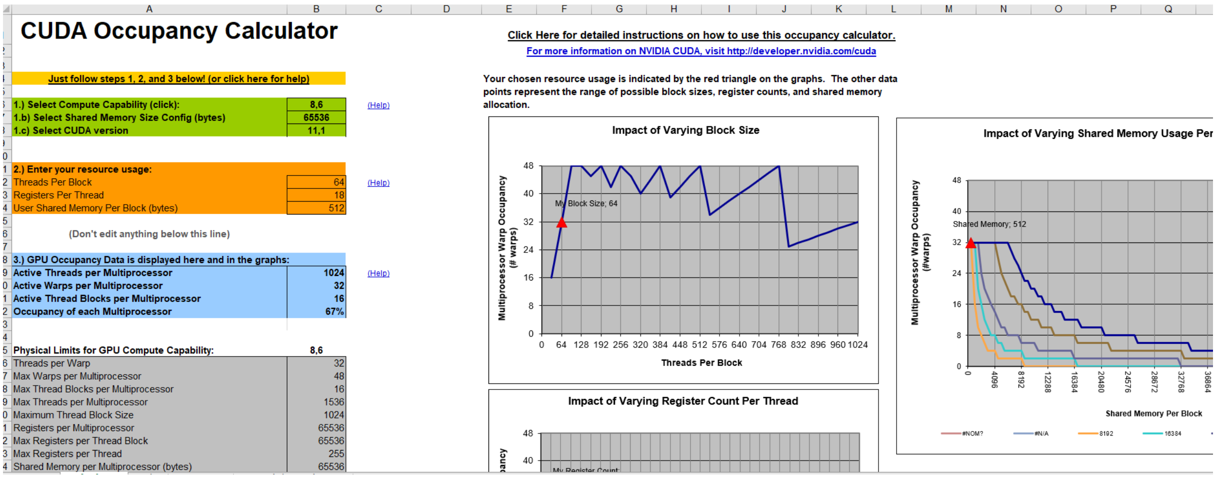
that is about 50ms faster.

6. **Use the occupancy calculator to compute the occupancy for TILE_SIZE equals to 8, 16 and 32. Which size gives the best computational time ?**

*For TILE_WIDTH=8*

*For Occupancy*=67%

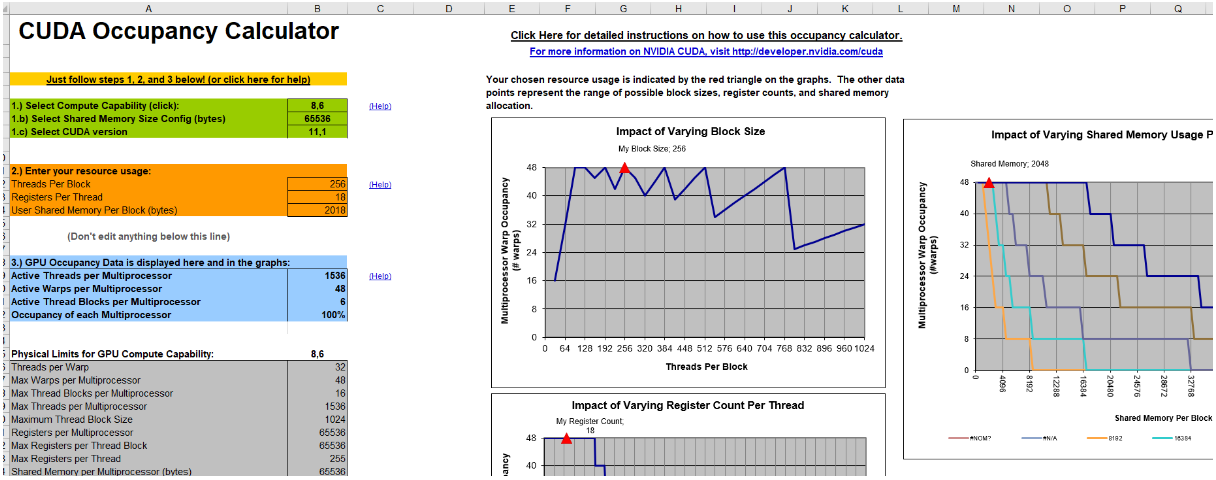
```
mbedere@scinfe051:~/Bureau/gpgpu/Chapitre 4 exercice 2$ nvcc --ptxas-options=-v 1-basicMatMu
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z5dgemmPfS_S_iiii' for 'sm_30'
ptxas info    : Function properties for _Z5dgemmPfS_S_iiii
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 18 registers, 512 bytes smem, 360 bytes cmem[0]
```




*For TILE_WIDTH=16*

*For Occupancy*=100%

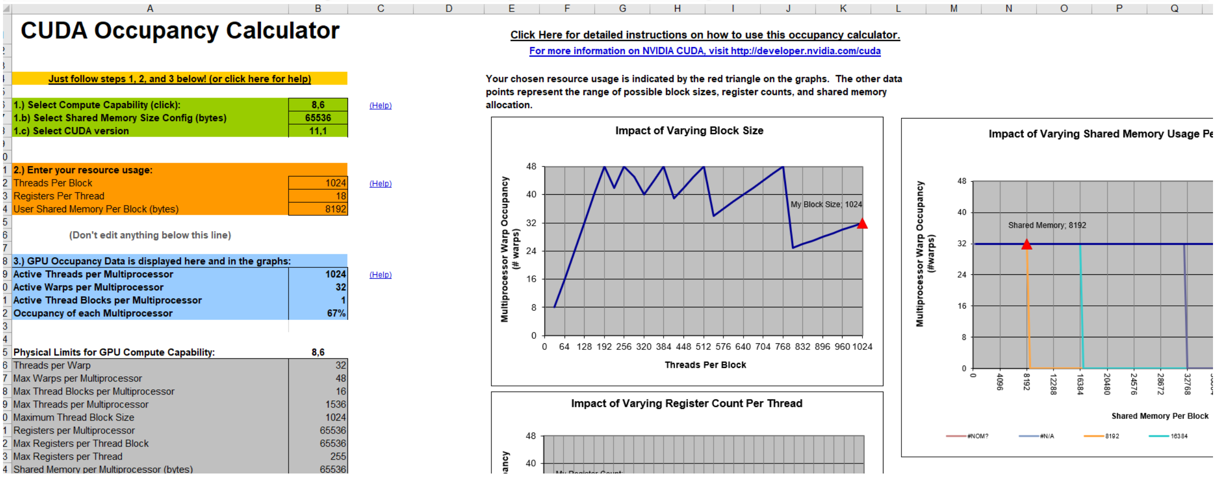
```
mbedere@scinfe051:~/Bureau/gpgpu/Chapitre 4 exercice 2$ nvcc --ptxas-options=-v
nvcc fatal   : No input files specified; use option --help for more information
mbedere@scinfe051:~/Bureau/gpgpu/Chapitre 4 exercice 2$ nvcc --ptxas-options=-v 1-basicMatMul.cu
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z5dgemmPfS_S_iiii' for 'sm_30'
ptxas info    : Function properties for _Z5dgemmPfS_S_iiii
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 18 registers, 2048 bytes smem, 360 bytes cmem[0]
```




*For TILE_WIDTH=32*

*For Occupancy=*$67\%$

```
mbedere@scinfe051:~/Bureau/gpgpu/Chapitre 4 exercice 2$ nvcc --ptxas-options=-v 1-basicMatMul.cu
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z5dgemmPfS_S_iiii' for 'sm_30'
ptxas info    : Function properties for _Z5dgemmPfS_S_iiii
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 18 registers, 8192 bytes smem, 360 bytes cmem[0]
```




7. **For 1000 × 1000 matrices, which is the most time-consuming element in the tiled version of the matrix multiplication program ?**

```
==7938== NVPROF is profiling process 7938, command: ./1-basicMatMul 1000 20 1000
Ok
==7938== Profiling application: ./1-basicMatMul 1000 20 1000
==7938== Profiling result:
   Start  Duration            Grid Size      Block Size     Regs*    SSMem*    DSMem*      Size  Throughput  SrcMemType  DstMemType           Device   Context    Stream
  Name
333.79ms  9.2800us                    -               -         -         -         -  78.125KB  8.0286GB/s    Pageable      Device  Quadro RTX 4000         1         7
  [CUDA memcpy HtoD]
333.82ms  8.7680us                    -               -         -         -         -  78.125KB  8.4975GB/s    Pageable      Device  Quadro RTX 4000         1         7
  [CUDA memcpy HtoD]
333.84ms  1.2800us                    -               -         -         -         -  3.8147MB  2910.4GB/s      Device           -  Quadro RTX 4000         1         7
  [CUDA memset]
333.85ms  142.84us        (125 125 1)         (8 8 1)        20      512B        0B         -           -           -           -  Quadro RTX 4000         1         7
  dgemm(float*, float*, float*, int, int, int, int) [113]
334.00ms  977.28us                    -               -         -         -         -  3.8147MB  3.8119GB/s      Device    Pageable  Quadro RTX 4000         1         7
  [CUDA memcpy DtoH]

Regs: Number of registers used per CUDA thread. This number includes registers used internally by the CUDA driver and/or tools and can be more than what the compiler sh
ows.
SSMem: Static shared memory allocated per CUDA block.
DSMem: Dynamic shared memory allocated per CUDA block.
SrcMemType: The type of source memory accessed by memory operation/copy
DstMemType: The type of destination memory accessed by memory operation/copy
```

For 1000 × 1000 matrices, the most time consuming element in the tiled version of the matrix multiplication program is the dgemm function with in particular the for loop and the two if conditions.

8. **Suppose you have matrices with dimensions bigger than the max thread dimensions. Sketch an algorithm that would perform matrix multiplication algorithm that would perform the multiplication in this case.**

9. **Suppose you have matrices that would not fit in global memory. Sketch an algorithm that would perform matrix multiplication algorithm that would perform the multiplication out of place.**

NB: I would like to mention that I used some of the images from my class colleague Manon Beder having been not feeling well for the past couple of days, I didn't have time to do all the questions and deliver on time. Everything written in this report is very well understood and open to discussion.

La fin.