Tecnológico de Monterrey

Campus Santa Fe

Programación Multinúcleo

Assignment 1

By Luis Carlos Arias Camacho

Objective.

The purpose of this assignment is to develop a software that solves a matrix multiplication with 2 matrices of NxN in three different manners:

- Sequentially on CPU.
- On Parallel on CPU.
- On Parallel on GPU.

With this information we are going to be able to learn and comprehend some of the basics of multithreading in CPU and GPU.

Test Specifications

All the test were made on the server that the professor gave us access to test our programs, that is because I do not have a Laptop with an Nvidia GPU (MacBook Pro 2012).

Also in the assignment there was specified that the values for N in NxN for the different scenarios should be 1000, 2000 & 4000 but because of memory capacity it was not possible, so the test here made were done with N = 300, N = 400 and N = 500 because of memory capacity in the professor's server.

Below we will show how some test were unsuccessful with bigger values for N.

Files in the Assignment and Functionality

This assignment has 9 files, we will explain what every file does,contains and how to compile the cpp files:

1. **custom.h**
   This file just have 3 simple functions:
   - printMatrix(*Matrix, nx, ny) where *Matrix is an integer pointer to a Matrix, nx and ny are the size of N in NxN.
   - initialData(* Matrix, size) where *Matrix in an integer pointer to a Matrix to be filled and size is nx * ny (total size)
   - checkResult(*Matrix1, *Matrix2, size) where *Matrix1 and *Matrix2 are two integer pointer matrices to be compared and size is the total size of them.

2. **common.h**
   This file just defines the SAFE CALLS for the device call in GPU

3. **multMatrixOnHost.h**
   This file has the function multMatrixOnHost(*A, *B, *C, nx, ny) which multiplies A & B in C with the number of columns nx and number of rows ny in a sequential manner on CPU.

4. **multMatrixOMP.h**
   This file has the function multMatrixOMP(*A, *B, *C, nx, ny) which multiplies A & B in C with the number of columns nx and number of rows ny in a parallel manner with OpenMP on CPU.

5. **multMatrixOnGPU2d1d.h**
   This file has the function multMatrixOnGPU2d1d(*A, *B, *C, nx, ny) which multiplies A & B in C with the number of columns nx and number of rows ny in a parallel manner with GPU.

6. **matrixMult_nothreads.cpp**
   This files makes the sequential matrix multiplication with two matrices of 300x300, 400x400 and 500x500.

   Compile line:
   g++ -o **matrixMult_nothreads matrixMult_nothreads.cpp** -std=c++11 -Wall

7. **matrixMult_omp.cpp**
   This files makes the sequential and parallel matrix multiplication (using OpenMP) with two matrices of 300x300, 400x400 and 500x500. Then compares the execution times and calculates speedups.

   Compile line:
   g++ -o **matrixMult_omp matrixMult_omp.cpp** -std=c++11 -fopenmp

8. **matrixMult_GPU.cpp**
   This files makes the sequential and parallel matrix multiplication (using GPU) with two matrices of 300x300, 400x400 and 500x500. Then compares the execution times and calculates speedups.

   Compile line:
   nvcc -o **matrixMult_GPU matrixMult_GPU.cpp** -std=c++11

9. **Assignment1.cpp (This is the important one)**
   This files makes the sequential and parallel matrix multiplication (using OpenMP GPU) with two matrices of 300x300, 400x400 and 500x500. Then compares the execution times with the three methods and calculates speedups.

   Compile line:
   nvcc -o **Assignment1 Assignment1.cpp** -std=c++11 -Xcompiler -fopenmp
   *(You have to add the -Xcompiler flag so OpenMP can work on a Nvidia device with cuda code)

Testings:

Here we can see how some testings were killed by the server because it could not handle so much memory, thats why I decided to test with smaller values:

```
A01364808@alien1-lab:~/.../Assignement_1$ ./matrixMult_omp
./matrixMult_omp starting...

Matrix size: nx 200 ny 200
Calculating in CPU
Average time for 100 iterations is 22.762741 ms for a multiplication in a 200x200 matrix on Host
Calculating in OpenMP
Average time for 100 iterations is 22.762741 ms for a multiplication in a 200x200 matrix with OpenMP
Checking result between cpu and OpenMP
Arrays match.

Average time in CPU 200x200 matrix: 22.762741
Average time in OpenMO 200x200 matrix: 6.267529
Speedup: 3.631853


Matrix size: nx 500 ny 500
Calculating in CPU
Average time for 100 iterations is 374.838348 ms for a multiplication in a 500x500 matrix on Host
Calculating in OpenMP
Average time for 100 iterations is 374.838348 ms for a multiplication in a 500x500 matrix with OpenMP
Checking result between cpu and OpenMP
Arrays match.

Average time in CPU 500x500 matrix: 374.838348
Average time in OpenMO 500x500 matrix: 103.433914
Speedup: 3.623940


Matrix size: nx 800 ny 800
Calculating in CPU
Average time for 100 iterations is 1534.730957 ms for a multiplication in a 800x800 matrix on Host
Calculating in OpenMP
Killed
```

```
A01364808@alien1-lab:~/.../Assignement_1$ ./matrixMult_omp
./matrixMult_omp starting...

Matrix size: nx 350 ny 350
Calculating in CPU
Average time for 100 iterations is 123.996727 ms for a multiplication in a 350x350 matrix on Host
Calculating in OpenMP
Average time for 100 iterations is 123.996727 ms for a multiplication in a 350x350 matrix with OpenMP
Checking result between cpu and OpenMP
Arrays match.

Average time in CPU 350x350 matrix: 123.996727
Average time in OpenMO 350x350 matrix: 34.588333
Speedup: 3.584929


Matrix size: nx 500 ny 500
Calculating in CPU
Average time for 100 iterations is 374.643402 ms for a multiplication in a 500x500 matrix on Host
Calculating in OpenMP
Average time for 100 iterations is 374.643402 ms for a multiplication in a 500x500 matrix with OpenMP
Checking result between cpu and OpenMP
Arrays match.

Average time in CPU 500x500 matrix: 374.643402
Average time in OpenMO 500x500 matrix: 103.663261
Speedup: 3.614042


Matrix size: nx 650 ny 650
Calculating in CPU
Average time for 100 iterations is 885.001160 ms for a multiplication in a 650x650 matrix on Host
Calculating in OpenMP
Killed
```

- **Secuantially on CPU with matrixMult_nothreads.cpp**

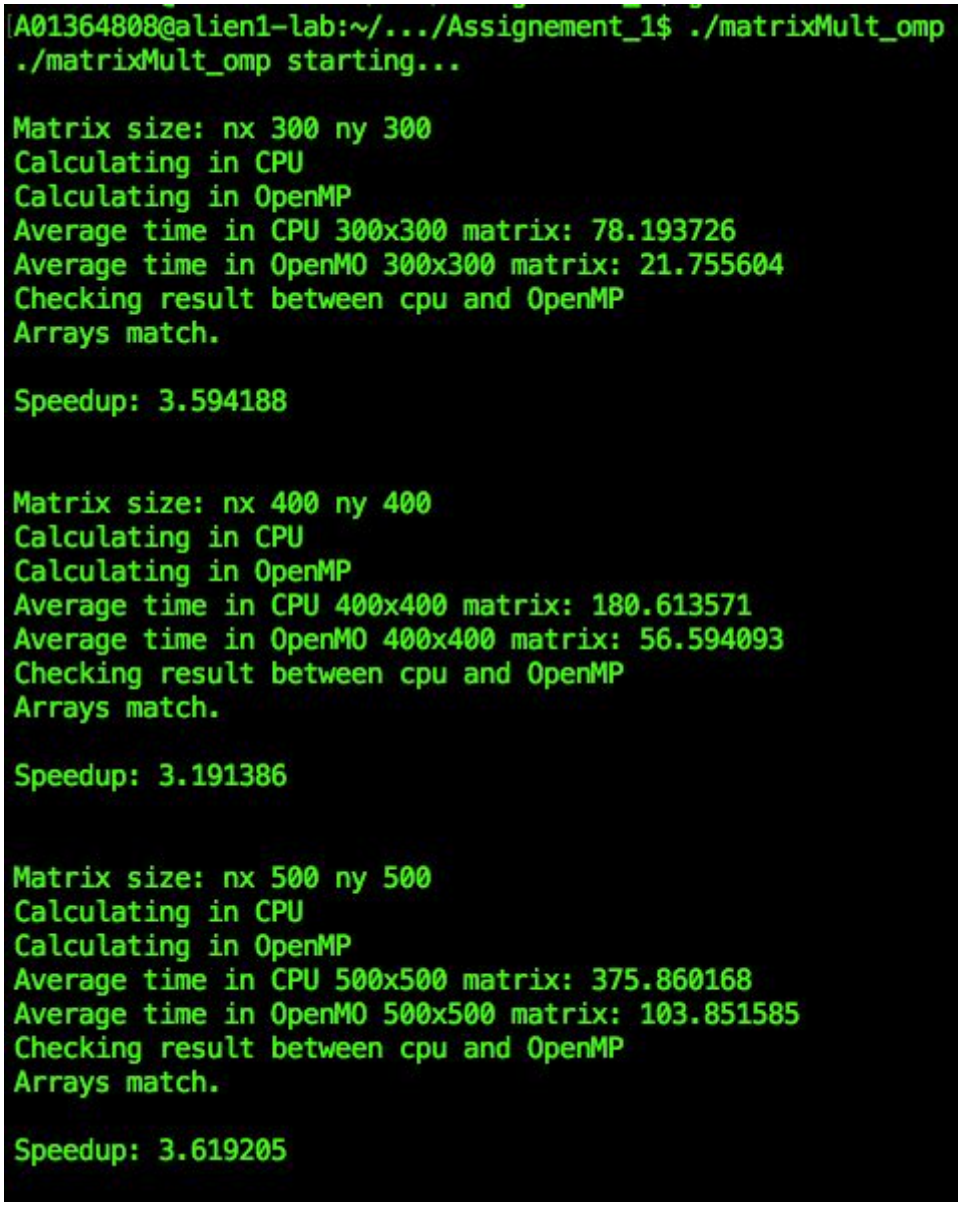
  Compile line:
  g++ -o **matrixMult_nothreads matrixMult_nothreads.cpp** -std=c++11 -Wall

  This file is just to see how much time is elapsed with a CPU sequential implementation

```
A01364808@alien1-lab:~/.../Assignement_1$ ./matrixMult_nothreads
./matrixMult_nothreads starting...
Matrix size: nx 200 ny 200
Average time for 100 iterations is 21.761385 ms for a multiplication in a 200x200 matrix on Host


Matrix size: nx 500 ny 500
Average time for 100 iterations is 375.661743 ms for a multiplication in a 500x500 matrix on Host

Matrix size: nx 800 ny 800
Average time for 100 iterations is 1561.496582 ms for a multiplication in a 800x800 matrix on Host
```

- **Comparison between normal sequential implementation and OpenMP implementation with file matrixMult_omp.cpp**

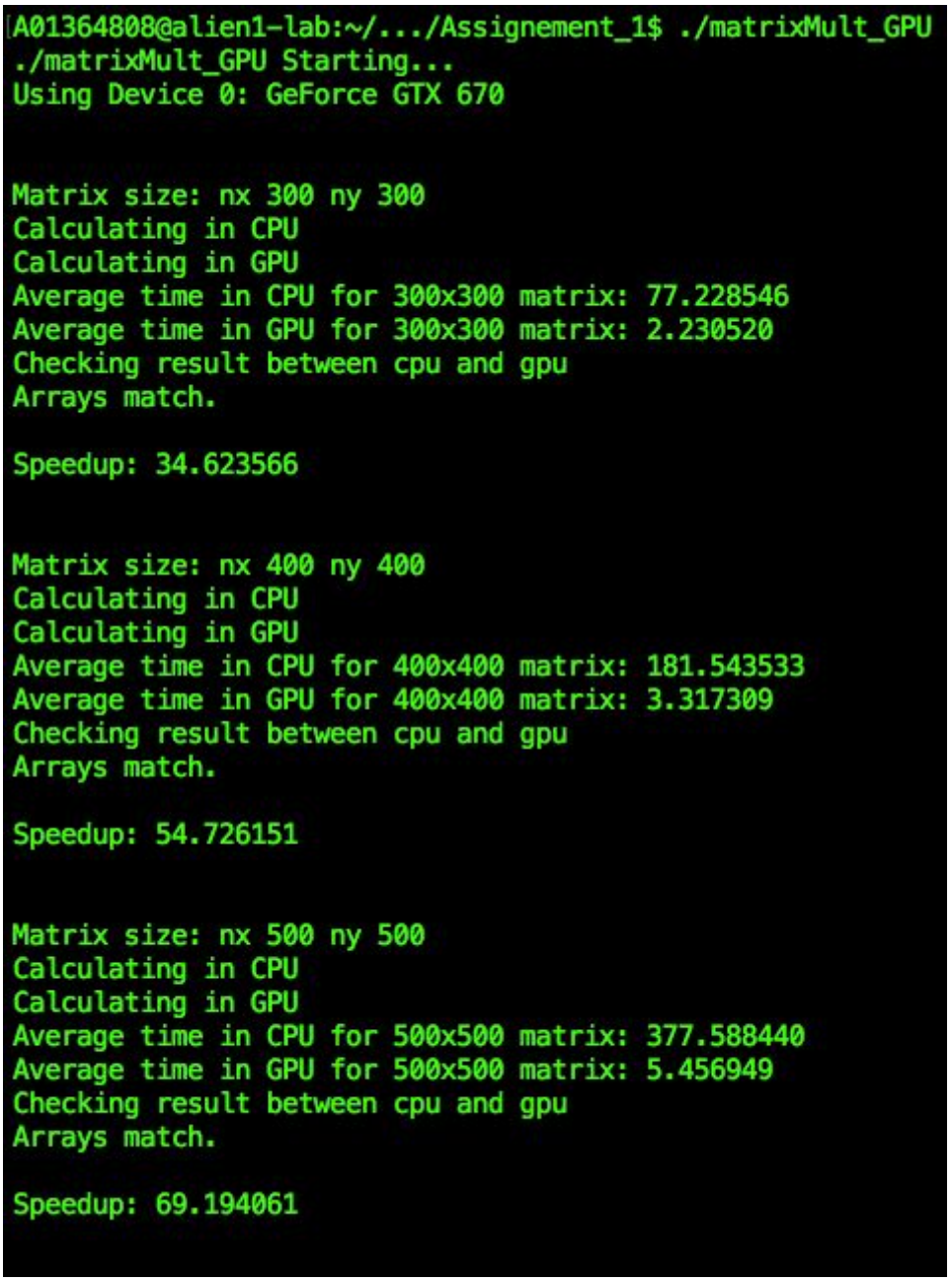
  Compile line:
  g++ -o **matrixMult_omp matrixMult_omp.cpp** -std=c++11 -fopenmp



On this test we can see that the implementation with OpenMP is from 3.19 to 3.61 faster than in a sequential manner. This test calculate 100 times the multiplication, so the times that we see in the image is an average of those 100 iterations per established size.

- **Comparison between normal sequential implementation and GPU implementation with file matrixMult_GPU.cpp**

  Compile line:
  nvcc -o **matrixMult_GPU matrixMult_GPU.cpp** -std=c++11



On this test we can see that the implementation with GPU is faster every time that the matrices in the multiplication are bigger. For example with 300x300 matrices we have a speedup from CPU to GPU of 34.62, but with 400x400 increased to 54.77 and with 500x500 increased even more to almost 70 times faster with GPU. This test calculate 100 times the multiplication, so the times that we see in the image is an average of those 100 iterations per established size.

- **Comparison with normal sequential implementation, OpenMP and GPU implementation with file Assignment1.cpp**

Compile line:
nvcc -o **Assignment1 Assignment1.cpp** -std=c++11 -Xcompiler -fopenmp
*(You have to add the -Xcompiler flag so OpenMP can work on a Nvidia device with cuda code)

This is the main file of the assignment, in this file we will test different numbers for threads in GPU and compare the speedups between different implementations

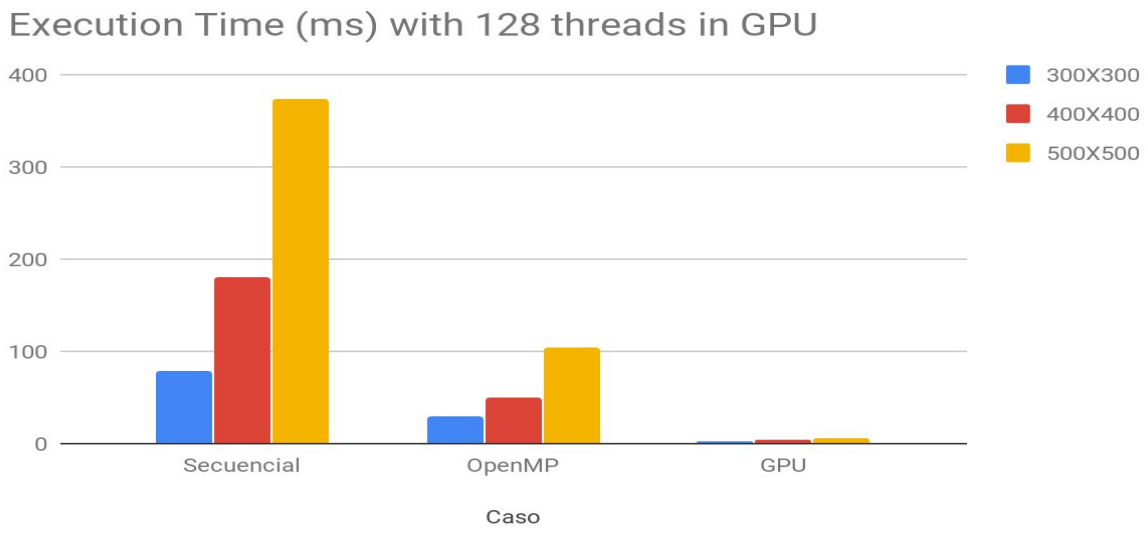
Implementation with 128 threads in a block for GPU

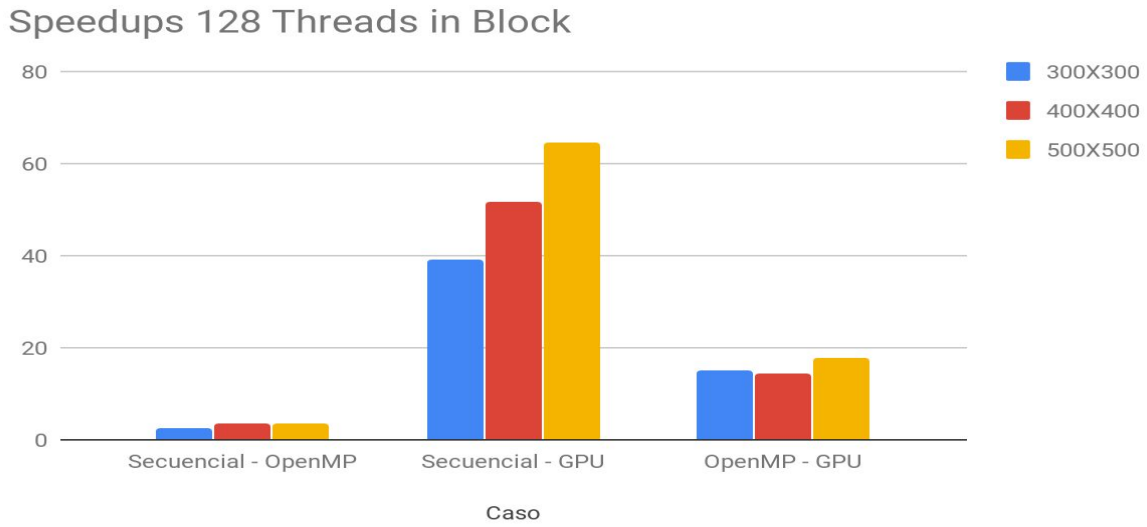
```
A01364808@alien1-lab:~/.../Assignement_1$ ./Assignement1
./Assignement1 Starting...
Using Device 0: GeForce GTX 670
Matrix size: nx 300 ny 300
Calculating on CPU with 300x300
Average time for 100 multiplications in host(no threads) with a matrix of 300 x 300 is 77.922104 ms
Calculating on OpenMP with 300x300
Average time for 100 multiplications in host(using OpenMP) with a matrix of 300 x 300 is 30.315580 ms
Calculating on GPU with 300x300
Average time for 100 multiplications in GPU with a matrix of 300 x 300 is 1.993754 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 2.570365
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 39.083111
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 15.205277
```

```
Matrix size: nx 400 ny 400
Calculating on CPU with 400x400
Average time for 100 multiplications in host(no threads) with a matrix of 400 x 400 is 179.843109 ms
Calculating on OpenMP with 400x400
Average time for 100 multiplications in host(using OpenMP) with a matrix of 400 x 400 is 50.219357 ms
Calculating on GPU with 400x400
Average time for 100 multiplications in GPU with a matrix of 400 x 400 is 3.480811 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.581151
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 51.667011
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 14.427486
```

```
Matrix size: nx 500 ny 500
Calculating on CPU with 500x500
Average time for 100 multiplications in host(no threads) with a matrix of 500 x 500 is 373.899048 ms
Calculating on OpenMP with 500x500
Average time for 100 multiplications in host(using OpenMP) with a matrix of 500 x 500 is 103.560173 ms
Calculating on GPU with 500x500
Average time for 100 multiplications in GPU with a matrix of 500 x 500 is 5.777647 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.610452
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 64.714760
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 17.924282
```

Tables.





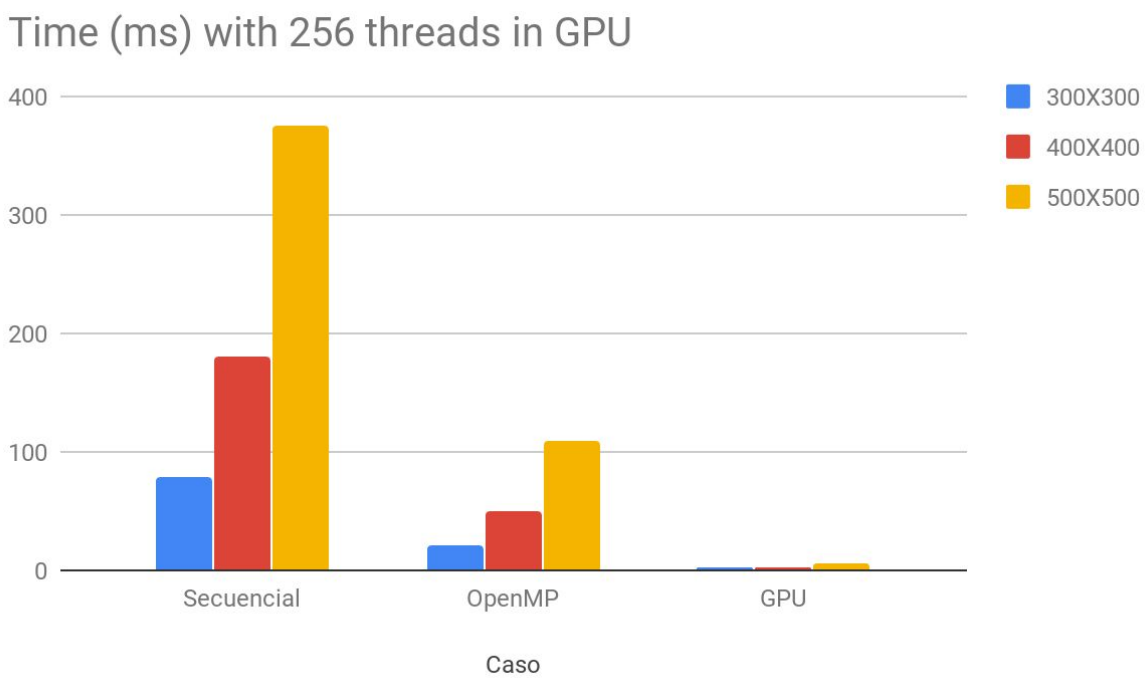
Implementation with 256 threads in a block for GPU

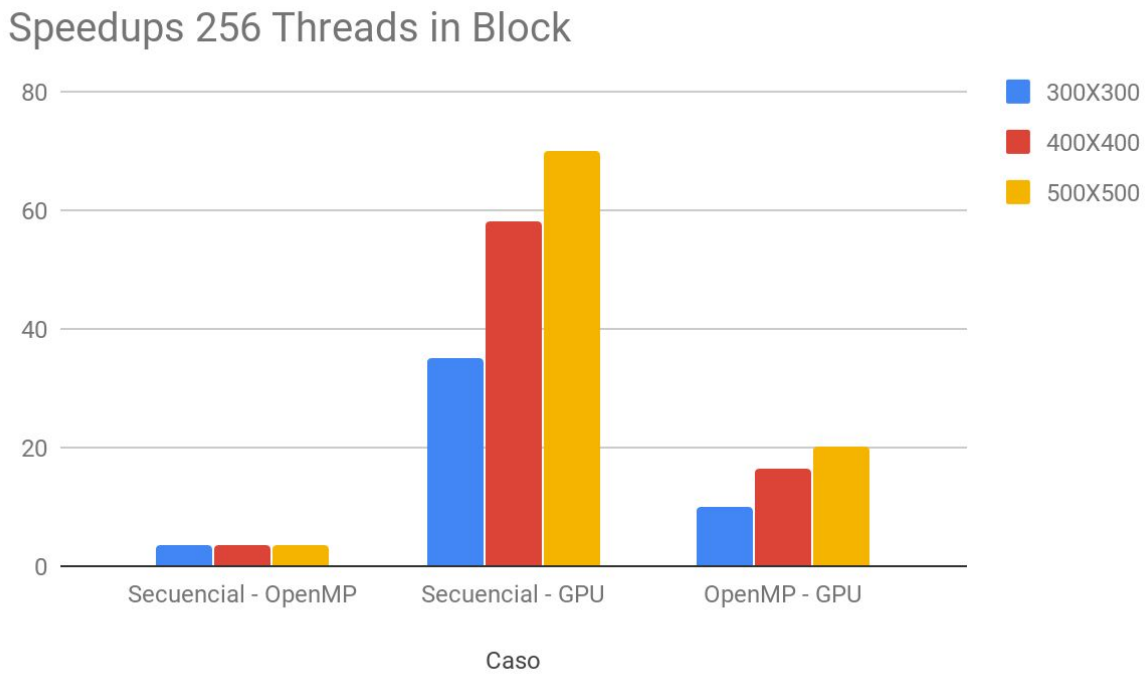
```
A01364808@alien1-lab:~/.../Assignement_1$ ./Assignement1
./Assignement1 Starting...
Using Device 0: GeForce GTX 670
Matrix size: nx 300 ny 300
Calculating on CPU with 300x300
Average time for 100 multiplications in host(no threads) with a matrix of 300 x 300 is 78.150467 ms
Calculating on OpenMP with 300x300
Average time for 100 multiplications in host(using OpenMP) with a matrix of 300 x 300 is 21.832693 ms
Calculating on GPU with 300x300
Average time for 100 multiplications in GPU with a matrix of 300 x 300 is 2.220484 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.579516
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 35.195236
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 9.832401
```

```
Matrix size: nx 400 ny 400
Calculating on CPU with 400x400
Average time for 100 multiplications in host(no threads) with a matrix of 400 x 400 is 180.348343 ms
Calculating on OpenMP with 400x400
Average time for 100 multiplications in host(using OpenMP) with a matrix of 400 x 400 is 50.421825 ms
Calculating on GPU with 400x400
Average time for 100 multiplications in GPU with a matrix of 400 x 400 is 3.100746 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.576791
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 58.162880
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 16.261190
```

```
Matrix size: nx 500 ny 500
Calculating on CPU with 500x500
Average time for 100 multiplications in host(no threads) with a matrix of 500 x 500 is 376.069611 ms
Calculating on OpenMP with 500x500
Average time for 100 multiplications in host(using OpenMP) with a matrix of 500 x 500 is 108.953720 ms
Calculating on GPU with 500x500
Average time for 100 multiplications in GPU with a matrix of 500 x 500 is 5.364536 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.451645
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 70.102913
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 20.309998
```

Tables.


Time (ms) with 256 threads in GPU
400
300
200
100
0
Secuencial
OpenMP
GPU
Caso
300X300
400X400
500X500


Speedups 256 Threads in Block
80
60
40
20
0
Secuencial - OpenMP
Secuencial - GPU
OpenMP - GPU
Caso
300X300
400X400
500X500

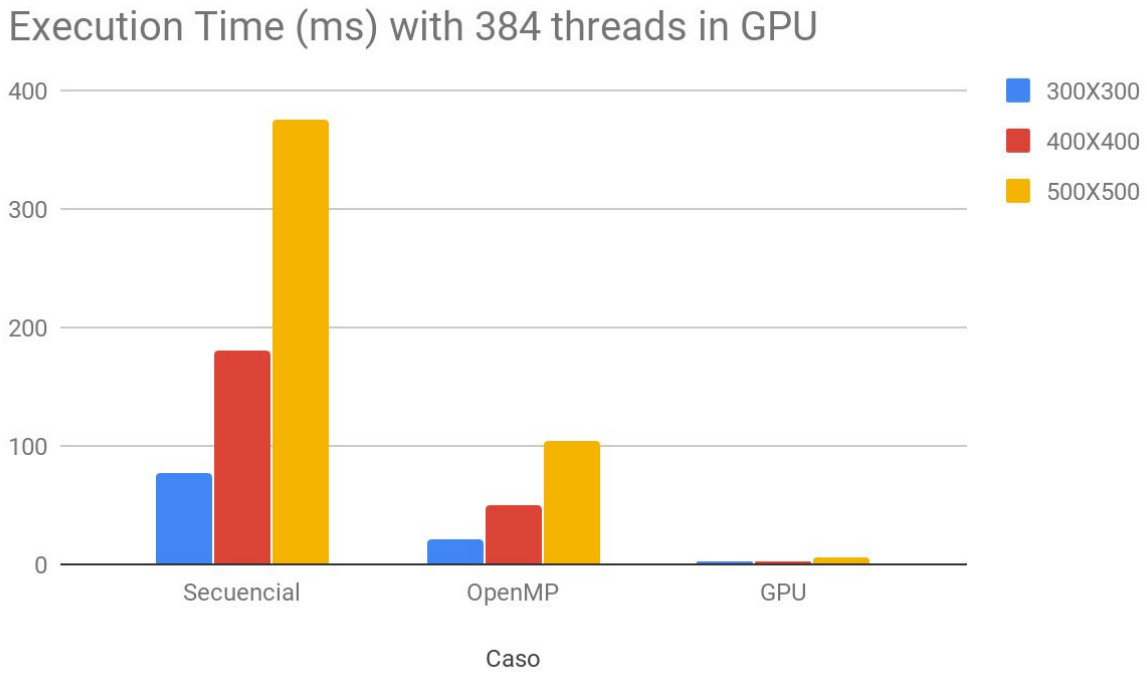
Implementation with 384 threads in a block for GPU

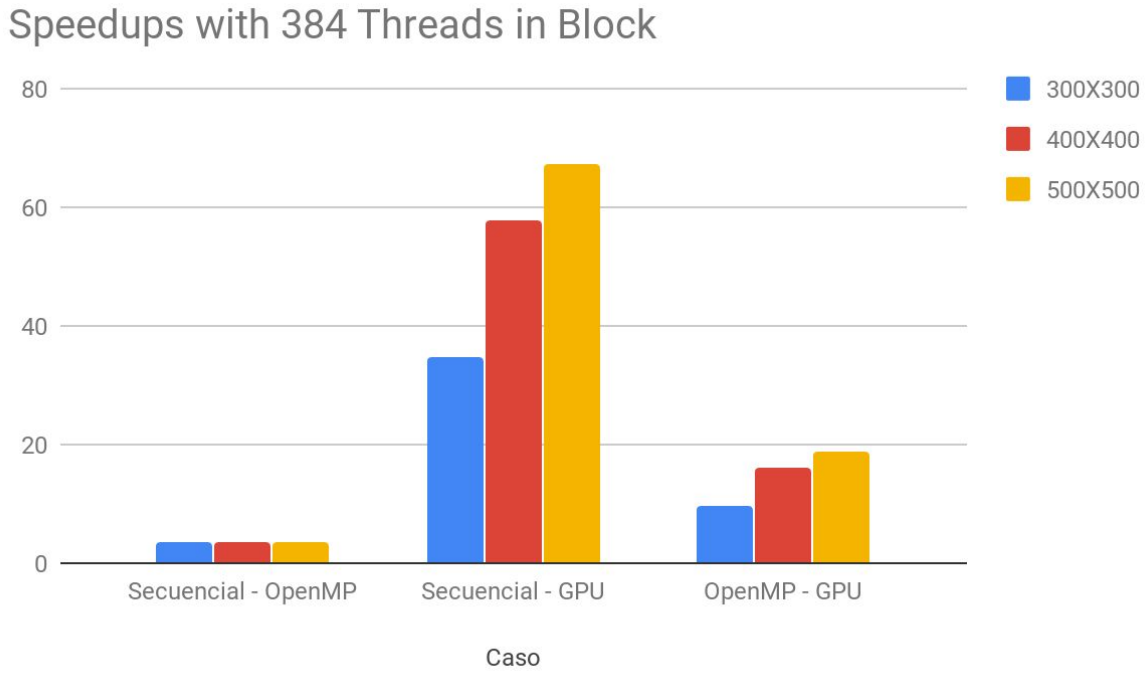
```
A01364808@alien1-lab:~/.../Assignement_1$ ./Assignement1
./Assignement1 Starting...
Using Device 0: GeForce GTX 670
Matrix size: nx 300 ny 300
Calculating on CPU with 300x300
Average time for 100 multiplications in host(no threads) with a matrix of 300 x 300 is 76.949898 ms
Calculating on OpenMP with 300x300
Average time for 100 multiplications in host(using OpenMP) with a matrix of 300 x 300 is 21.683762 ms
Calculating on GPU with 300x300
Average time for 100 multiplications in GPU with a matrix of 300 x 300 is 2.213225 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.548734
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 34.768215
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 9.797359
```

```
Matrix size: nx 400 ny 400
Calculating on CPU with 400x400
Average time for 100 multiplications in host(no threads) with a matrix of 400 x 400 is 180.689316 ms
Calculating on OpenMP with 400x400
Average time for 100 multiplications in host(using OpenMP) with a matrix of 400 x 400 is 50.345802 ms
Calculating on GPU with 400x400
Average time for 100 multiplications in GPU with a matrix of 400 x 400 is 3.118625 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.588965
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 57.938774
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 16.143589
```

```
Matrix size: nx 500 ny 500
Calculating on CPU with 500x500
Average time for 100 multiplications in host(no threads) with a matrix of 500 x 500 is 374.538971 ms
Calculating on OpenMP with 500x500
Average time for 100 multiplications in host(using OpenMP) with a matrix of 500 x 500 is 104.731171 ms
Calculating on GPU with 500x500
Average time for 100 multiplications in GPU with a matrix of 500 x 500 is 5.576104 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.576194
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 67.168579
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 18.782143
```

Tables.


Execution Time (ms) with 384 threads in GPU
400
300
200
100
0
Secuencial
OpenMP
GPU
Caso
300X300
400X400
500X500


Speedups with 384 Threads in Block
80
60
40
20
0
Secuencial - OpenMP
Secuencial - GPU
OpenMP - GPU
Caso
300X300
400X400
500X500

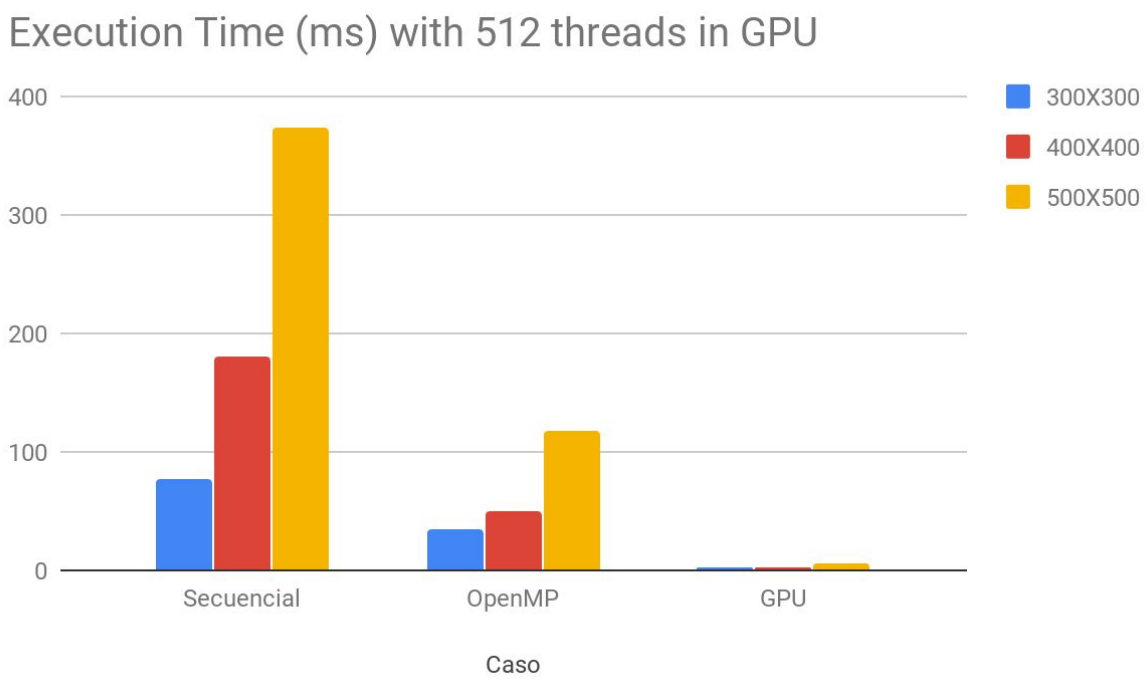
Implementation with 512 threads in a block for GPU

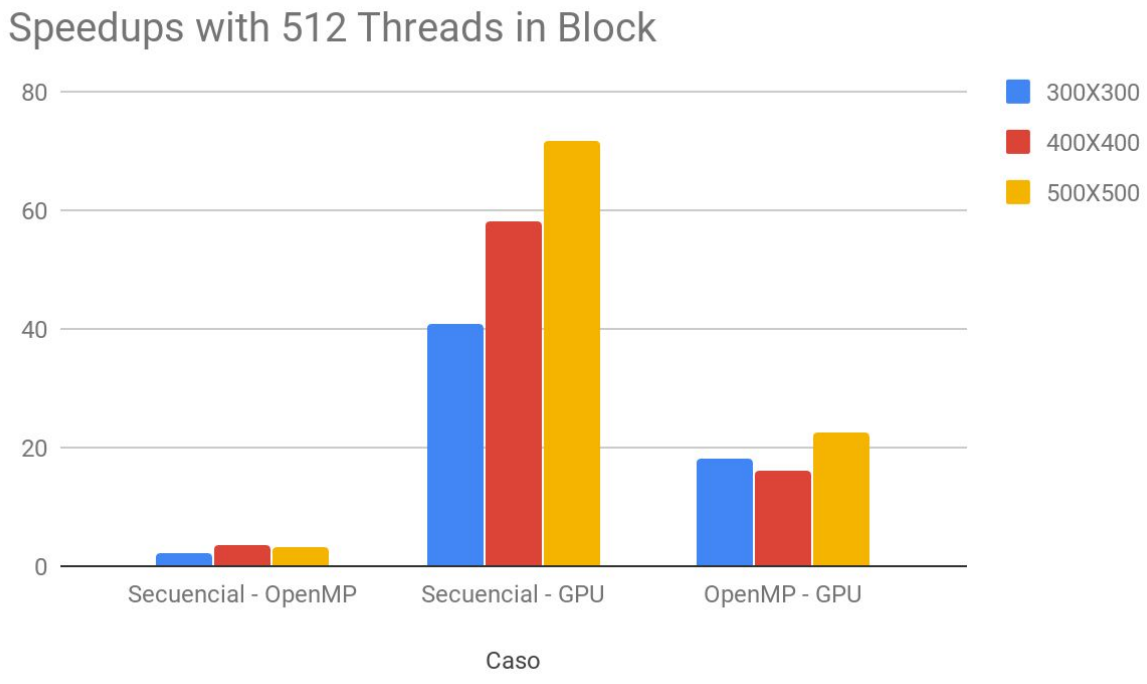
```
A01364808@alien1-lab:~/.../Assignement_1$ ./Assignement1
./Assignement1 Starting...
Using Device 0: GeForce GTX 670
Matrix size: nx 300 ny 300
Calculating on CPU with 300x300
Average time for 100 multiplications in host(no threads) with a matrix of 300 x 300 is 77.242912 ms
Calculating on OpenMP with 300x300
Average time for 100 multiplications in host(using OpenMP) with a matrix of 300 x 300 is 34.191086 ms
Calculating on GPU with 300x300
Average time for 100 multiplications in GPU with a matrix of 300 x 300 is 1.896871 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 2.259154
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 40.721233
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 18.024996
```

```
Matrix size: nx 400 ny 400
Calculating on CPU with 400x400
Average time for 100 multiplications in host(no threads) with a matrix of 400 x 400 is 180.622009 ms
Calculating on OpenMP with 400x400
Average time for 100 multiplications in host(using OpenMP) with a matrix of 400 x 400 is 50.386040 ms
Calculating on GPU with 400x400
Average time for 100 multiplications in GPU with a matrix of 400 x 400 is 3.100493 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.584763
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 58.255898
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 16.250977
```

```
Matrix size: nx 500 ny 500
Calculating on CPU with 500x500
Average time for 100 multiplications in host(no threads) with a matrix of 500 x 500 is 374.275543 ms
Calculating on OpenMP with 500x500
Average time for 100 multiplications in host(using OpenMP) with a matrix of 500 x 500 is 118.281052 ms
Calculating on GPU with 500x500
Average time for 100 multiplications in GPU with a matrix of 500 x 500 is 5.234645 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.164290
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 71.499702
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 22.595812
```

Tables.


Execution Time (ms) with 512 threads in GPU
400
300
200
100
0
Secuencial
OpenMP
GPU
Caso
300X300
400X400
500X500


Speedups with 512 Threads in Block
80
60
40
20
0
Secuencial - OpenMP
Secuencial - GPU
OpenMP - GPU
Caso
300X300
400X400
500X500

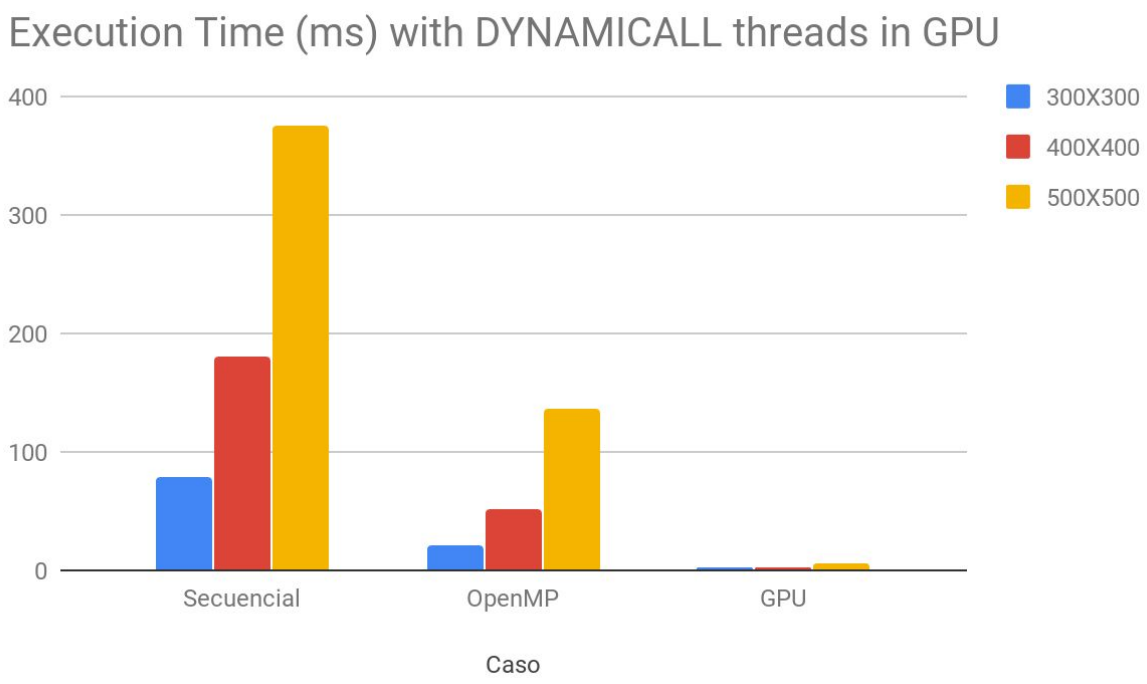
Implementation with DYNAMICALLY calculated threads in a block for GPU

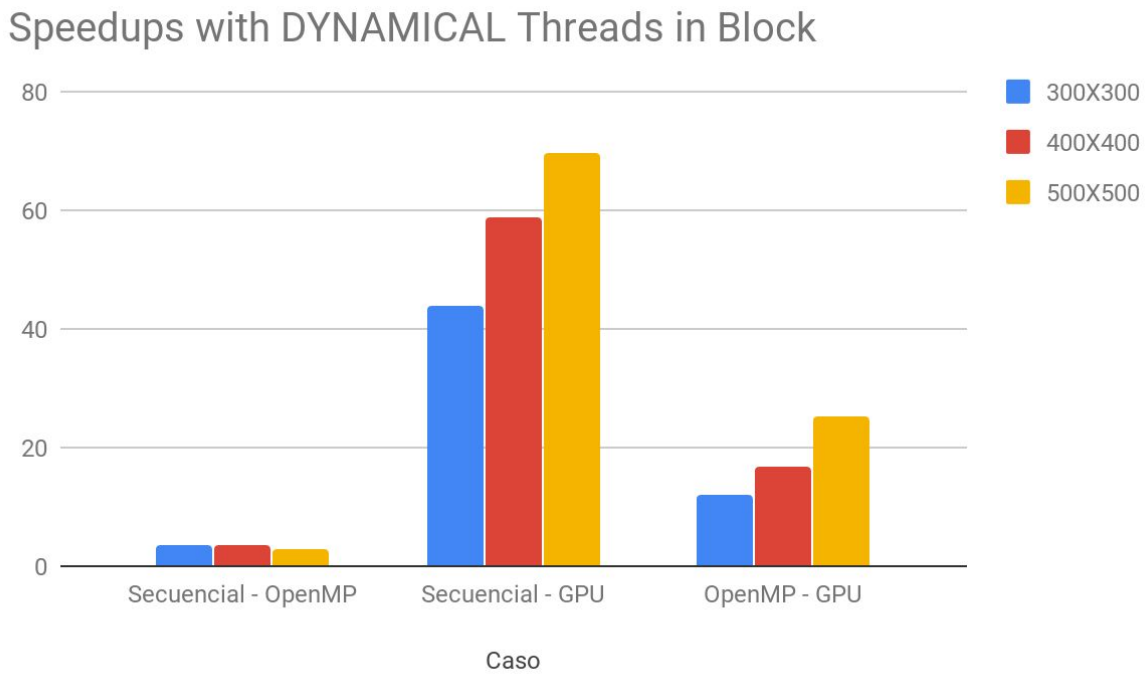
```
A01364808@alien1-lab:~/.../Assignement_1$ ./Assignement1
./Assignement1 Starting...
Using Device 0: GeForce GTX 670
Matrix size: nx 300 ny 300
Calculating on CPU with 300x300
Average time for 100 multiplications in host(no threads) with a matrix of 300 x 300 is 78.341721 ms
Calculating on OpenMP with 300x300
Average time for 100 multiplications in host(using OpenMP) with a matrix of 300 x 300 is 21.748741 ms
Calculating on GPU with 300x300
Average time for 100 multiplications in GPU with a matrix of 300 x 300 is 1.786766 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.602127
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 43.845551
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 12.172129
```

```
Matrix size: nx 400 ny 400
Calculating on CPU with 400x400
Average time for 100 multiplications in host(no threads) with a matrix of 400 x 400 is 180.930237 ms
Calculating on OpenMP with 400x400
Average time for 100 multiplications in host(using OpenMP) with a matrix of 400 x 400 is 51.198292 ms
Calculating on GPU with 400x400
Average time for 100 multiplications in GPU with a matrix of 400 x 400 is 3.069536 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 3.533912
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 58.943832
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 16.679487
```

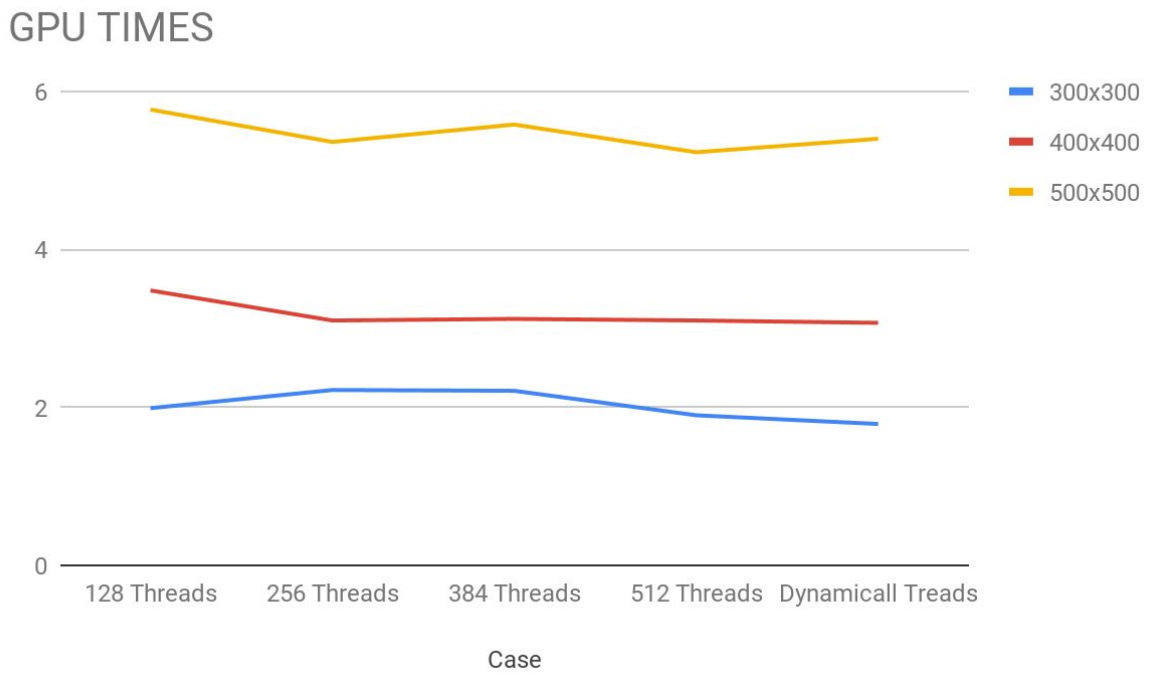
```
Matrix size: nx 500 ny 500
Calculating on CPU with 500x500
Average time for 100 multiplications in host(no threads) with a matrix of 500 x 500 is 376.089020 ms
Calculating on OpenMP with 500x500
Average time for 100 multiplications in host(using OpenMP) with a matrix of 500 x 500 is 136.685257 ms
Calculating on GPU with 500x500
Average time for 100 multiplications in GPU with a matrix of 500 x 500 is 5.399083 ms
Checking result between CPU and OpenMP
Arrays match.

Speedup between CPU and OpenMP: 2.751497
Checking result between CPU and GPU
Arrays match.

Speedup between CPU and GPU: 69.657944
Checking result between OpenMP and GPU
Arrays match.

Speedup between OpenMP and GPU: 25.316383
```

Tables.


Execution Time (ms) with DYNAMICALL threads in GPU
400
300
200
100
0
Secuencial
OpenMP
GPU
Caso
300X300
400X400
500X500


Speedups with DYNAMICAL Threads in Block
80
60
40
20
0
Secuencial - OpenMP
Secuencial - GPU
OpenMP - GPU
Caso
300X300
400X400
500X500

Execution Times Comparison with Different Threads per Block



How to dynamically calculate the threads in a grid block?
Using the ceil function with multiples of 128 with the next function:

```
if (nx > 1024) {
  //If nx > 1024 set the threads number in a block to 128
  dimx = 128;
}
else{
  //Dinamically set the number of threads per block
  dimx = 128 * ((nx + 128 -1) / 128);
}
```

Conclusions.

The good execution or implementation of an algorithm in parallel in a GPU implementation can speed up by, in some cases, 70 times. Algo threads implementations can speed speed up a software time execution a lot, but is not comparable with the GPU´s speed. I also realized that the number of threads by block in a gpu grid can be dynamically calculated, and, at least based in my results, it was the most efficient way to calculate a matrix multiplication. It increased when the number of threads in a grid block was closer to a multiple of 128 and calculated with a ceil function.