

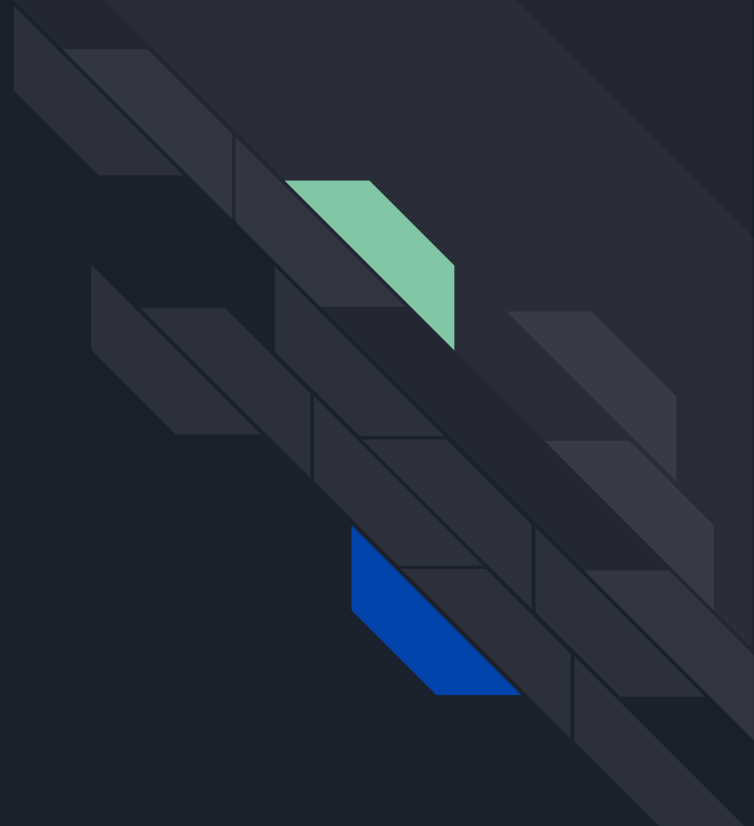


# De Cuda para OpenMP

Gabriela Eckel

Analizando o desempenho do trabalho de Geração de Imagem em Paralelo com CUDA, percebi que obtivemos resultados bem satisfatórios, então pensei... Será que é possível utilizando OpenMP??

Acredito que não seja justo comparar duas ferramentas diferentes, em que cada uma é executada em um lugar diferente, CPU e GPU. Porém gostaria de saber se é possível chegar aos pés de cuda com openmp.. Então, vamos lá?



O experimento





## Parte 1 em Cuda

```
global
void fazPixel(int width, int frames, unsigned char*
pic)
{
    int ix = threadIdx.x;
    int of = blockDim.x;
    for (int frame = ix; frame < frames; frame += of)
    {
        //for (int frame = 0; frame < frames; frame++) {
            for (int row = 0; row < width; row++) {
                for (int col = 0; col < width; col++) {
                    float fx = col - 1024/2;
                    float fy = row - 1024/2;
                    float d = sqrtf( fx * fx + fy * fy );
                    unsigned char color = (unsigned char)
(160.0f + 127.0f *

cos(d/10.0f - frame/7.0f) /

(d/50.0f +
1.0f));
                    pic[frame * width * width + row * width +
col] = (unsigned char) color;
                }
            }
        }
    }
```



## Parte 1 em OpenMP

```
global
void fazPixel(int width, int frames, unsigned char*
pic)
{
    int ix = threadIdx.x;
    int of = blockDim.x;
    for (int frame = ix; frame < frames; frame += of)
    {
        //for (int frame = 0; frame < frames; frame++) {
            for (int row = 0; row < width; row++) {
                for (int col = 0; col < width; col++) {
                    float fx = col - 1024/2;
                    float fy = row - 1024/2;
                    float d = sqrtf( fx * fx + fy * fy );
                    unsigned char color = (unsigned char)
(160.0f + 127.0f *
cos(d/10.0f - frame/7.0f) /
(d/50.0f +
1.0f));
                    pic[frame * width * width + row * width +
col] = (unsigned char) color;
                }
            }
        }
    }
```



## Parte 2 em Cuda

```
global
void fazPixel(int width, int frames, unsigned char*
pic)
{
    int ix = threadIdx.x;
    int of = blockDim.x;
    for (int frame = ix; frame < frames; frame += of)
    {
        //for (int frame = 0; frame < frames; frame++) {
            for (int row = 0; row < width; row++) {
                for (int col = 0; col < width; col++) {
                    float fx = col - 1024/2;
                    float fy = row - 1024/2;
                    float d = sqrtf( fx * fx + fy * fy );
                    unsigned char color = (unsigned char)
(160.0f + 127.0f *

cos(d/10.0f - frame/7.0f) /

(d/50.0f +
1.0f));
                    pic[frame * width * width + row * width +
col] = (unsigned char) color;
                }
            }
        }
    }
```



## Parte 2 em OpenMP

```
global
void fazPixel(int width, int frames, unsigned char*
pic)
{
    int ix = threadIdx.x;
    int of = blockDim.x;
    for (int frame = ix; frame < frames; frame += of)
    {
        //for (int frame = 0; frame < frames; frame++) {
            for (int row = 0; row < width; row++) {
                for (int col = 0; col < width; col++) {
                    float fx = col - 1024/2;
                    float fy = row - 1024/2;
                    float d = sqrtf( fx * fx + fy * fy );
                    unsigned char color = (unsigned char)
(160.0f + 127.0f *

cos(d/10.0f - frame/7.0f) /

(d/50.0f +
1.0f));
                    pic[frame * width * width + row * width +
col] = (unsigned char) color;
                }
            }
        }
    }
```

# Análise de tempo







## Parte 1 em Cuda

<b>Tempo em MS</b>	<b>1024 100</b>	<b>1024 200</b>	<b>2048 100</b>	<b>2048 200</b>
<b>Paralelo</b>	9,4128 ms	17.29366ms	34.42861ms	65.34582m
<b>Sequencial</b>	53.53100ms	105.340000ms	210.847000ms	421.674000ms



## Parte 1 em OpenMP

<b>Tempo em MS</b>	<b>1024 100</b>	<b>1024 200</b>	<b>2048 100</b>	<b>2048 200</b>
<b>Paralelo</b>				
<b>Sequencial</b>				



## Parte 2 em Cuda

<b>xxx</b>	<b>524</b>	<b>1024</b>	<b>2048</b>	<b>4068</b>
<b>32</b>	3.251136ms	3.395616ms	12.61853ms	185.2839ms
<b>64</b>	9.784736ms	6.874368ms	24.14803ms	317.3417ms



## Parte 2 em OpenMP

xxx	524	1024	2048	4068
32				
64				