Fractal com OMP CUDA e OPENCL

Antonio Aldisio 20/2028211
Fernando Miranda 19/0106566
Lorrany Souza 18/0113992

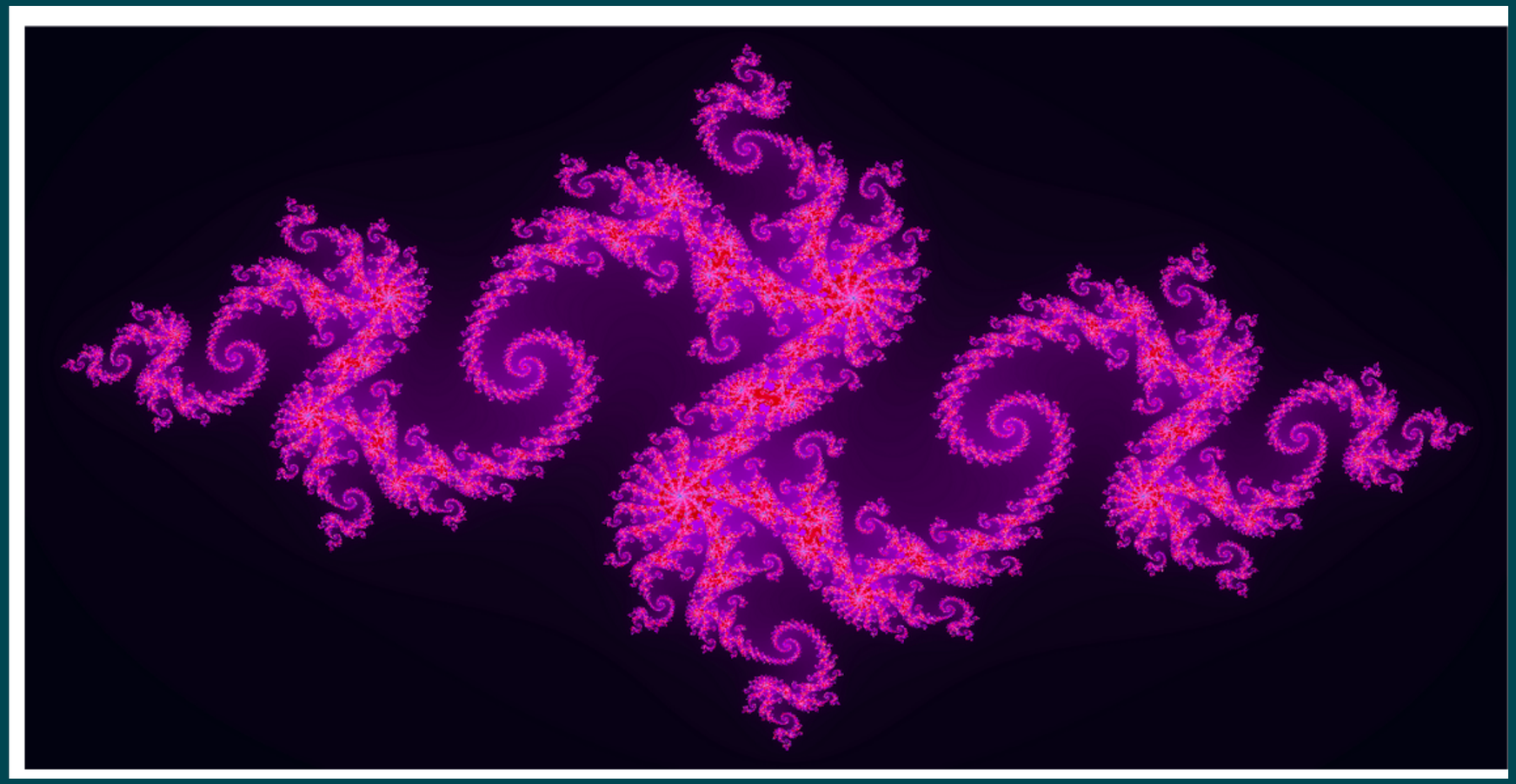
OMP

```
119   #pragma omp parallel for
120   for (int i = 0; i < n; i++)
121     for (int j = 0; j < largura * 3; j += 3) {
122       compute_julia_pixel(j / 3, i, largura, altura, 1.0, rgb);
123       pixel_array[local_i] = rgb[0];
124       local_i++;
125       pixel_array[local_i] = rgb[1];
126       local_i++;
127       pixel_array[local_i] = rgb[2];
128       local_i++;
129     }
```

OMP - Dificuldades

- Não encontramos dificuldades no desenvolvimento

OMP - Resultado



CUDA

```
143   compute_julia_pixels<<<numBlocks, threadsPerBlock>>>(d_pixel_array, largura, altura);
```

```
131   dim3 threadsPerBlock(2, 2);
132   dim3 numBlocks((largura + threadsPerBlock.x - 1) / threadsPerBlock.x, (altura + threadsPerBlock.y - 1) / threadsPerBlock.y);
133
```

CUDA

```
55  __global__ void compute_julia_pixels(unsigned char *pixel_array, int largura, int altura) {
56    int x = blockIdx.x * blockDim.x + threadIdx.x;
57    int y = blockIdx.y * blockDim.y + threadIdx.y;
58    int area = largura * altura;
59    if (x < largura && y < altura) {
60      int local_i = (y * largura + x) * 3;
61      unsigned char rgb[3];
62      compute_julia_pixel(x, y, largura, altura, 1.0, rgb);
63      pixel_array[local_i] = rgb[0];
64      pixel_array[local_i + 1] = rgb[1];
65      pixel_array[local_i + 2] = rgb[2];
66    }
67  }
```

CUDA

```
21  __device__ int compute_julia_pixel(int x, int y, int largura, int altura, float tint_bias, unsigned char *rgb) {
22    // Check coordinates
23    if ((x < 0) || (x >= largura) || (y < 0) || (y >= altura)) {
24      printf("Invalid (%d,%d) pixel coordinates in a %d x %d image\n", x, y, largura, altura);
25      return -1;
26    }
27    // "Zoom in" to a pleasing view of the Julia set
28    float X_MIN = -1.6, X_MAX = 1.6, Y_MIN = -0.9, Y_MAX = +0.9;
29    float float_y = (Y_MAX - Y_MIN) * (float)y / altura + Y_MIN ;
30    float float_x = (X_MAX - X_MIN) * (float)x / largura  + X_MIN ;
31    // Point that defines the Julia set
32    float julia_real = -.79;
33    float julia_img = .15;
34    // Maximum number of iteration
35    int max_iter = 300;
36    // Compute the complex series convergence
37    float real=float_y, img=float_x;
38    int num_iter = max_iter;
39    while (( img * img + real * real < 2 * 2 ) && ( num_iter > 0 )) {
40      float xtemp = img * img - real * real + julia_real;
41      real = 2 * img * real + julia_img;
42      img = xtemp;
43      num_iter--;
44    }
45
46    // Paint pixel based on how many iterations were used, using some funky colors
47    float color_bias = (float) num_iter / max_iter;
48    rgb[0] = (num_iter == 0 ? 200 : - 500.0 * pow(tint_bias, 1.2) *  pow(color_bias, 1.6));
49    rgb[1] = (num_iter == 0 ? 100 : -255.0 *  pow(color_bias, 0.3));
50    rgb[2] = (num_iter == 0 ? 100 : 255 - 255.0 * pow(tint_bias, 1.2) * pow(color_bias, 3.0));
51
52    return 0;
53  }
```

CUDA - Dificuldades

- Como definir de forma adequada numBlocks e threadsPerBlock
- Alocação de memória
- Cor do Fractal

CUDA - Resultado

OPENCL

```
184   ret = clEnqueueNDRangeKernel(command_queue, kernel, 2, NULL, global_item_size, local_item_size, 0, NULL, NULL);
185
```

OPENCL

```
__kernel void compute_julia_pixel(__global int* input_pixels, __global uchar* output_pixels, const int largura, const int altura) {
    int x = get_global_id(0);
    int y = get_global_id(1);

    // Check coordinates
    if ((x < 0) || (x >= largura) || (y < 0) || (y >= altura)) {
        return;
    }

    // Compute the index of the pixel in the input array
    int index = y * largura + x;

    // Extract the RGB values from the input pixel
    int r = (input_pixels[index] & 0xFF0000) >> 16;
    int g = (input_pixels[index] & 0x00FF00) >> 8;
    int b = (input_pixels[index] & 0x0000FF);

    // Compute the new RGB values for the output pixel using the compute_julia_pixel function
    float tint_bias = 0.5;
    unsigned char rgb[3];
    int result = compute_julia_pixel(x, y, largura, altura, tint_bias, rgb);

    // Write the new RGB values to the output pixel
    output_pixels[index * 3] = rgb[0];
    output_pixels[index * 3 + 1] = rgb[1];
    output_pixels[index * 3 + 2] = rgb[2];
}

```

OPENCL - Dificuldades

- Como definir de forma adequada numBlocks e threadsPerBlock
- Construção do arquivo .CL
- Configuração do openCL
- Alocação de memória
- Cor do Fractal

OPENCL - Resultado

OMP vs CUDA vs OPENCL

Host	Tempo de execução - OMP (segundos)	Tempo de execução - CUDA (segundos)	Tempo de execução - OpenCL (segundos)
cm1 (Intel i7-8700)	0,47	-	-
pos1(Intel i7-9700)	0,45	-	-
gpu1(Ryzen 7 2700 + RTX 3060)	0,61	0,0482	0,000004
gpu2(Ryzen 7 2700 + GTX 1650)	0,61	0,1169	0,000004

OMP vs CUDA vs OPENCL

Host	Tempo de execução - OMP (segundos)	Tempo de execução - CUDA (segundos)	Tempo de execução - OpenCL (segundos)
cm1 (Intel i7-8700)	Real 0,815 User 1,918 Sys 0,07	-	-
pos1(Intel i7-9700)	Real 0,889 User 1,904 Sys 0,163	-	-
gpu1(Ryzen 7 2700 + RTX 3060)	Real 1,002 User 2,620 Sys 0,092	Real 0,365 User 0,057 Sys 0,192	Real 0,862 User 0,625 Sys 0,127
gpu2(Ryzen 7 2700 + GTX 1650)	Real 0,948 User 2,466 Sys 0,120	Real 0,401 User 0,131 Sys 0,154	Real 0,892 User 0,623 Sys 0,157

Obrigado.