Trabalho 4 - Fractais de Mandelbrot

Lana Bertoldo Rossato

Estratégias adotadas

- O primeiro **for** calcula os frames. Logo, paralelizar o cálculo dos mesmos.

```
for (int frame = 0; frame < frames; frame++) {
  const double xMin = xMid - delta;
  const double yMin = yMid - delta;
  const double dw = 2.0 * delta / width;
  for (int row = 0; row < width; row++) {
    const double cy = yMin + row * dw;
    for (int col = 0; col < width; col++) {
      const double cx = xMin + col * dw;
      double x = cx;
      double y = cy;
      int depth = 256;
      double x2, y2;
      do {
        x2 = x * x;
        y2 = y * y;
        y = 2 * x * y + cy;
        x = x2 - y2 + cx;
        depth--;
      } while ((depth > 0) && ((x2 + y2) < 5.0));
      pic[frame * width * width + row * width + col] = (unsigned char)depth;
    }
  }
  delta *= 0.98;
}
```

Estratégias adotadas

- Uso de **#pragma omp parallel for** modificando o schedule.
- Algumas modificações foram necessárias para utilizar a estratégia escolhida.

fractalpar1.cpp

```
#pragma omp parallel for schedule(dynamic) num_threads(threads) private(frame)
for (int frame = 0; frame < frames; frame++) {
  delta = Delta * pow(0.98, frame);
```

fractalpar2.cpp

```
#pragma omp parallel for schedule(static) num_threads(threads) private(frame)
for (int frame = 0; frame < frames; frame++) {
  delta = Delta * pow(0.98, frame);
```

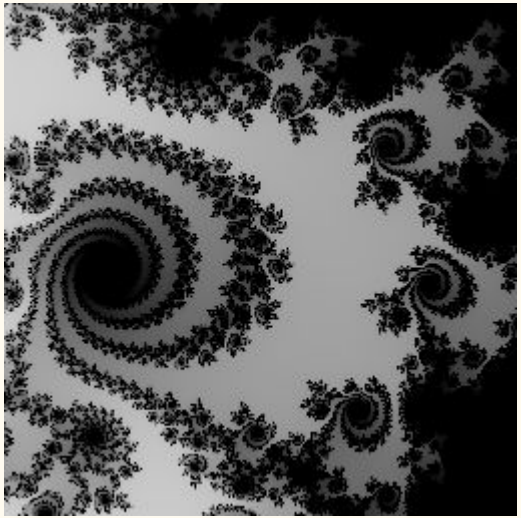
Experimentos realizados

- Além de modificar o número de threads (2, 4, 8), foram utilizados as seguintes variações de tamanho e quantidade de frames:
  1. 1024 32
  2. 1024 64
  3. 512 64

- 10 execuções foram realizadas para cada configuração diferente.

Experimentos realizados

- Antes de tudo, para confirmar que as modificações estavam certas, GIFs foram feitos com o ImageMagick.



Experimentos realizados

- Após, foram calculadas as médias.

Speedup					
		Nº de threads			
programa		1 thread			
	fractal	39,2149			1024 32
		74,86768			1024 64
		18,3874			512 64
		Nº de threads			
		2 thread	4 threads	8 threads	
	fractarpar1	38,1832	38,4988	38,86861	1024 32
		73,7405	73,3610	74,7706	1024 64
		18,6555	18,8107	18,4577	512 64
		Nº de threads			
		2 thread	4 threads	8 threads	
	fractalpar2	38,5009	39,4145	39,2123	1024 32
		73,8646	75,2244	75,8731	1024 64
		18,7075	19,0074	19,1582	512 64

Tam-frame

Experimentos realizados

- E também o cálculo de speedup e eficiência.

Speedup			
fractalpar1			
Tam	2 threads	4 threads	8 threads
1024 32	1,027021	1,018601	1,008909
1024 64	1,015286	1,020538	1,001298
512 64	0,98563	0,977496	0,996192

Speedup			
fractalpar2			
Tam	2 threads	4 threads	8 threads
1024 32	1,018546	0,994937	1,000066
1024 64	1,01358	0,995258	0,986748
512 64	0,98289	0,967379	0,959768

Eficiencia			
fractalpar1			
Tam	2 threads	4 threads	8 threads
1024 32	0,51351	0,25465	0,126114
1024 64	0,507643	0,255135	0,125162
512 64	0,492815	0,244374	0,124524

Eficiencia			
fractalpar2			
Tam	2 threads	4 threads	8 threads
1024 32	0,509273	0,248734	0,125008
1024 64	0,50679	0,248814	0,123344
512 64	0,491445	0,241845	0,119971

Discussão dos resultados

- Pode-se concluir que a os tempos médios foram praticamente os mesmo, comparando a execução sequencial com as paralelas para os mesmo argumentos.
- Conforme o número de threads aumenta, o tempo de execução também aumenta.
- Os valores de speedup variam, não dando confiança para dizer que um determinado número de threads deixa a execução mais rápida. Porém, no programa fractalpar2.cpp, com 8 threads todas as configurações tiverem um speedup menor.
- Já na eficiência, a execução com 8 threads obteve menores valores de eficiência.