Centro Universitário Senac
Bacharelado em Ciência da Computação
Arquiteturas paralelas e distribuídas - Trabalho 01

Professor: Leonardo Takuno
{leonardo.takuno@gmail.com}

10 de abril de 2021

1 O conjunto de Mandelbrot

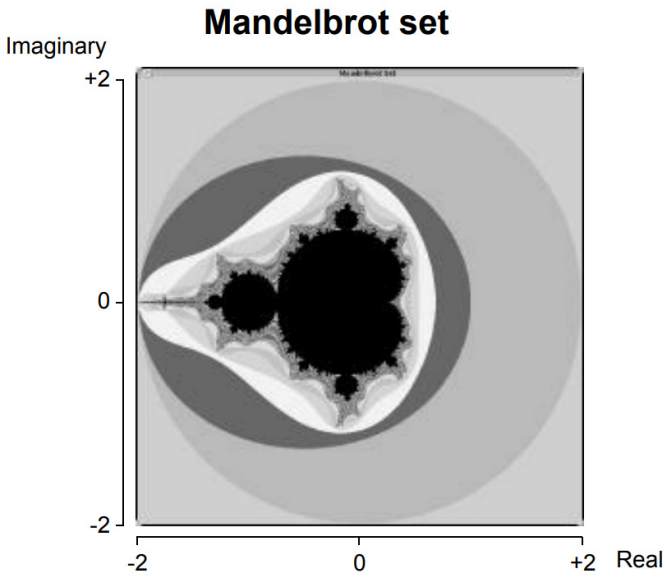
O conjunto de Mandelbrot é um fractal definido como o conjunto de pontos c no plano complexo para o qual a sequência é definida recursivamente:

$$z_0 = 0$$
$$z_{k+1} = z_k^2 + c$$

onde z_{k+1} é a $(k+1)$-ésima iteração do número complexo $z = a + bi$ e c é um ponto no plano complexo. As iterações continuam até que a magnitude de z seja maior que 2 ou o número de iterações alcance um limite arbitrário. A magnitude de z é o tamanho do vetor dado por

$$z_{length} = \sqrt{a^2 + b^2}$$

2 Exemplo de fractal



3 Código sequencial

Para o código sequencial, represente o número complexo usando a estrutura:

```
structure complex {
   float real;
   float imag;
};
```

Uma rotina para calcular z para o ponto c e retornar uma cor pode ser definido como segue:

```
int cal_pixel(complex c) {
   int count, max;
   complex z;
   float temp, lengthsq;
   max = 256;
   z.real = 0; z.imag = 0;
   count = 0; /* number of iterations */

   do {
      temp = z.real * z.real - z.imag * z.imag + c.real;
      z.imag = 2 * z.real * z.imag + c.imag;
      z.real = temp;
      lengthsq = z.real * z.real + z.imag * z.imag;
      count++;
   } while ((lengthsq < 4.0) && (count < max));

   return count;
}
```

Pare quando $\sqrt{a^2 + b^2} \geq 2$ ou quando count alcançar um determinado valor máximo. Isto significa que os pontos de Mandelbrot devem estar dentro do círculo centrado na origem, com raio 2.

4 O sistema de coordenadas

Para construir uma imagem em um mapa de pixels de altura h e largura w. Primeiro, define-se uma janela retangular que pode ser posicionada em qualquer posição no plano complexo.

Então, é preciso mapear cada pixel no plano complexo para determinar o valor correspondente para c. Tais valores encontram-se no *range*: $c_{min} = (real\_min, imag\_min)$ até $c_{max} = (real\_max, imag\_max)$.

Para isso, aplica-se um mapeamento de escala para cada valor (x, y), $0 \leq x < w$ e $0 \leq y < h$ como segue:

```
c.real = real_min + x *(real_max - real_min)/disp_width;
c.imag = imag_min + y *(imag_max - imag_min)/disp_height;
```

Para melhorar a velocidade, defina:

```
scale_real = (real_max - real_min) /disp_width;
scale_imag = (imag_max - imag_min) /disp_height;
```

Para processar cada ponto, utilize:

```
for (x = 0; x < w; x++)
   for (y = 0; y < h; y++) {
      c.real = real_min + ((float)x * scale_real);
      c.imag = imag_min + ((float)y * scale_imag);
      color = cal_pixel(c);
      display(x,y,color);
   }
```

Onde `display()` é uma rotina adequada para mostrar os pixels com a cor indicada.

5 Tarefa

Você deve paralelizar o código para o cálculo do conjunto de Mandelbrot usando OpenMP e CUDA. Esses programas devem, necessariamente, gerar um arquivo PNG como saída, ao invés de gerar para a tela do computador. Depois, você deve medir o tempo de execução para diferentes tamanhos e números de threads.

A implementação deste trabalho em OpenMP e CUDA é relativamente simples. A parte trabalhosa é a elaboração de um relatório, no formato *Jupyter Notebook*, que apresente gráficos com os resultados obtidos com as paralelizações.

5.1 Experimentos

Para cada uma das versões do programa, você deverá realizar as medições do tempo de execução para diferentes tamanhos de entradas e também para diferentes números de threads.

Você deve fazer um número de medições e analisar a variação dos valores obtidos. Sugerimos 10 medições para cada experimento, e também que você use a média e o intervalo de confiança das 10 medições nos seus gráficos. Caso observem variabilidade muito grande nas medições, resultando o intervalo de confiança muito grande, você pode realizar mais medições, sempre apresentando a média e o intervalo de confiança. Não é recomendado fazer menos de 10 medições.

5.2 Apresentação dos resultados

Depois de realizar os experimentos você deverá elaborar gráficos que evidenciam o comportamento das versões do programa de acordo com a variação do tamanho da entrada e número de threads. Os gráficos deverão ser claros e legíveis. Deverão apresentar a média e o intervalo de confiança das 10 execuções (no mínimo) para cada cenário experimental.

5.3 Discussão dos resultados

Você deverá analisar os resultados obtidos e tentar responder a algumas perguntas:

- Como as versões do programa se comportam com a variação:
  - Do tamanho da entrada?
  - Do número de threads?
- Quais outras perguntas interessantes que podem detectar elementos que influenciam no desempenho do seu programa?
- Como você poderia melhorar o desempenho destas versões ? Ou seja, qual outra estratégia você adotaria?

6 Entrega

Você deverá entregar via blackboard os seguintes itens:

- Um arquivo `.ipynb` com as análises e gráficos;
- Uma versão sequencial, uma versão em OpenMP e uma versão em CUDA;
- Submeter o código fonte escrita em C ou Python. Esse código deve, necessariamente, organizado e comentado.
- Um arquivo `.csv` com as medições feitas.

- Gravar um vídeo explicando os detalhes da implementação. E depois demonstrando o funcionamento. Duração do vídeo é de 10 à 15 minutos no máximo.

O trabalho é individual. Evidentemente, você pode discutir as possíveis soluções do trabalho, no entanto, cada aluno deve ser responsável pelo seu próprio trabalho. Qualquer suspeita de fraude será tratada com rigor.

Entrega: 02 de maio de 2021 (domingo).