Implementação e Análise de Linguagens de Programação na Geração do Conjunto de Mandelbrot

Claudio Duarte, Filipe Rutz, Gustavo Estivalet e Otávio Corrêa

Para realização deste trabalho, utilizamos o Conjunto de Mandelbrot com duas linguagens de programação: **Go** e **Python**. O objetivo principal é demonstrar a interoperabilidade entre as linguagens, onde Go é utilizada para a execução intensiva de cálculos matemáticos e Python para a visualização dos dados gerados, aproveitando a vasta biblioteca de visualização matplotlib. O Conjunto de Mandelbrot é um fractal conhecido, gerado por um processo iterativo, o que o torna um excelente caso de estudo para comparar o desempenho e a aplicação de diferentes linguagens de programação.

Metodologia

A abordagem adotada consiste em dividir as responsabilidades da seguinte forma:

- **Go**: Responsável pelo núcleo computacional. O código em Go foi otimizado para calcular as iterações de convergência para cada ponto em um plano complexo, com dimensões de 800 x 800 pixels. A escolha de Go se deu por sua alta performance e concorrência, características ideais para tarefas que envolvem processamento intensivo. O resultado das iterações é impresso na saída padrão como uma matriz de valores inteiros.
- **Python**: Atua como a camada de visualização. O script em Python executa o programa em Go como um subprocesso, captura a saída padrão e processa os dados. Utilizando as bibliotecas numpy e matplotlib, a matriz de dados é convertida em um array e, em seguida, renderizada em uma imagem. A biblioteca matplotlib foi escolhida por sua capacidade de criar visualizações gráficas de alta qualidade de forma simples e eficaz.

---

3. Análise dos Códigos e Implementação

3.1. Implementação em Go

O código em Go implementa a lógica do Conjunto de Mandelbrot de forma direta e eficiente. A função principal percorre uma grade de pixels, e para cada pixel, calcula o número de iterações necessárias para que a sequência zn+1=zn2+c se afaste do círculo de raio 2. A utilização do tipo complex128 nativo da linguagem simplifica bastante a manipulação de números complexos. A saída do programa é uma série de números inteiros, representando a contagem de iterações para cada pixel, separados por espaços e quebras de linha.

Código em Go:

```go
package main

import (
    "fmt"
)

const (
    width   = 800
    height  = 800
    maxIter = 255
)

func main() {
    for y := 0; y < height; y++ {
        for x := 0; x < width; x++ {
            c := complex(-2.0+float64(x)/width*3.0, -1.5+float64(y)/height*3.0)
            z := complex(0, 0)
            iter := 0
            for ; iter < maxIter && real(z)*real(z)+imag(z)*imag(z) < 4; iter++ {
                z = z*z + c
            }
            fmt.Printf("%d ", iter)
        }
        fmt.Println()
    }
}
```

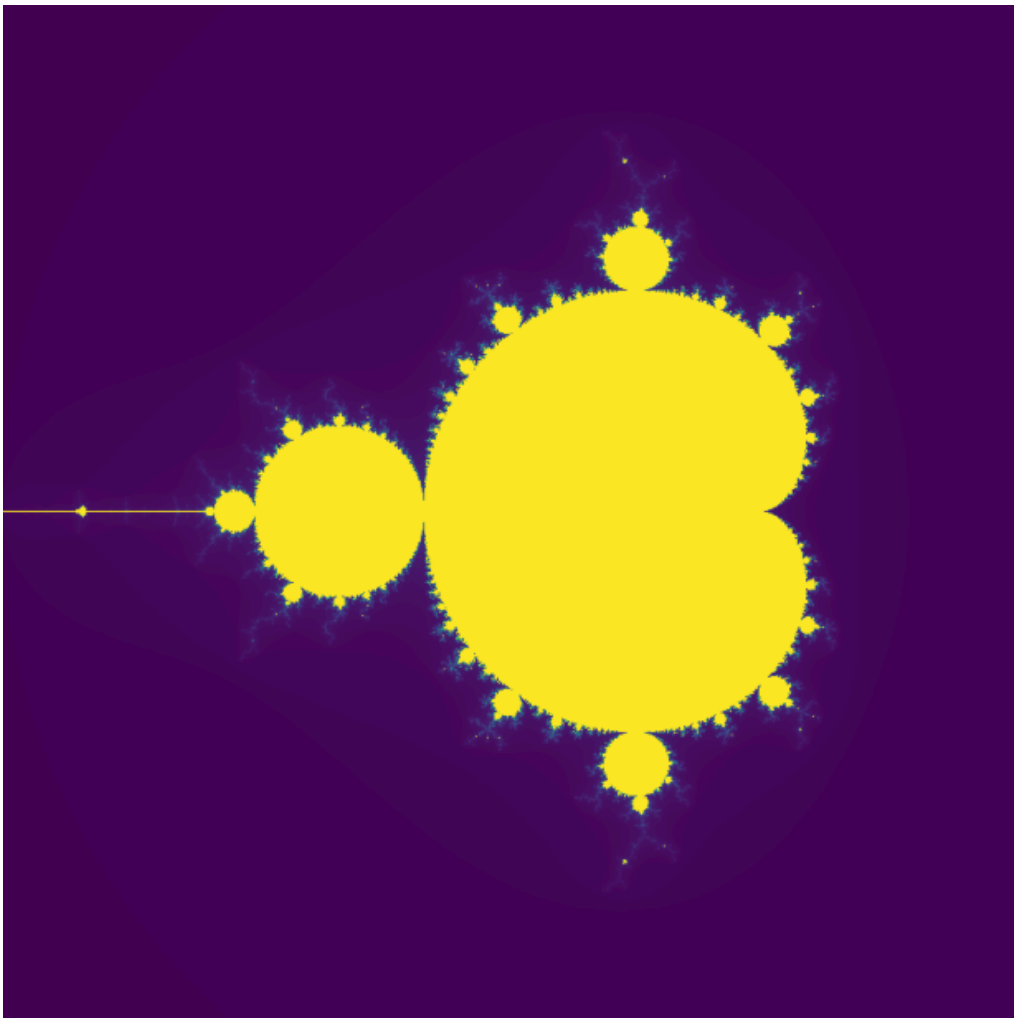
3.2. Implementação em Python

O script em Python atua como um "motor de visualização". Ele utiliza o módulo subprocess para executar o programa Go, capturando toda a saída gerada. Em seguida, a saída é processada linha por linha para construir uma matriz de dados. As bibliotecas numpy e matplotlib são então usadas para converter esta matriz em um array numérico e plotar a imagem, aplicando um mapa de cores que realça as complexas estruturas do fractal.

Código em Python:

```python
import numpy as np
import matplotlib.pyplot as plt
import subprocess

def main():
    process = subprocess.run(["go", "run", "mandelbrot.go"], capture_output=True, text=True)
    go_output = process.stdout

    lines = go_output.strip().split('\n')
    data = []
    for line in lines:
        row = [int(val) for val in line.split()]
        data.append(row)

    mandelbrot_array = np.array(data)

    plt.figure(figsize=(12,8))
    plt.imshow(mandelbrot_array, cmap='viridis')
    plt.axis('off')
    plt.show()

if __name__ == "__main__":
    main()
```

Resultado da execução:



---

4. Conclusão

A combinação das linguagens Go e Python se mostrou uma solução eficiente e prática para a geração do Conjunto de Mandelbrot. O código Go foi capaz de realizar os cálculos complexos de forma rápida, provando ser uma excelente escolha para a parte computacional. Por sua vez, o script Python transformou os dados brutos gerados por Go em uma imagem nítida e detalhada, sem a necessidade de reimplementar a lógica de cálculo. Essa abordagem demonstra a capacidade de interoperabilidade entre linguagens, permitindo que cada uma seja utilizada em sua área de maior força: Go para processamento e Python para visualização.