# Paradigmas de Processamento Paralelo na Resolução do Fractal de Mandelbrot

Bruno Pereira dos Santos

Dany Sanchez Dominguez


**Universidade Estadual de Santa Cruz**

# Cronograma

# Introdução

- Serial vs Processamento Paralelo
  - Processamento Paralelo
    - Resolução de problemas computacionais de grande porte.
      - Engenharias
      - Física Médica
      - Bioinformática
      - Genética
        - » Fontes [Aiping D, 2011] [Alonso P. 2009], [Goddeke D. 2007]
    - Redução de tempo
      - Cluster e Grides
    - Fontes de processamento
      - CPU versus GPU

# Processamento Paralelo

- MPI (Message Passing Interface)
  - Utiliza a CPU
  - Memória distribuída
  - Cooperação na realização da tarefa
- OpenMP (Open MultiProcessing)
  - Utiliza CPU multi-processada
  - Memória Compartilhada
  - Vários *cores* compartilhando memória na cumprimento da tarefa.
- CUDA (Computing Unified Device Architecture)
  - Utiliza a GPU (massivamente paralela)
  - Threads utilizam uma hierarquia de memória para a execução da tarefa.

# Processamento Paralelo

| Características | MPI | OpenMP | CUDA |
|---|---|---|---|
| FONTE DE PODER COMPUTACIONAL | CPU | CPU | GPU |
| MEMÓRIA DISTRIBUÍDA | SIM | NÃO | SIM |
| MEMÓRIA COMPARTILHADA | NÃO | SIM | SIM |

# Problema Computacional

- Fractal
  - São funções recursivas
  - São continuas em todo seu domínio, no entanto em nenhum ponto é diferenciavel
  - Plotagem manual é impraticável
- Benoît Mandelbrot (1924 - 2010)
  - $z = z^2 + c$. Eq. De Pierre Fatou (1878 - 1929)
  - Primeiro conjunto a ser utilizado plotado por um computador
  - A plotagem em resoluções superiores 1200x1200 são excessivamente letas

# Problema Computacional

- Fractal de Mandelbrot

$$z_0 = 0 \qquad (1)$$
$$z_{n+1} = z_n^2 + C \qquad (2)$$

  - Onde $z_0$ e $z_{n+1}$ são iterações $n$ e $n + 1$ e
  - $\mathrm{C} = \mathrm{a} + bi$ é a posição de um ponto no plano complexo que se deseja iterar
- Desenvolvendo as partes real e imaginária obtemos:

$$x_{n+1} = x_n^2 - y_n^2 + a \qquad (3)$$
$$y_{n+1} = 2x_n y_n + b \qquad (4)$$

# Problema Computacional

- Algoritmo
  - Condições de parada
    - ITR é a quantidade de iterações máxima
    - Distância máxima da origem |2|
  - Retorna 0 ou i

```
int conj_mandelbrot(complexo c){
    int I = 0; ITR = 255;
    float x = 0; y =0; tmp = 0;
    enquanto(x² + y² < 2² && i < ITR){
        tmp = x² - y² + c.real;
        y = 2 * y * x + c.img;
        i++;
    }
    Se (i<ITR) retorne i;
    Senão retorne 0;
}
```

# Problema Computacional

- Imagem produzida.

# Resultados obtidos

| Configuração da estação de trabalho 1 | |
|---|---|
| Processador | Intel ® Core i7 CPU 860 2,8GHz |
| Memória RAM | 8GB |
| Placa Aceleradora Gráfica | GPU Nvidia GeForce 9800GT, 112 cores, 512 de RAM, 256bits PCI Express 16x |
| Experimentos com as verões | Serial, OpenMP e CUDA |

| Configuração da estação de trabalho 2 | |
|---|---|
| Processadores | 8 nós Genuine Intel ia-64, modelo Madison com 9M cachê |
| Memória RAM | 16GB Compartilhada |
| Experimentos com a versão | MPI |

# Resultados obtidos

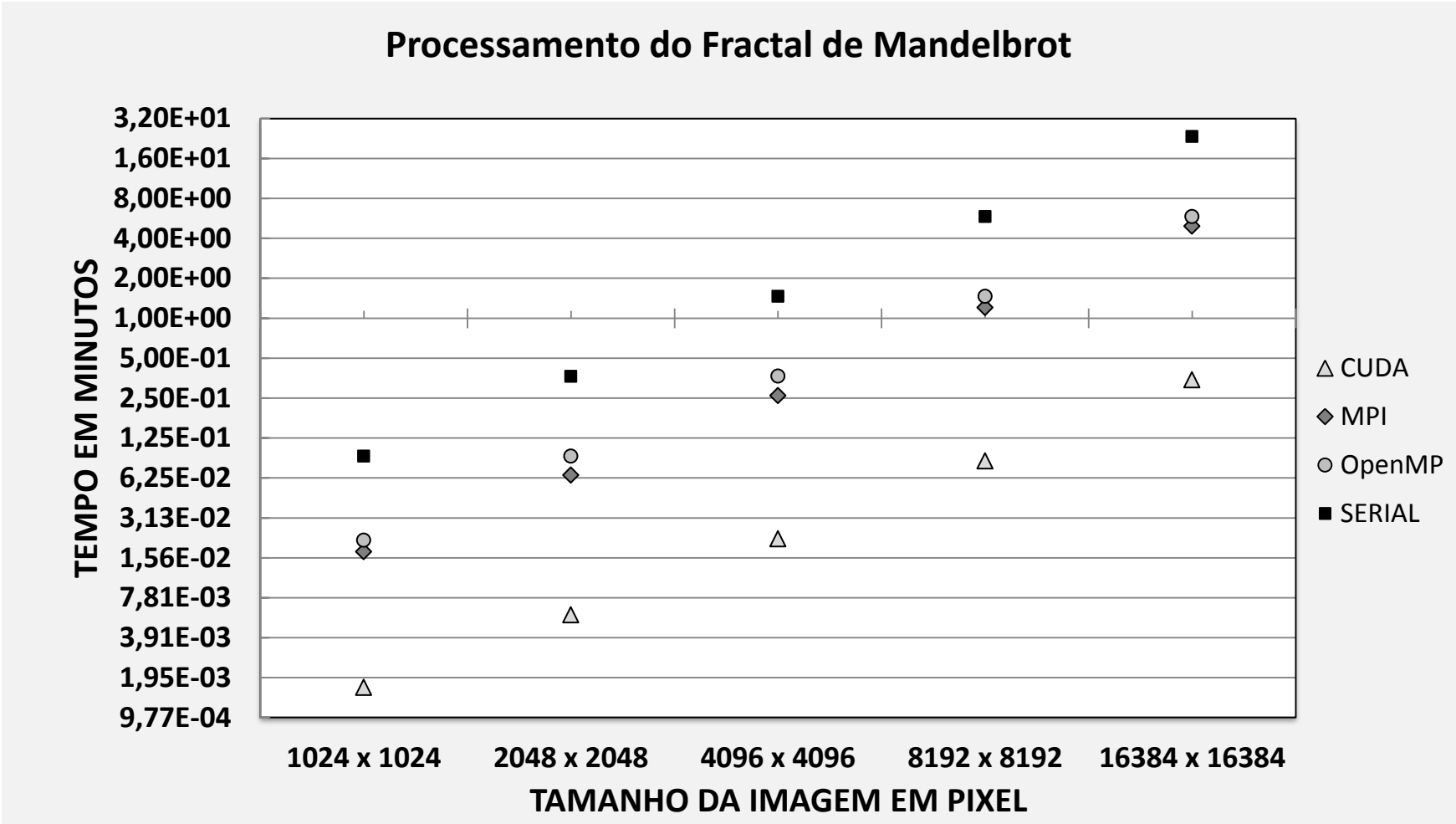


**Gráfico 1 – Tempo de processamento das versões e diversos tamanhos de imagem com ITR = 4096.**

# Resultados obtidos

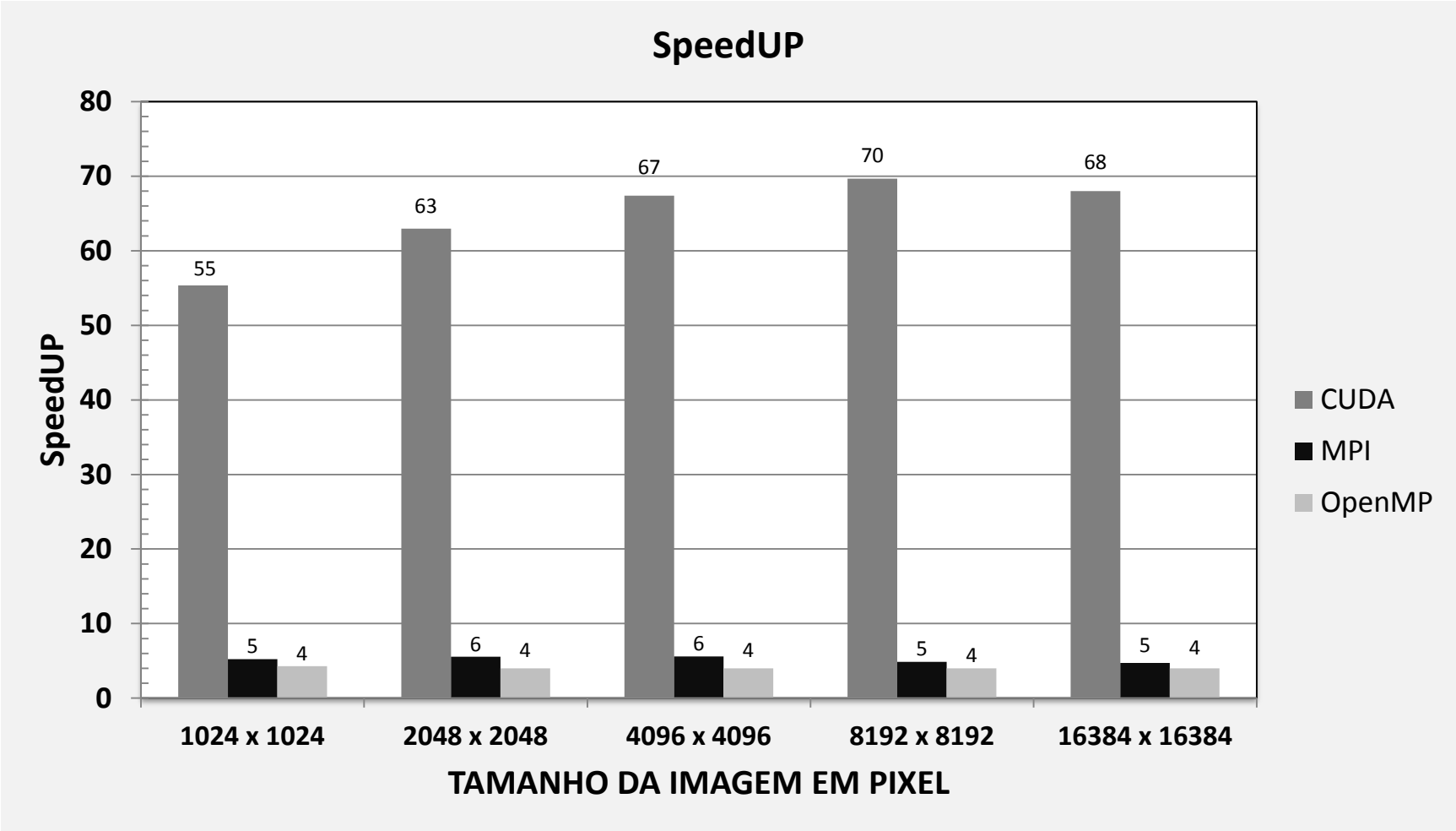


**Gráfico 2 – SpeedUP de processamento das versões em diversos tamanhos de imagem.**

# Conclusões

- Versões paralelas obtiveram melhor resultado na construção do fractal de alta resolução
  - Versão CUDA alcançou melhor tempo de processamento
    - Justificativa
      - Arquitetura com grande quantidade de *cores*
      - Algoritmo altamente paralelizável (Independência dos dados)
      - Baixa transferência de dados
  - MPI
    - SpeedUP parecido com o da versão OpenMP
    - MPI ficando com melhor speedup quando comparado com OpenMP
    - Justificativa
      - Configuração da estação de trabalho utilizada
        - » Processadores do cluster mais robusto
  - Serial
    - Fica claro que é a versão mais lenta obtendo os maiores tempos de processamento

# Conclusões

- Observações sobre técnicas tradicionais (MPI e openMP)
  - Pontos fortes
    - Obtiveram melhor resultado
      - Alto grau de independência dos dados
      - Baixa necessidade de comunicação
      - Sendo boas alternativas a serem exploradas
    - Baixa curva de aprendizado (OpenMP)
  - Pontos fracos
    - Custo do hardware (MPI)
    - Grande espaço e outros recursos auxiliares (MPI)
    - Alta curva de aprendizado (MPI)

# Conclusões

- Observações sobre CUDA
  - Pontos fortes
    - A técnica em GPU apesar de recente é altamente poderosa
      - Especialmente em aplicações altamente paralelizáveis
      - Baixa curva de aprendizagem
      - Menor custo e espaço pelo hardware
  - Pontos fracos
    - Necessidade de um hardware habilitado para CUDA
      - Em contrapartida existe padronizações
        - » OpenCL(Open Computing  Language)

# Trabalhos Futuros

- Melhorar e criar implementações
  - NVIDIA OpenCL Best Practices Guide
  - Versões híbridas
    - OpenMP e MPI
    - OpenMP e CUDA
    - MPI e cuda
    - MPI, OpenMP e CUDA
- Em desenvolvimento
  - Versão paralela em CUDA para o problema de escoamento monofásico de petróleo
    - Fonte: [M. Santos, Dominguez e Orellana 2009]

# Dúvidas