Processamento Paralelo Utilizando GPU

Universidade
Estadual de Santa Cruz

fapesb
Fundação de Amparo
à Pesquisa do Estado da Bahia

Bahia
TERRA DE TODOS NÓS
Secretaria de Ciência,
Tecnologia e Inovação

Bruno Pereira dos Santos
Dany Sanchez Dominguez
Esbel Evalero Orellana

NVIDIA
CUDA

Cronograma

- Breve introdução sobre processamento paralelo
- O que é CUDA?
- Problema abordado (Fractal de Mandelbrot)
- Características da Tecnologia CUDA
- Implementação do Algoritmo
- Resultados Obtidos
- Conclusões e Trabalhos Futuros

Introdução

- Processamento Paralelo
  - Resolução de problemas computacionais de grande porte
    - Engenharia nuclear
    - Física médica
    - Bioinformática
    - Engenharia genética
      - Fontes [Aiping D, 2011] [Alonso P. 2009], [Goddeke D. 2007]
  - Redução de tempo
    - Clusters
    - Grides
- Tecnologias
  - CPU *versus* GPU

Introdução

- Processamento Paralelo em GPU
  - Multi- e many-cores CPU
  - Massively parallel accelerators

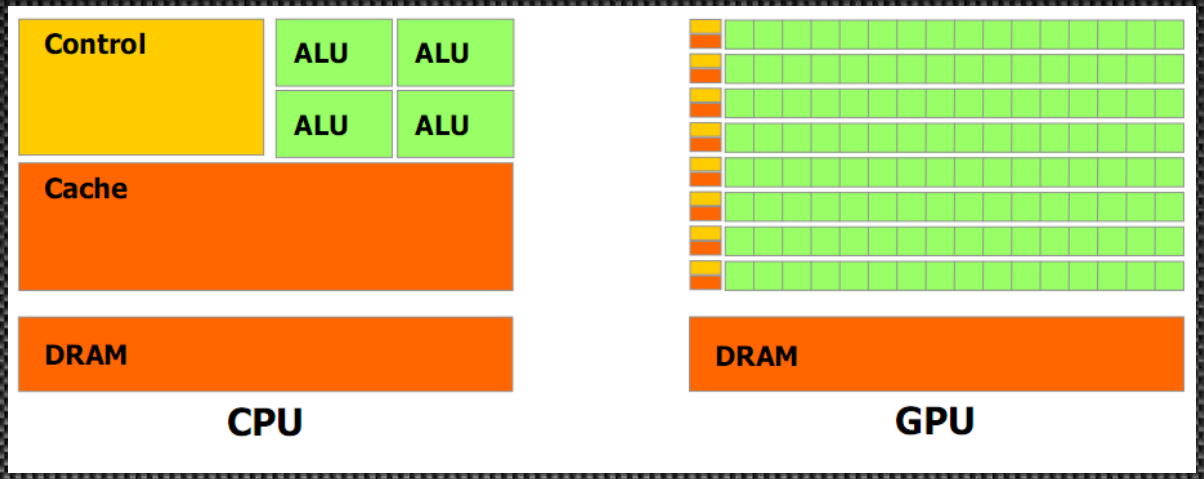


Figura 1 – Arquiteturas CPU x GPU [Nvidia – 1]

Introdução

- GPGPU (acrônimo de General-purpose Computing on Graphics Processing Units)
  - Marks Harris em 2002 definiu o uso das GPUs para fins não gráficos
    - Fonte [GPGPU.org]
- OpenCL (Open Computing Language)
  - Visa por em Prática a GPGPU
  - Framework mantido pelas empresas
    - Intel, AMD, Nvidia, Apple Inc, ATI.
      - Fonte [Nvidia – 3]

O que é CUDA?

- CUDA (Computing Unified Device Architecture)
  - Criada pela Nvidia
  - Aplicar o GPGPU nas placas da Nvidia
  - Extensão da linguagem C e C++
  - Oferece uma API (Application Programming Interface)
    - Driver
    - CUDA runtime e bibliotecas

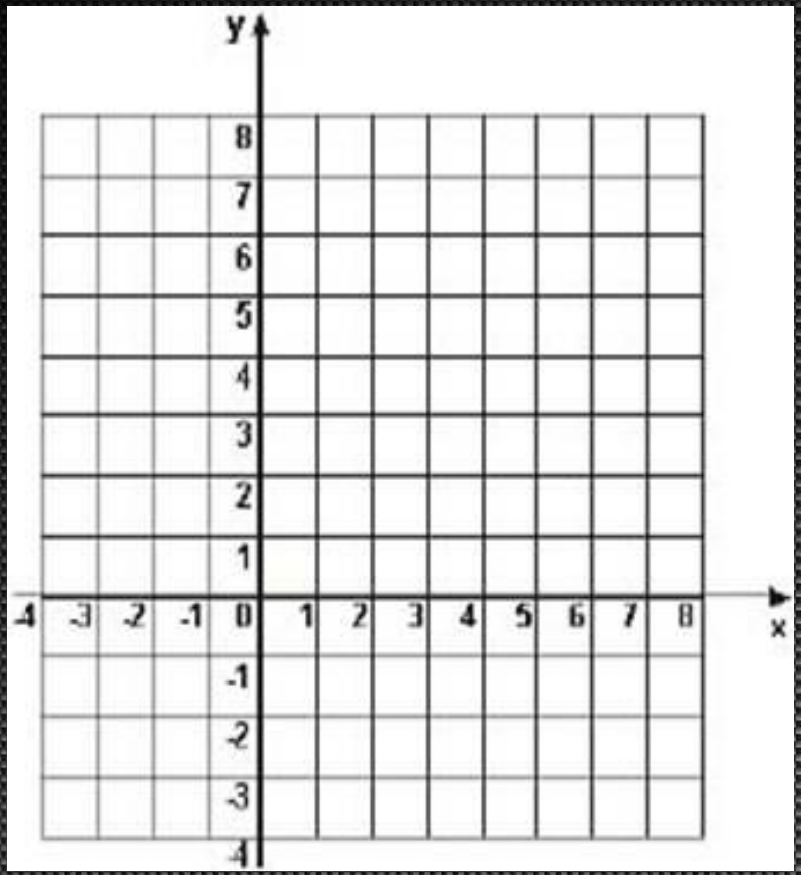
Problema Computacional Abordado

- Fractal de Mandelbrot
  - Funções recursivas
  - Difícil plotagem
  - Foi o primeiro fractal a ser resolvido em um computador
  - Conjunto específico de pontos no plano complexo

Problema Computacional Abordado

- Propriedades do Fractal de Mandelbrot
  - É definido pela recorrência do numero complexo:
    - $Z = x + yi$
    - $Z_0 = 0$
    - $Z_{\{n+1\}} = Z_n^2 + C$
      - Onde Z_0 e $Z_{\{n+1\}}$ são iterações n e $\{n+1\}$
      - e $C = a + bi$ fornece a posição de um ponto do plano complexo a ser iterado
  - Distância máxima de 2 da origem
  - Quantidade máxima de iterações

Problema Computacional Abordado



- Algoritmo

```
int 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*x*y + c.img;
    i++;
}
se(i < ITR)   retorne i;
senão  retorne 0;
}
```

Características da Tecnologia CUDA

- CUDA como um conjunto sorftware e hardware
  - Acrescenta uma nova nomenclatura para as arquiteturas paralelas
    - SIMT (Single Instruction Multiple Threads)

Características da Tecnologia CUDA

- CUDA como um conjunto sorftware e hardware
  - Novo modelo de compilação para arquiteturas paralelas
    - Compilador nvcc
      - Fonte [Nvidia – 2]

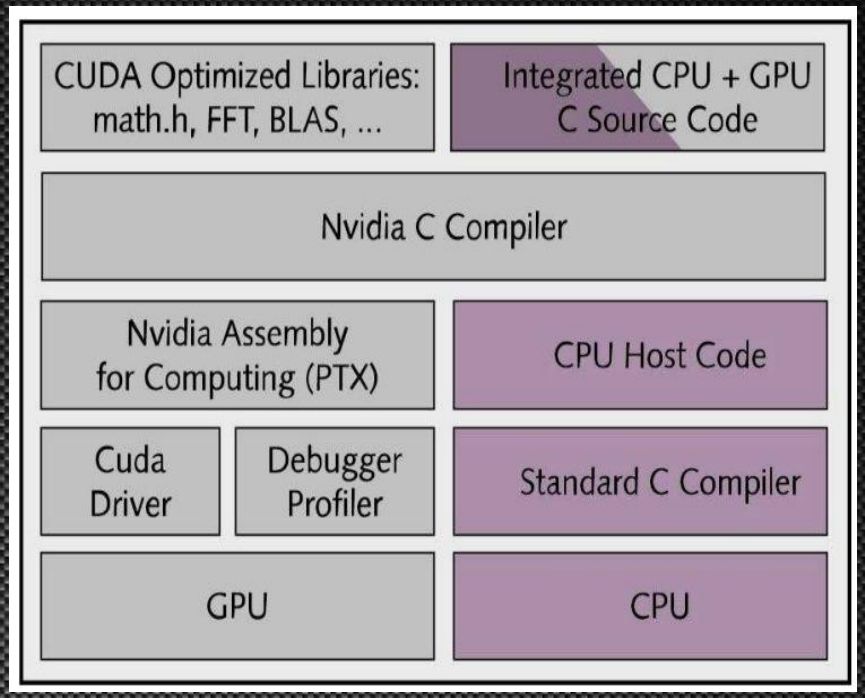


Figura 2 – Compilação [HALFHILL, T.R.]

Características da Tecnologia CUDA

- CUDA como um conjunto sorftware e hardware
  - Fluxo de execução

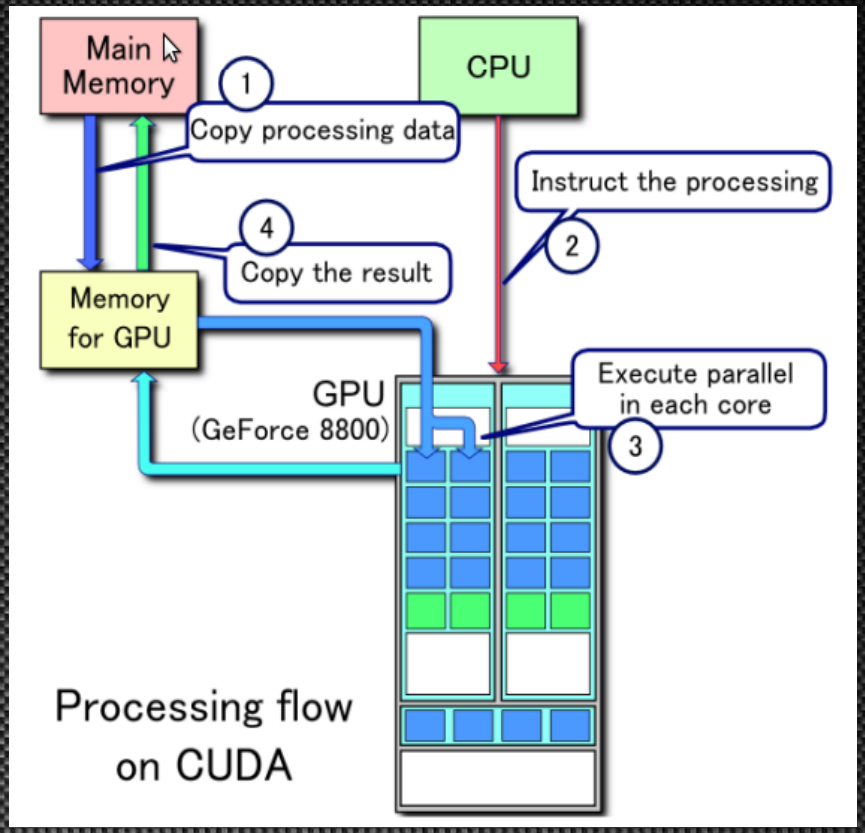
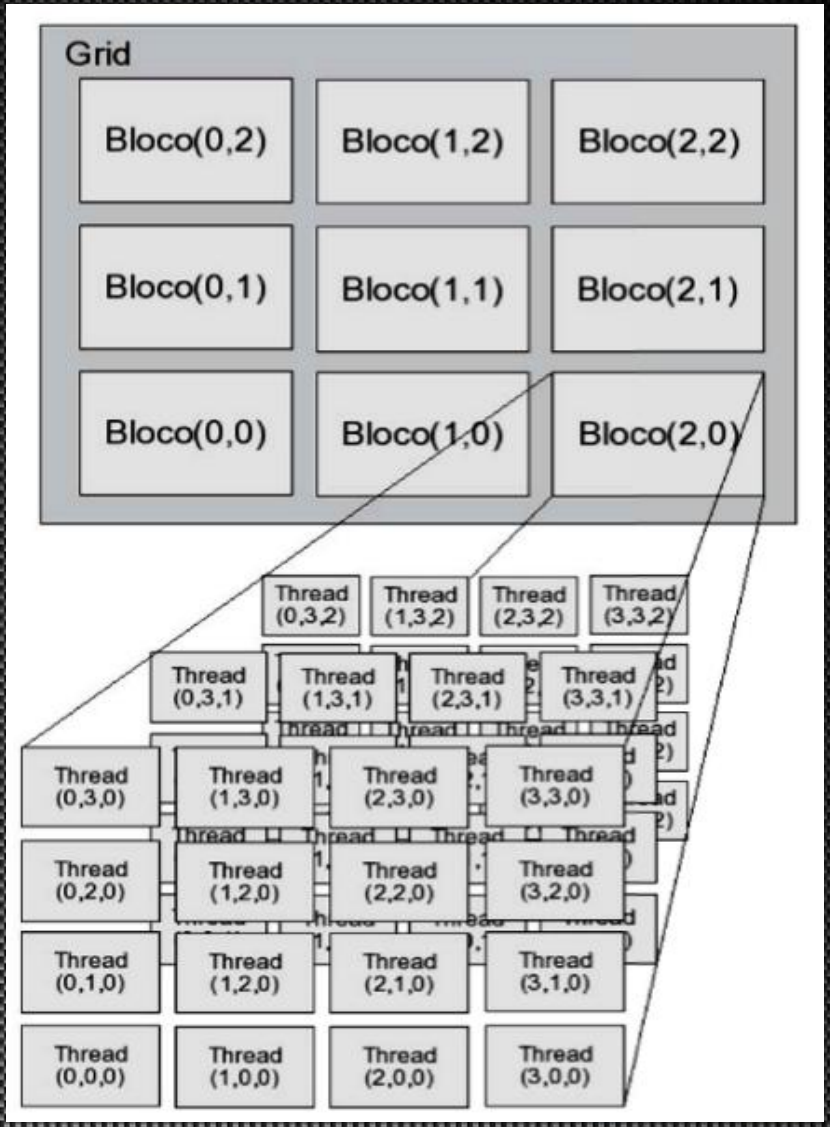


Figura 3 – Fluxo de execução [Nvidia – 1]

Características da Tecnologia CUDA

- Kernel
  - Podem ser organizados em diversas hierarquias
    - Grids formados por blocos
      - Podem ser organizados em até 2 dimensões
    - Bloco formados por threads
      - Podem ser organizados em até 3 dimensões



Características da Tecnologia CUDA

- Extensões de linguagem
  - Qualificadores de tipo
    - Função: __global__, __device__, __host__
    - Variável: __device__, __constante__, __shared__
  - Identificadores de variável threads
    - threadIdx.x, threadIdx.y, threadIdx.z
    - blockIdx.x, blockIdx.y
  - Nova sintaxe para chamada de funções kernel
    - Nome_funcao<<grid, blocos>>(parametros)
  - Variáveis dim3
    - dim3 nome_variavel(x, y, z)

Implementação do Algoritmo

- Código Paralelo x Código Serial

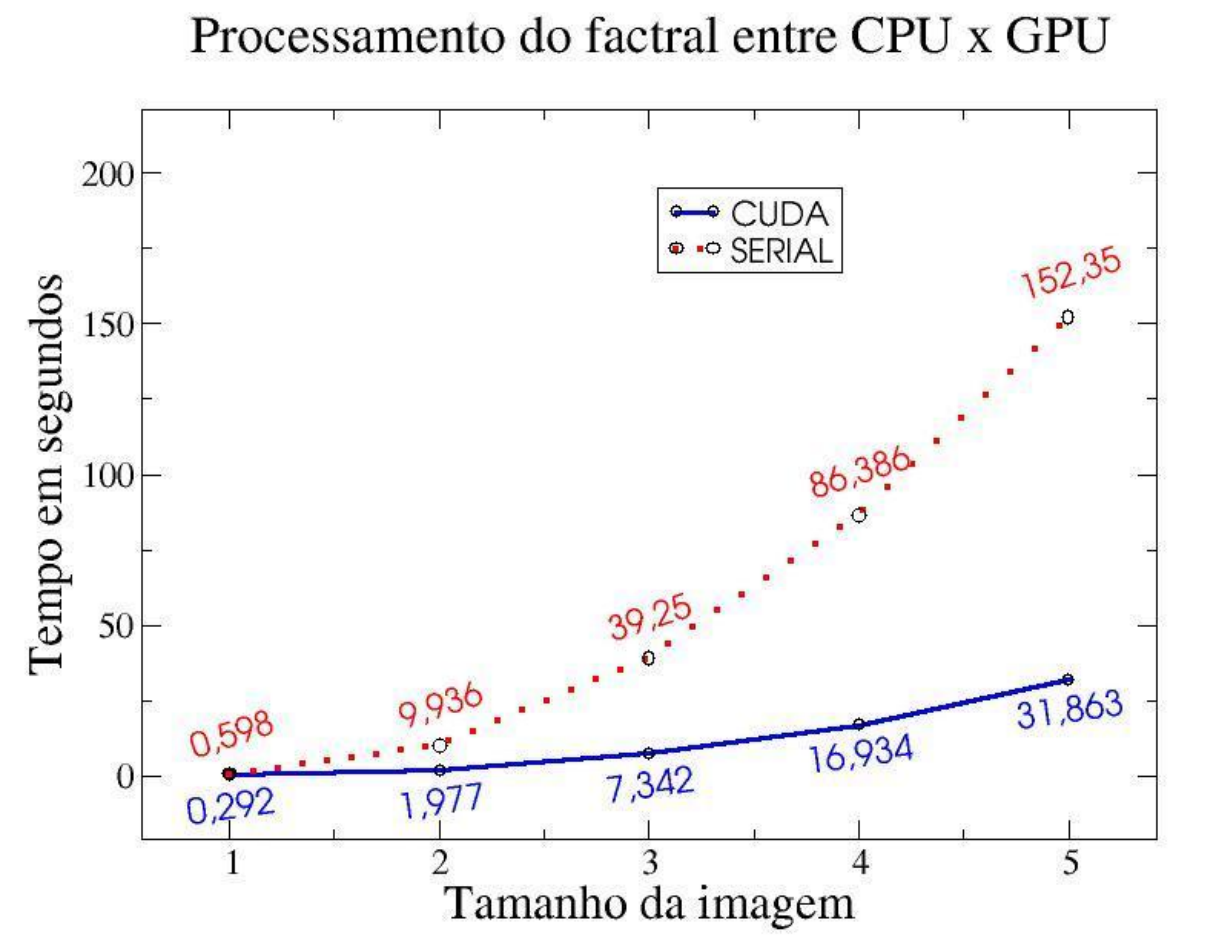
```
__global__  void ponto_fractal(complexo ini,
        char *plano_d,  float dx, float dy, int altura,
        int compr){
   int tx = threadIdx.x + blockIdx.x * blockDim.x;
   int ty = threadIdx.y + blockIdx.y * blockDim.y;
   float real, img;
   real = ini.real + (dx*tx);
   img = ini.img – (dy*ty);
   int i = 0;
   float x = 0; y = 0, tmp;
   while(x*x + y*y  <= 4 &&  i < ITR){
      tmp =  x*x – y*y + real;
      y = 2*x*y + img;
      x = tmp;
      i++;
   }
   if(i<ITR)
     plano_d[tx*comprimento +ty] = i;
   else
     plano_d[tx*comprimento +ty] = 0;
}
```

```
void ponto_fractal(complexo ini){
   int i = 0;
   float x = 0; y = 0, tmp;
   while(x*x + y*y  <= 4 &&  i < ITR){
      tmp =  x*x – y*y + real;
      y = 2*x*y + img;
      x = tmp;
      i++;
   }
   if(i<ITR)
     return = i;
   else
     return = 0;
}
```

Resultados Obtidos

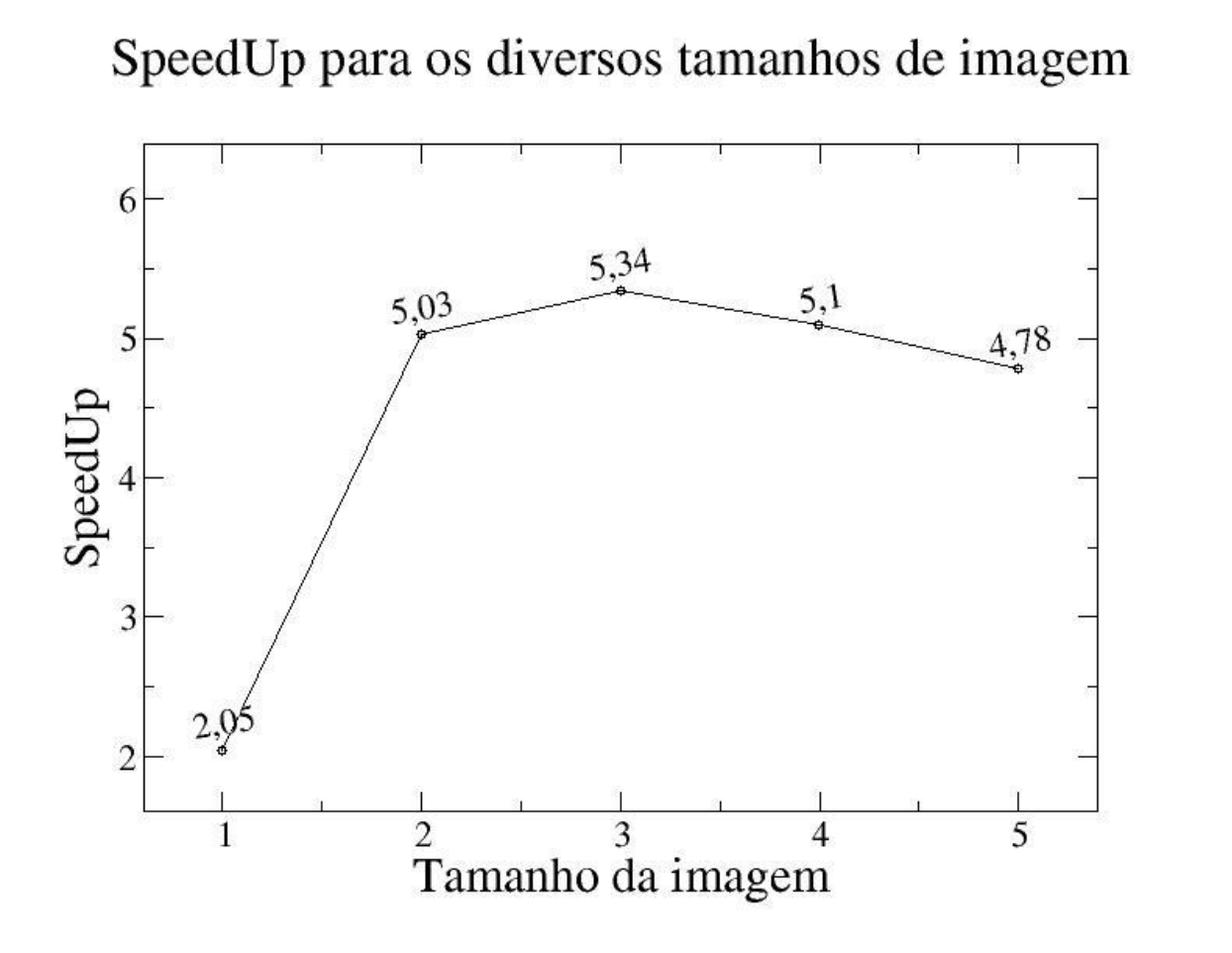
- Ferramentas para os experimentos numéricos
  - Máquina de testes
    - Processador intel (R) Core i7 CPU 860 2,8GHz
    - Placa gráfica Nvidia GeForce 9800GT
      - 512MB de memória princial
      - 112 cores
    - HD de 250GB
    - 8GB de memória RAM
  - Compiladores
    - gcc
    - nvcc

Resultados Obtidos



1 – 1.200x1.200px 2 – 5.000x5.000px
3 – 10.000x10.000px 4 –15.000x15.000px
5 – 20.000x20.000px

Resultados Obtidos



1 – 1.200x1.200px 2 – 5.000x5.000px
3 – 10.000x10.000px 4 –15.000x15.000px
5 – 20.000x20.000px

Conclusões

- Desafios
  - Absorção das diretivas introduzidas pela biblioteca CUDA
  - Novo modelo de arquitetura SIMT
  - Programação com threads
- Conquistas
  - Frameworks para padronização
  - Difusão no meio acadêmico
  - Introdução da tecnologia em computadores de grande porte

Trabalhos Futuros

- Otimizar o código CUDA
  - "NVIDIA OpenCL Best Practices Guide" [Nvidia – 3]
- Implementar o algoritmo com técnicas tradicionais de paralelismo
  - MPI
  - OpenMP
- Efetuar novos comparativos
  - Paralela (CUDA) *versus* Paralela (MPI e OpenMP)

Agradecimentos

Dúvidas

Referências

- Aiping Ding, Tianyu Liu, Chao Liang, Wei Ji, and X George Xu (2011) “EVALUATION OF SPEEDUP OF MONTE CARLO CALCULATIONS OF SIMPLE REACTOR PHYSICS PROBLEMS CODED FOR THE GPU/CUDA ENVIRONMENT”.
- Alonso, P., Cortina, R, Martínez-Zaldívar, F. J., Ranilla, J. (2009) “Neville elimination on multi- and many-core systems: OpenMP, MPI and CUDA, J. Supercomputing”, in press, doi:10.1007/s11227-009-0360-z, SpringerLink Online Date: Nov. 18.
- Goddeke D, Strzodk R, Mohd-Yusof J, McCormick P, H.M Buijssen S, Grajewski M. e Turek S. (2007) "Exploring weak scalability for FEM calculations on a GPU-enhanced cluster".
- GPGPU.org (2011). Disponível em: http://gpgpu.org/about/, Março.
- NVIDIA Corporation, (2011) “NVIDIA CUDA C ProgrammingGuide 3.1.” Disponível em: http://developer.nvidia.com/object/cuda_download.html, Março.
- NVIDIA Corporation, (2011) “The CUDA Compiler Driver NVCC”,disponível em: http://moss.csc.ncsu.edu/~mueller/cluster/nvidia/2.0/nvcc_2.0.pdf, Março.
- NVIDIA Corporation, (2011) “NVIDIA OpenCL Best Practices Guide”. Disponível em: http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/OpenCL_Best_Practices_Guide.pdf, Março.
- HALFHILL,T.R. (2008) “Parallelprocessingwithcuda. MicroprocessorReport”, Janeiro (2011)