Otimizando a Execução de uma Aplicação de Dinâmica dos Fluidos para GPUs

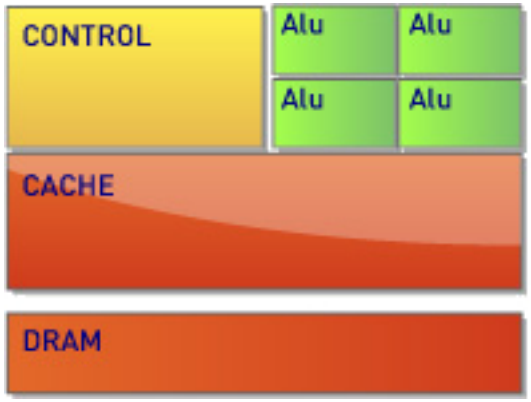
Matheus da Silva Serpa, Claudio Schepke
matheusserpa@alunos.unipampa.edu.br

LEA: Laboratório de Estudos Avançados em Computação
Ciência da Computação – Campus Alegrete

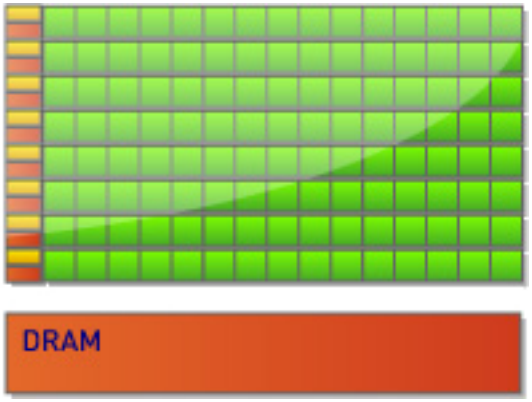
Roteiro

- Introdução
- Modelo de Programação CUDA
- Problema
- Método de *Lattice Boltzmann*
- Metodologia
- Resultados
- Conclusões e Trabalhos Futuros

Introdução



- **Arquitetura multi-core**

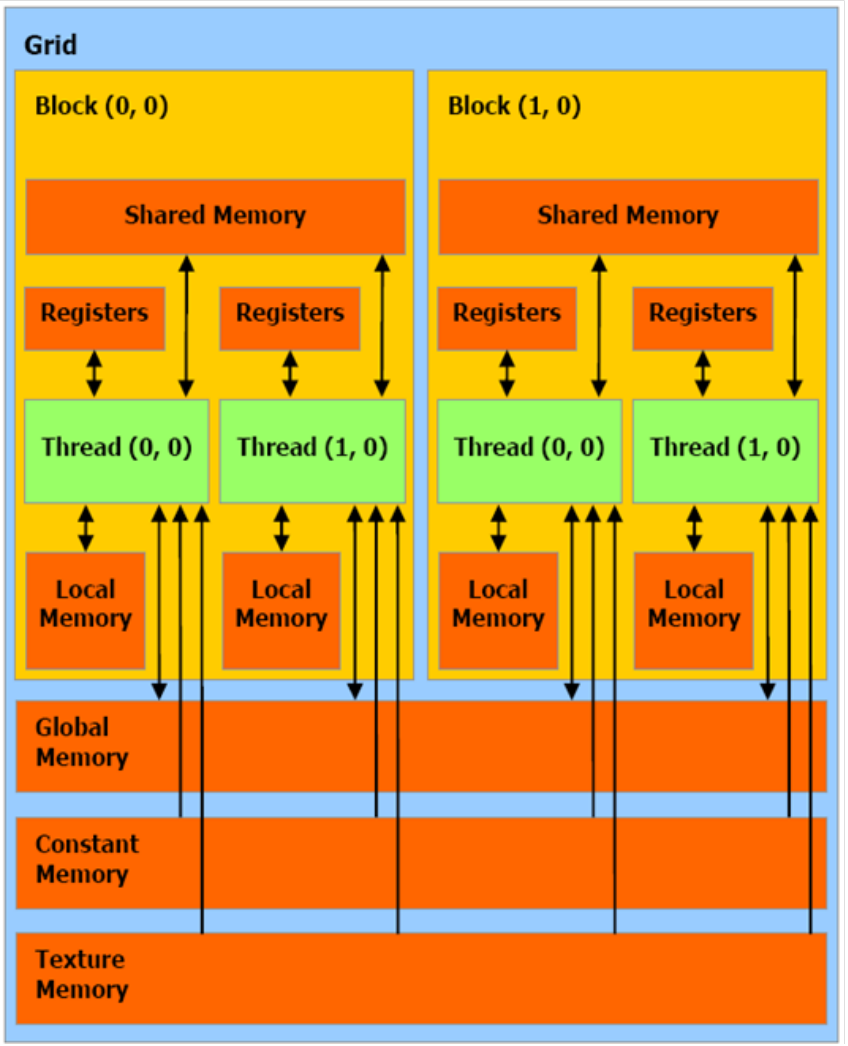


- **Arquitetura many-core**

- Unidades de Processamento Gráfico
  - **Nvidia** vs AMD
  - **CUDA** vs OpenCL vs OpenACC

Modelo de Programação - CUDA

- **Grid –** Conjunto de blocos independentes.
- **Block –** Conjunto de *threads* que podem se comunicar.
- **Warp –** 32 *threads* que executam a mesma instrução.
- Escalonamento de ***warps*** para os processadores.



Problema

Como escolher a dimensão do bloco visando otimizar a execução de aplicações em GPUs?

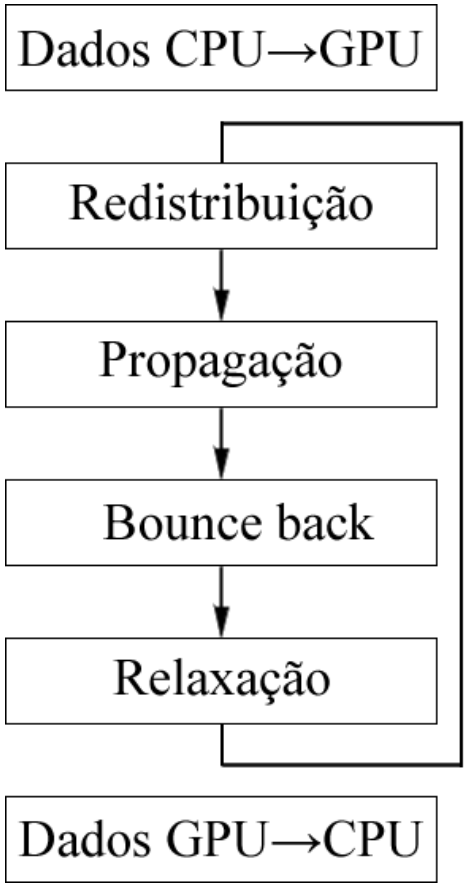
Como escolher a dimensão do bloco?

- Variar e testar manualmente.
- Modelo matemático.
- Fixar no tamanho máximo.
- Otimizar a ocupação da GPU.
  - CUDA 6.5 – Agosto de 2014.
  - cudaOccupancyMaxActiveBlocksPerMultiprocessor(funcao);

Metodologia

- *Benchmark*
  - Método de *Lattice Boltzmann.*
  - 128 x 128, 256 x 256, 512 x 512, 1024 x 1024.
  - Precisão Dupla.
- Métricas
  - *Speedup.*
  - Média de 30 execuções.
  - Desvio Padrão → 0s.
- Ambiente de Execução
  - *Processador Intel Xeon E5.*
  - GeForce GT 740M.
  - 384 CUDA Cores – 2 GB Memória Global.

Método de *Lattice Boltzmann*



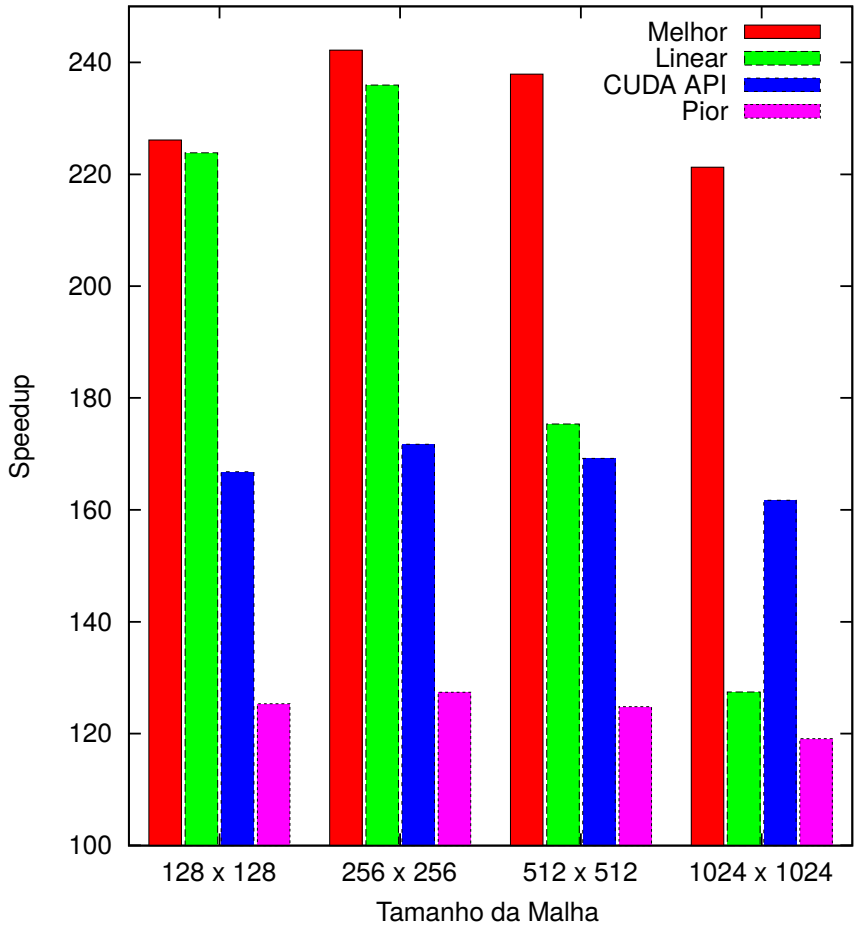
Laço Principal do MLB.

```
void relaxation(double *node, ...)
        for (unsigned int x = 0; x < lx; ++x)
                for(unsigned int y = 0; y < ly; ++y)
                        Operações sobre: node[x][y]

relaxation(node, ...);
```

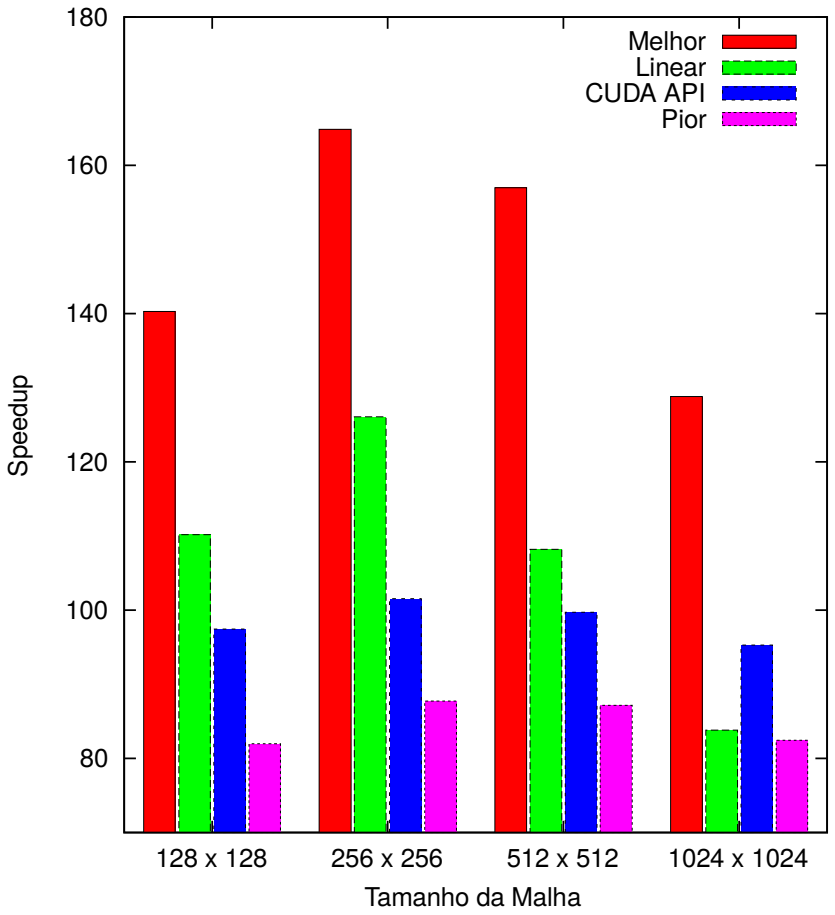
```
__global__ void relaxation(double *node, ...)
        unsigned int idx = blockIdx.x * blockDim.x + threadIdx.x;
        Operações sobre: node[idx]

relaxation<<<GRID_RELAXATION, BLOCK_RELAXATION>>>(d_node, ...);
```

Modificações da versão sequencial para versão CUDA.

Resultados



***Speedup* da função de Relaxação.**



***Speedup* do Método de *Lattice Boltzmann*.**

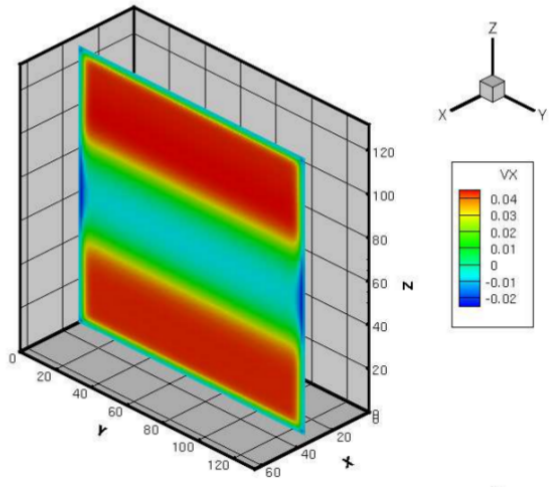
Conclusões

- Blocos menores [32, 96] foram os melhores.
  - Entretanto, a escolha é estática.
  - CUBLAS e CUFFT usam blocos de tamanho 64.
- Modelo Linear
  - Tamanho = dimensão y
  - Funciona até 256 x 256, mas não é escalável.
- API CUDA
  - Baseada em ocupação (Blocos de 640).
  - Tem a função como entrada.
- Tempo vs Desempenho
  - Desempenho: **Testar diversos blocos.**
  - Desenvolvimento rápido: **API CUDA.**

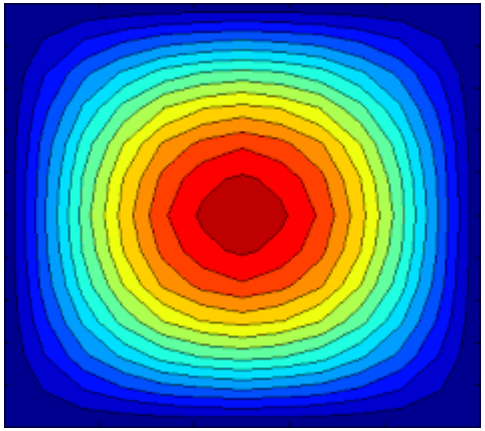
Trabalhos Futuros

- Balanceamento de Carga para Aplicações Híbridas.

- Desenvolver aplicações paralelas para arquiteturas híbridas (CPUs + GPUs).



Método de Lattice Boltzmann 3D.



Equação do Calor 2D.

Perguntas?

Matheus da Silva Serpa, Claudio Schepke
matheusserpa@alunos.unipampa.edu.br

LEA: Laboratório de Estudos Avançados em Computação
Ciência da Computação – Campus Alegrete

Trabalhos Relacionados

- Habich et al. 2013.
  - MLB D3Q19.
  - Pico de desempenho com 50% de ocupação..
- Toelke 2010.
  - MLB D2Q9.
  - Variaram de 32 até 256.
  - 1024 x 1024 o ideal foi de 64.
- Volkov 2010.
  - Técnicas de otimização com ocupação baixa.
  - CUBLAS e CUFFT usam blocos de tamanho 64.
- Diferencial
  - Utilizamos a **API CUDA**.
  - Dimensionamento automático.

Relaxation

Size	Sequencial	LY		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	49.11	0.22	223.23	128
256	197.74	0.84	235.41	256
512	793.58	4.53	175.18	512
1024	3179.32	24.95	127.43	1024

Relaxation

Size	Sequencial	Occupancy		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	49.11	0.29	169.34	640
256	197.74	1.15	171.95	640
512	793.58	4.69	169.21	640
1024	3179.32	19.66	161.72	640

Relaxation

Size	Sequencial	Best		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	49.11	0.22	223.23	64
256	197.74	0.82	241.15	96
512	793.58	3.34	237.6	96
1024	3179.32	14.37	221.25	96

Relaxation

Size	Sequencial	Worst		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	49.11	0.39	125.92	672
256	197.74	1.55	127.57	672
512	793.58	6.36	124.78	672
1024	3179.32	26.7	119.08	672

Redistribute

Size	Sequencial	LY		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	0.036	0.013	2.759	128
256	0.073	0.014	5.175	256
512	0.146	0.016	8.95	512
1024	0.302	0.022	13.728	1024

Redistribute

Size	Sequencial	Occupancy		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	0.036	0.013	2.759	128
256	0.073	0.014	5.175	256
512	0.146	0.016	8.95	512
1024	0.302	0.022	13.728	1024

Redistribute

Size	Sequencial	Best		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	0.036	0.013	2.793	64
256	0.073	0.013	5.41	128
512	0.146	0.015	9.659	128
1024	0.302	0.018	16.662	128

Redistribute

Size	Sequencial	Worst		
	Tempo (s)	Tempo (s)	Speedup	Threads
128	0.036	0.013	2.756	480
256	0.073	0.014	5.142	672
512	0.146	0.016	8.868	992
1024	0.302	0.022	13.728	1024