HPC aplicado à resolução da Equação de Sine-Gordon

Otaviano Cruz

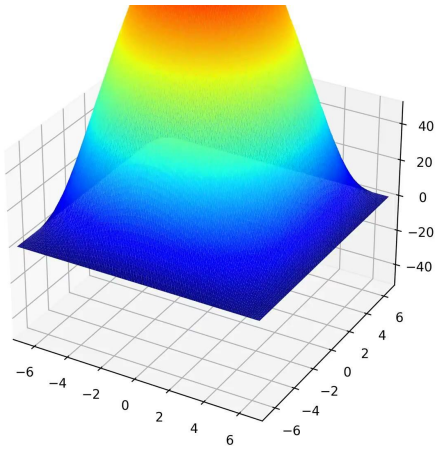
Sumário

- Equação de Sine-Gordon
- Código em serial
- Código em paralelo
- Conclusões

Equação de Sine-Gordon

Equação de Sine-Gordon

Equação de Sine-Gordon (SGE) , mais conhecida como a **Equação do Sóliton**, que descreve o movimento ondulatório acrescentando padrões dispersivos e não-linearidades.



$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = sen(u)$$

Condições iniciais e de contorno

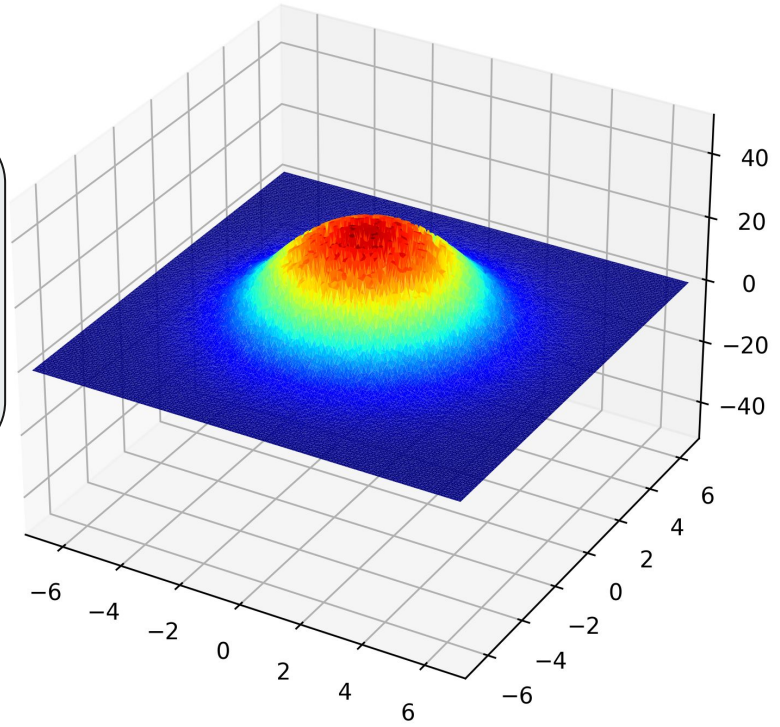
$$u(x, y, t = 0) = 4tan^{-1}(e^{3-\sqrt{x^2+y^2}})$$
$$\frac{\partial u}{\partial t}(x, y, t = 0) = 0$$

$$\frac{\partial u}{\partial x}(-x_0, y, t) = \frac{\partial u}{\partial x}(x_0, y, t) = \frac{\partial u}{\partial x}(x, -y_0, t) = \frac{\partial u}{\partial x}(x, y_0, t) = 0$$

Condição inicial

$$u(x, y, t = 0) = 4tan^{-1}(e^{3-\sqrt{x^2+y^2}})$$

$$\frac{\partial u}{\partial t}(x, y, t = 0) = 0$$



Diferenças finitas

Partindo da discretização da derivada :

$$\frac{du}{dx} = \frac{u(x+\Delta x)-u(x)}{\Delta x}$$

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = sen(u)$$

$$u_{m,l}^{n+1} \approx -u_{m,l}^{n-1} + (\frac{\Delta t}{\Delta x})^2 (u_{m+1,l}^{n} + u_{m-1,l}^{n} + u_{m,l+1}^{n} + u_{m,l-1}^{n+1})$$

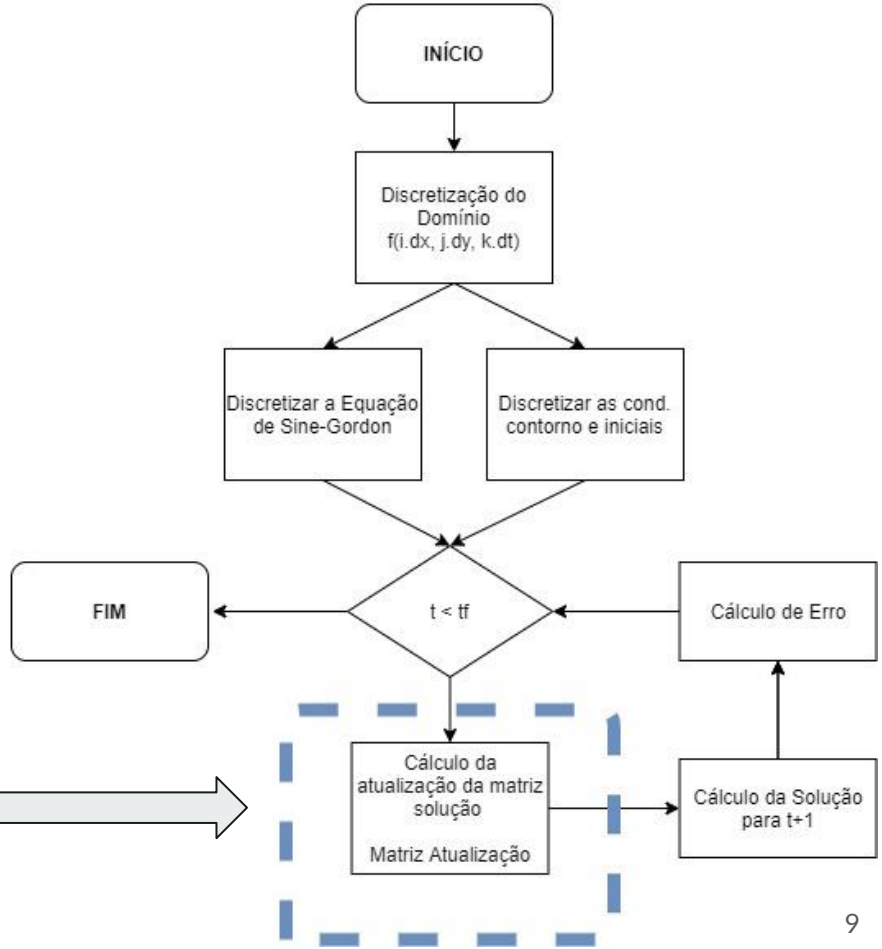
$$-\Delta t^2 sen(u_{m+1,l}^{n} + u_{m-1,l}^{n} + u_{m,l+1}^{n} + u_{m,l-1}^{n+1}) + 2[1 - 2(\frac{\Delta t}{\Delta x})^2]u_{m,l}^{n}$$

Código em serial

Código em serial

- Organização do programa
- Profile

92% do gasto computacional do programa está no cálculo da matriz solução.

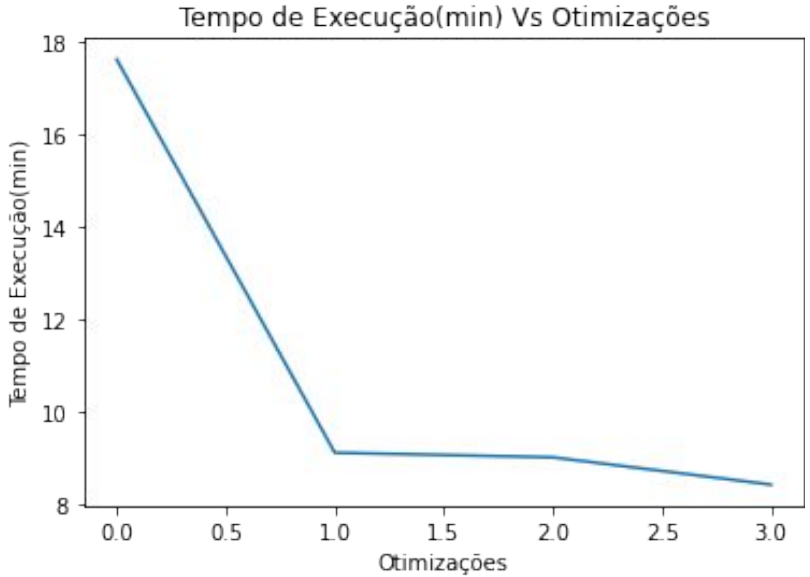


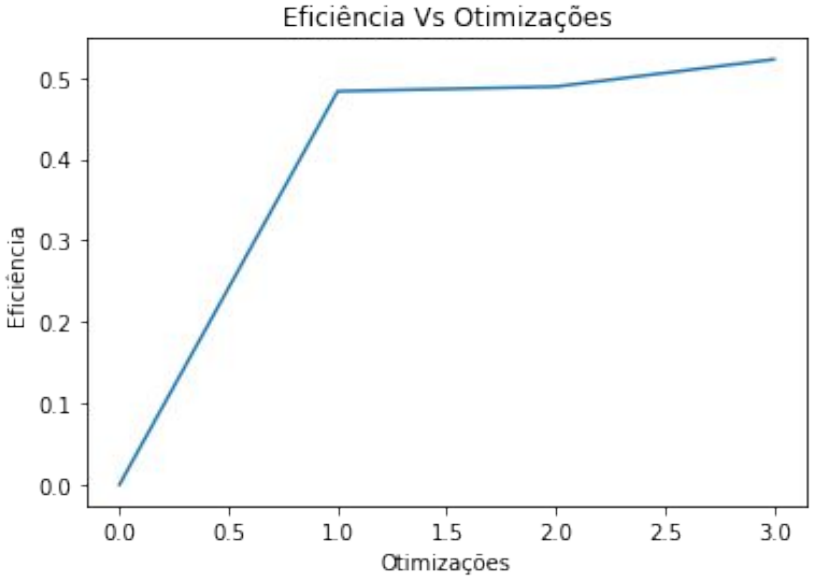
Otimização a nível de compilação

- O’s
- fexpensive-optimizations
- foptimize-register-move
- funroll-loops
- ffast-math
- mavx
- mtune=native

Tamanho da Malha : 10000
Domínio : 50
Iterações Temporais = 300

Tempo de Execução e Performance





Código em paralelo

OpenMP

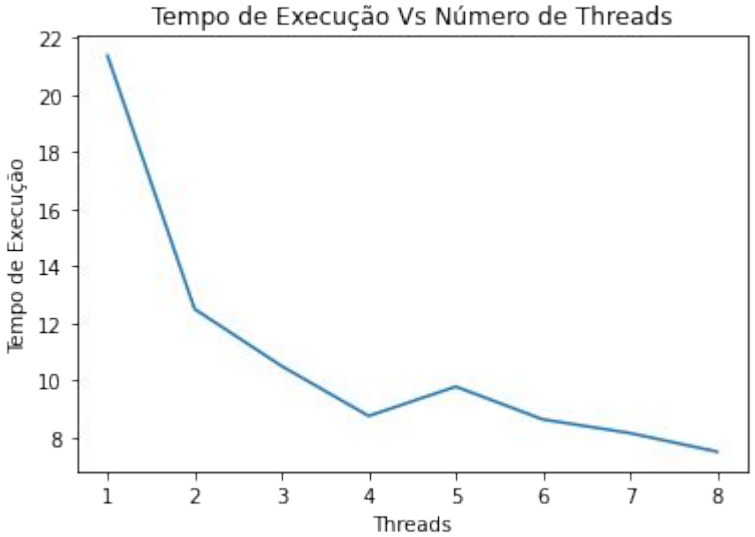
Código em paralelo (OpenMP)

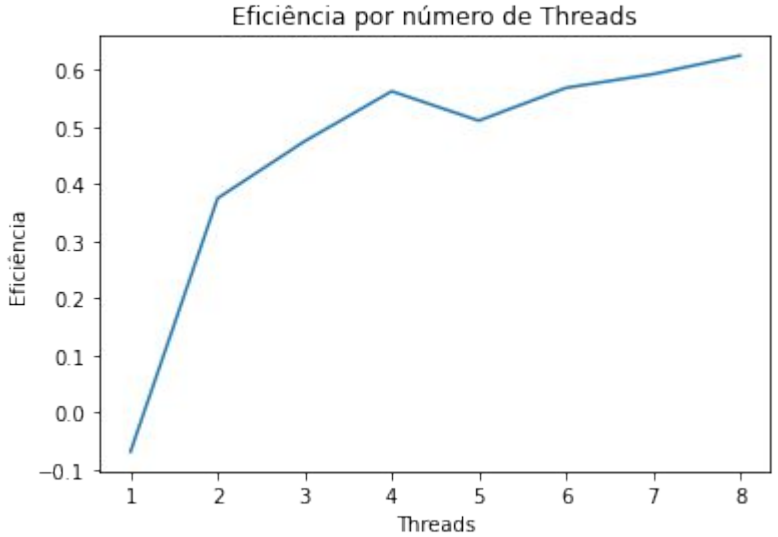
- API OpenMP
- Diretivas : Private, Shared, threadprivate e copyin.

As diretivas copyin e threadprivate são responsáveis pelo aumento substancial do uso de memória pelo programa.

Erro médio em relação ao código em serial de 1%

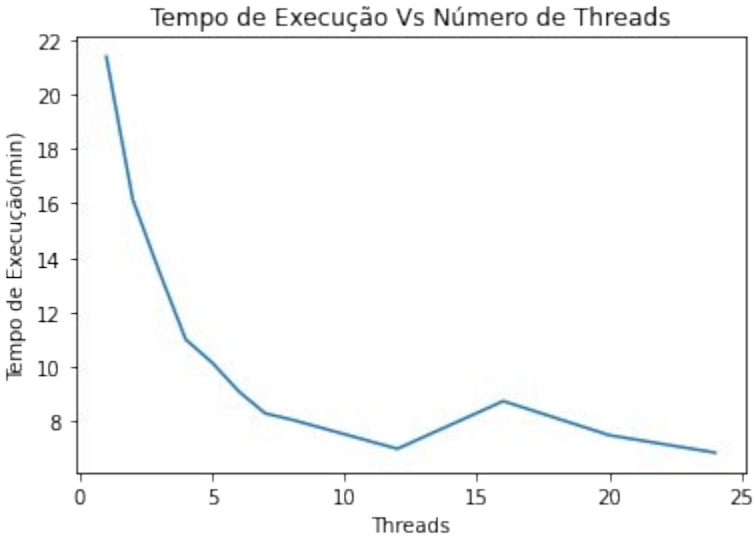
Tempo de Execução e Performance

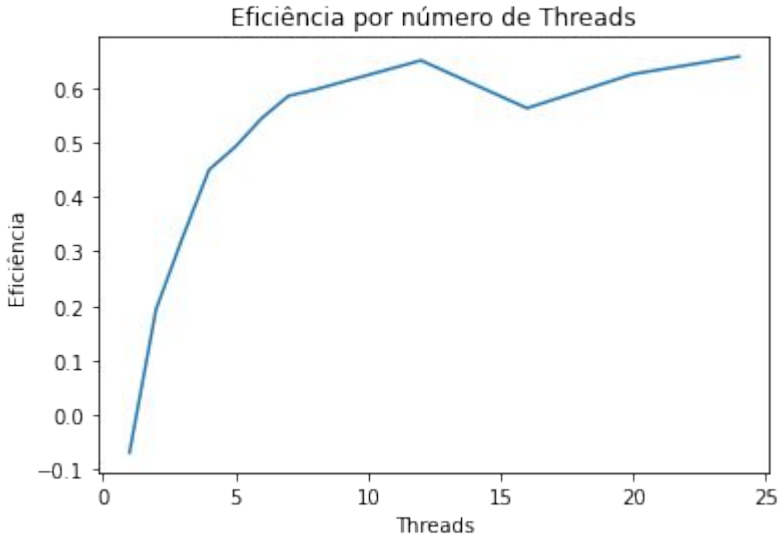




Core i7-2600K (Cluster)

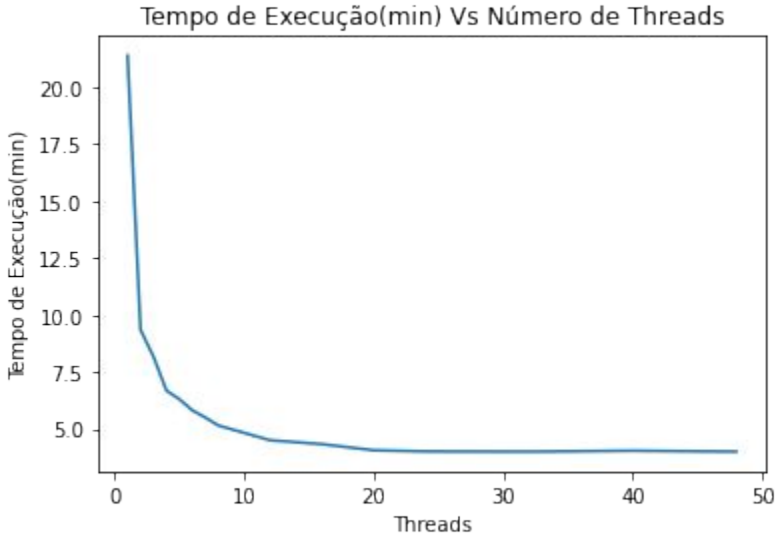
Tempo de Execução e Performance

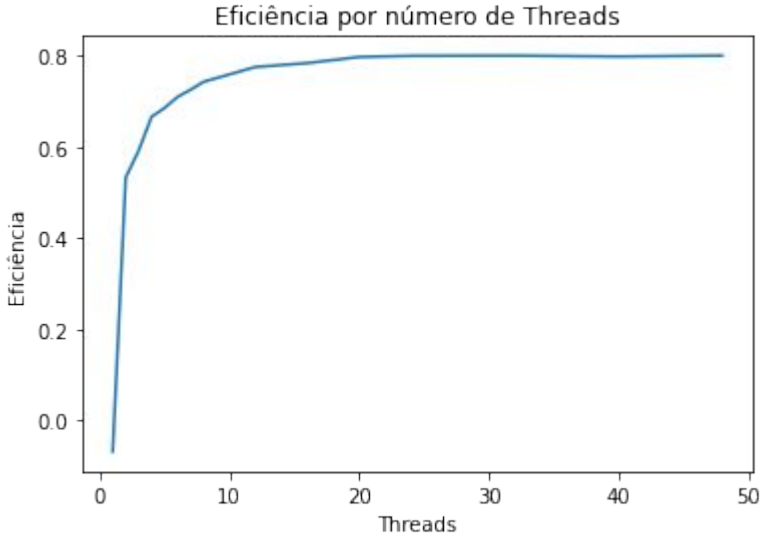




Xeon Ivy Bridge E5-2695v2 (BullX)

Tempo de Execução e Performance





Xeon Cascade Lake Gold 6252 (Sequana)

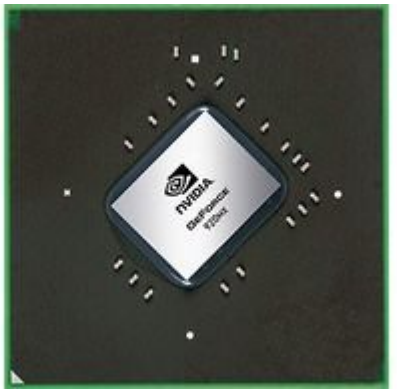
CUDA

Código em paralelo (CUDA)

- Cálculos rápido realizados na GPGPU
- Transferência de informações entre CPU e GPGPU.
- Modelos de placas de vídeo.

Tesla K40

RTX 2060



920MX

CUDA Cores e Performance

Modelo	N° CUDA Cores	Perf. Single Precision	Perf. Double Precision
920MX (2015)	384	508.4 GFLOPS	15.89 GFLOPS
GTX 1660 (2019)	1408	5.027 TFLOPS	157.1 GFLOPS
RTX 2060 (2019)	1920	6.451 TFLOPS	201.6 GFLOPS
RTX 2060 Super (2019)	2176	7.181 TFLOPS	224.4 GFLOPS
K40 (2013)	2880	4.29 TFLOPS	1.43 TFLOPS
Volta V100 (2017)	5120	14 TFLOPS	7 TFLOPS

Uso de memória para alocação de uma matriz 300 x 300

Modelo	Sing. Prec.	Double Prec.	Mem. Total
920MX	1314 MB	2348 MB	4 GB
K40	1296 MB	2257 MB	12 GB
RTX 2060	1304 MB	2336 MB	6 GB

Código em paralelo (CUDA)

Tamanho da Malha : 10000
Domínio : 50
Iterações Temporais = 300

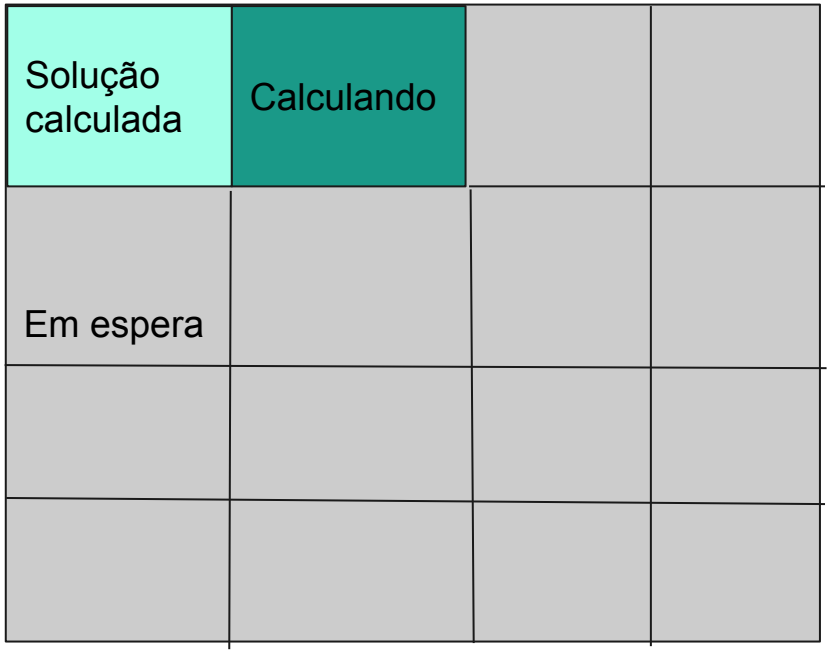
Dimensões das threads disponíveis para o uso.

Malha do Problema

Código em paralelo (CUDA)

Tamanho da Malha : 10000
Domínio : 50
Iterações Temporais = 300

Dimensões das threads disponíveis para o uso.



Profile

```
sudo nsys profile ./EXECUTAVEL.x
sudo nsys stats report1.qdrep > relatorio_profile.dat
```

Time(%)	Total Time (ns)	Num Calls	Average	Minimum	Maximum	Name
99.1	22,280,796,119	1,202	18,536,436.0	9,728	33,332,223	cudaMemcpy
0.9	197,883,223	5	39,576,644.6	3,039	197,438,722	cudaMalloc
0.0	648,064	5	129,612.8	5,649	271,058	cudaFree
0.0	123,921	300	413.1	209	3,062	cudaLaunchKernel

Time(%)	Total Time (ns)	Operations	Average	Minimum	Maximum	Operation
74.1	16,436,181,303	902	18,221,930.5	1,280	24,014,596	[CUDA memcpy HtoD]
25.9	5,736,267,902	300	19,120,893.0	18,421,562	33,013,634	[CUDA memcpy DtoH]

Profile

```
sudo nsys profile ./EXECUTAVEL.x
sudo nsys stats report1.qdrep > relatorio_profile.dat
```

Time(%)	Total Time (ns)	Num Calls	Average	Minimum	Maximum	Name
99.1	22,280,796,119	1,202	18,536,436.0	9,728	33,332,223	cudaMemcpy
0.9	197,883,223	5	39,576,644.6	3,039	197,438,722	cudaMalloc
0.0	648,064	5	129,612.8	5,649	271,058	cudaFree
0.0	123,921	300	413.1	209	3,062	cudaLaunchKernel

Time(%)	Total Time (ns)	Operations	Average	Minimum	Maximum	Operation
74.1	16,436,181,303	902	18,221,930.5	1,280	24,014,596	[CUDA memcpy HtoD]
25.9	5,736,267,902	300	19,120,893.0	18,421,562	33,013,634	[CUDA memcpy DtoH]

Otimizações

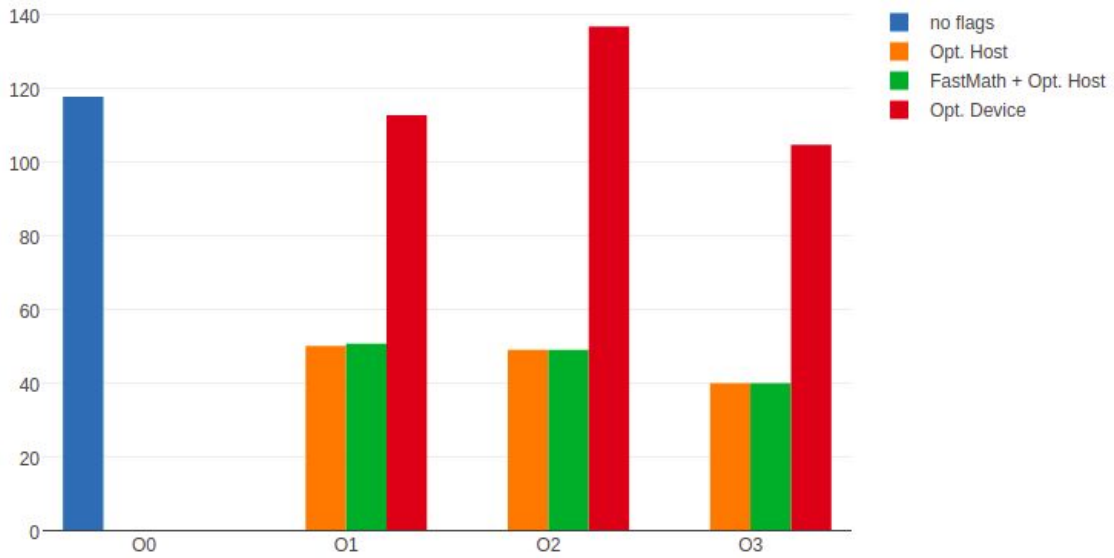
CPU

- O’s
- use_fast_math

GPGPU

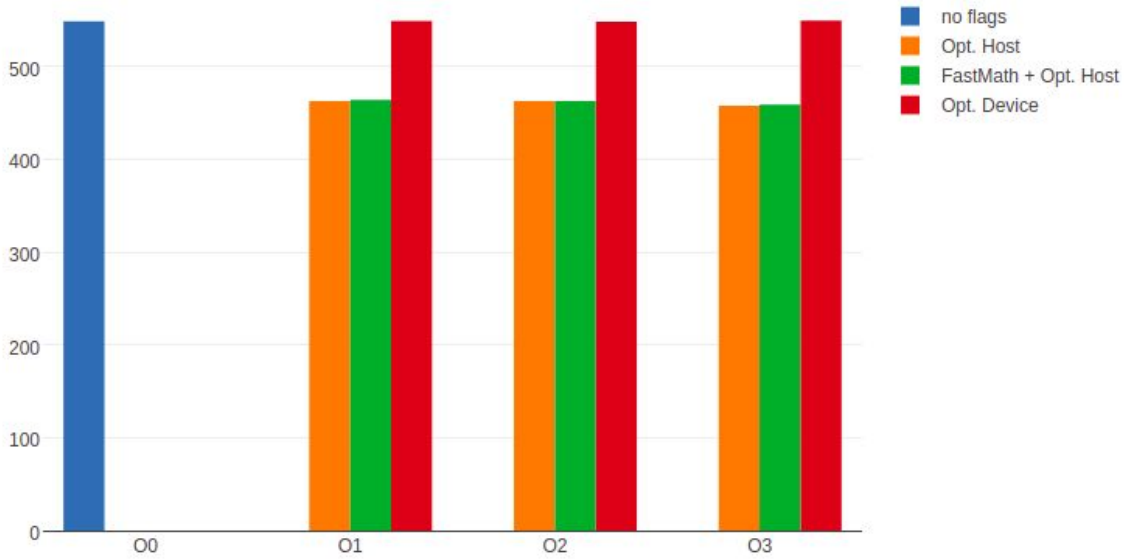
- O’s

Tempo de Execução e Performance



920 MX

Tempo de Execução e Performance

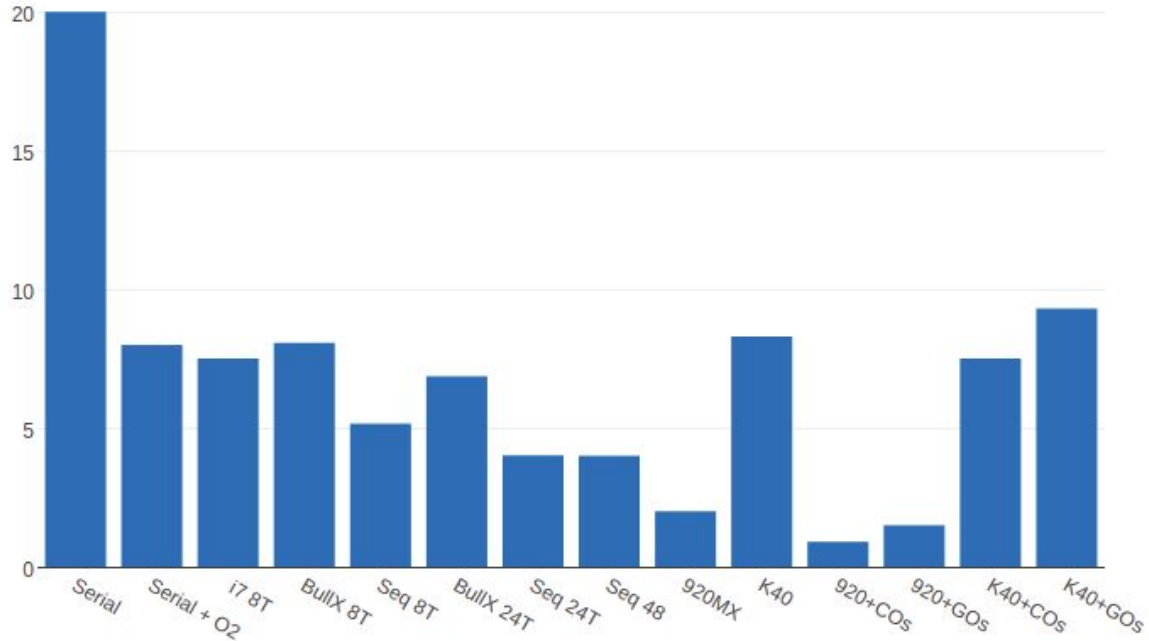


Tesla K40

Comparativos

Erro paralelo CPU : 1%

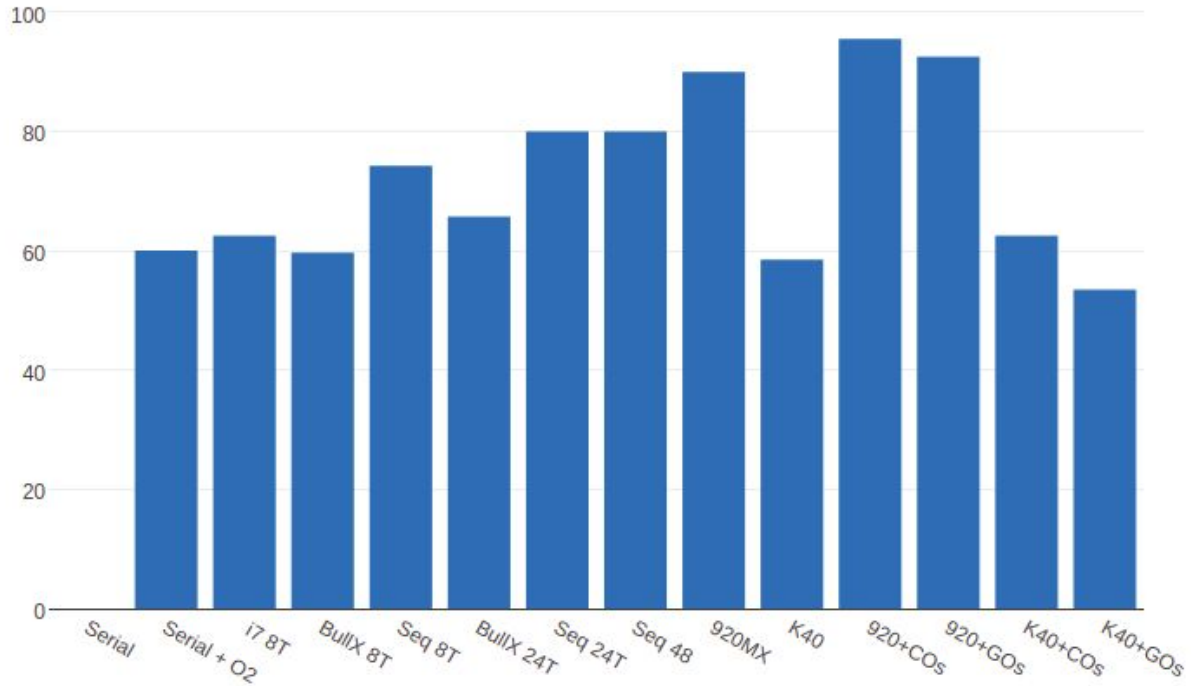
Erro paralelo GPGPU : 0.95%



Comparativos

Erro paralelo CPU : 1%

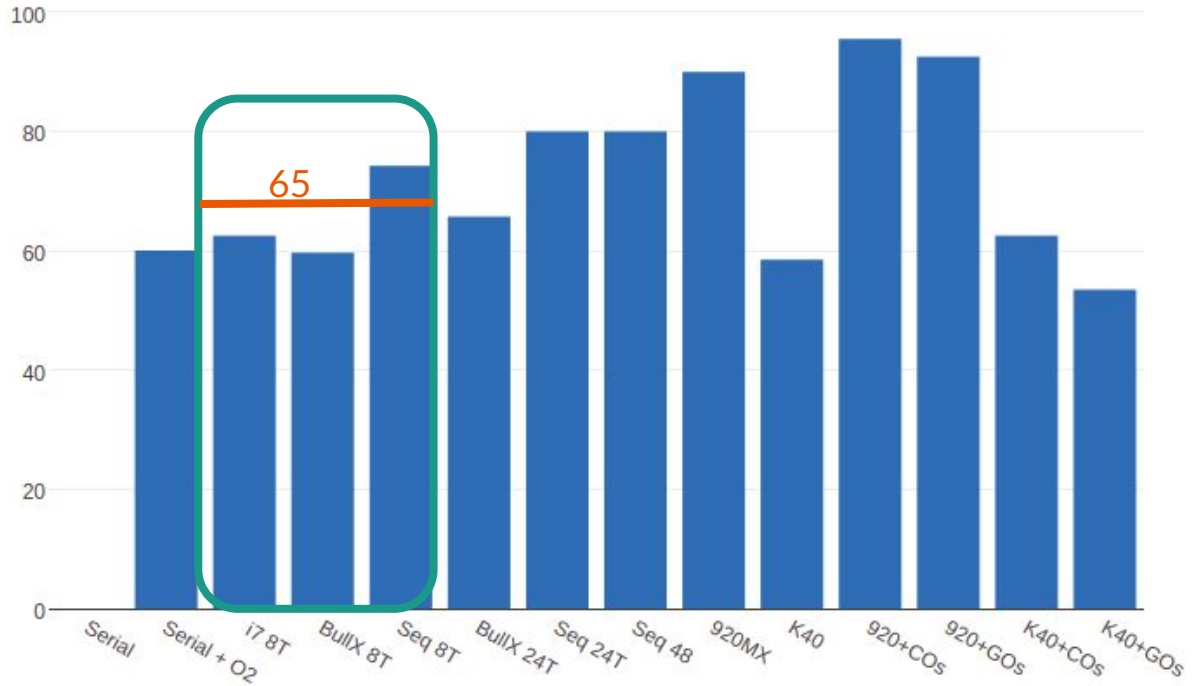
Erro paralelo GPGPU : 0.95%



Comparativos

Erro paralelo CPU : 1%

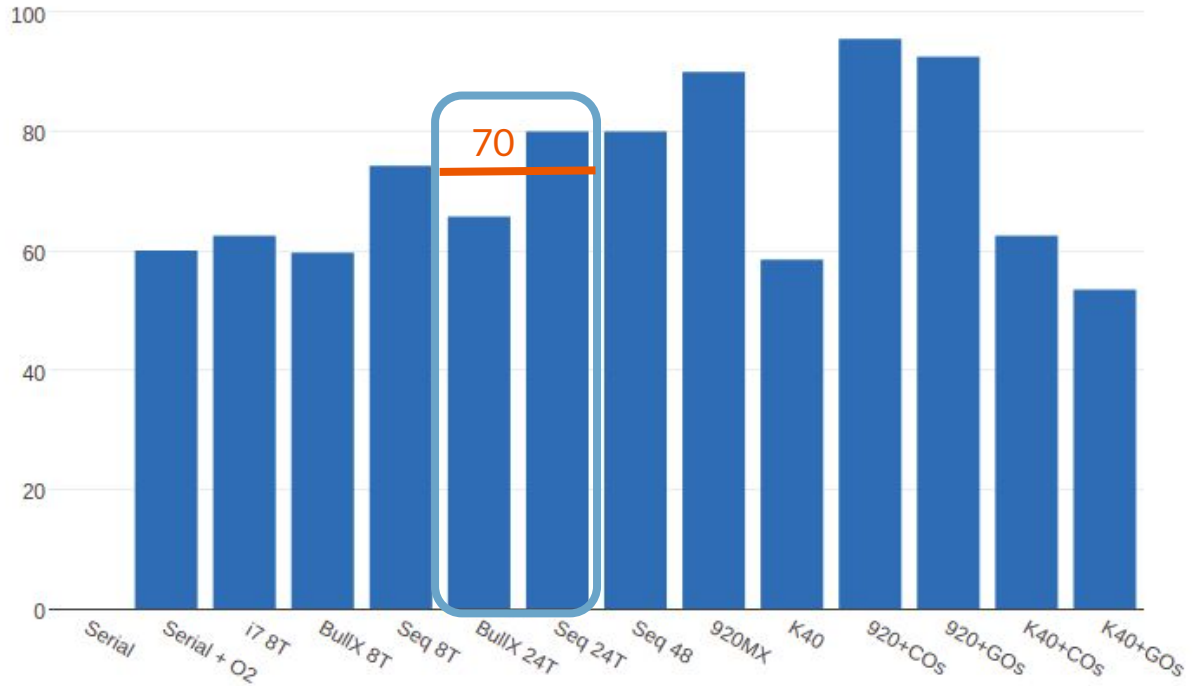
Erro paralelo GPGPU : 0.95%



Comparativos

Erro paralelo CPU : 1%

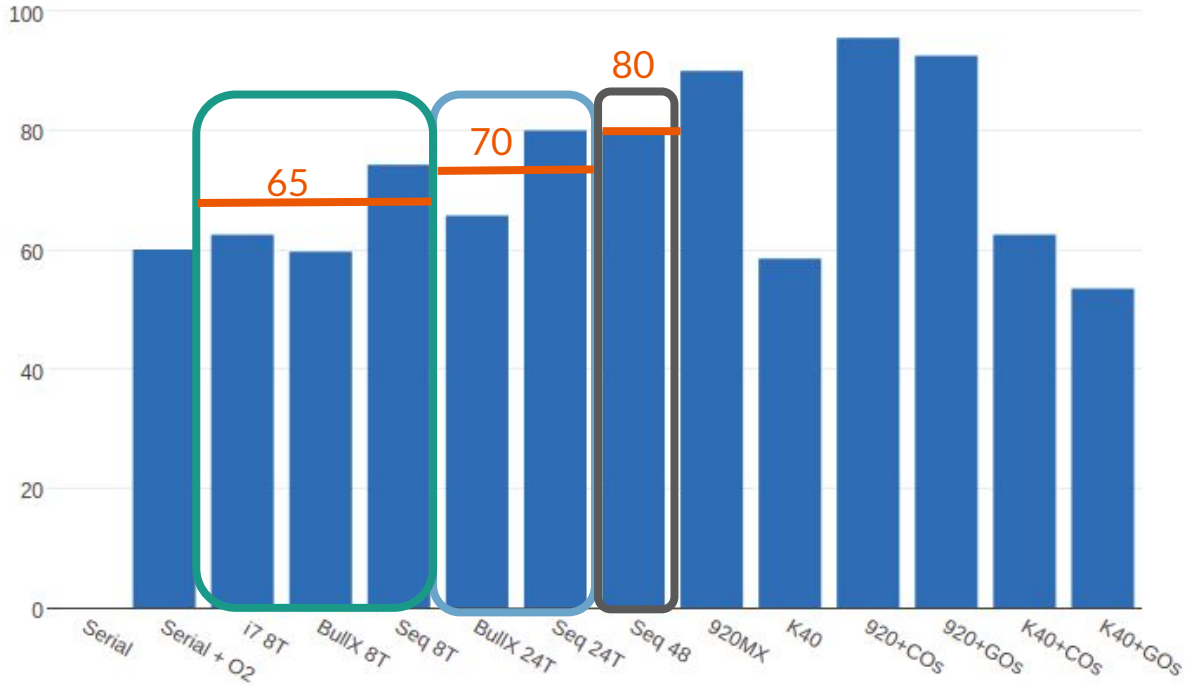
Erro paralelo GPGPU : 0.95%



Comparativos

Erro paralelo CPU : 1%

Erro paralelo GPGPU : 0.95%



Conclusões

Conclusões

- Otimização a nível de compilador foram mais eficientes em código serial.
- Otimização via compilador na GPGPU não foram expressivos.
- 8 threads trouxe a melhor eficiência com pouco recurso computacional.
- Acima de 8 threads foram obtidas reduções pouco significativas no tempo de execução.
- Paralelismo em CPU trouxeram um erro de 1%.
- Paralelismo em GPGPU trouxeram erro de 0.95% em dupla precisão.
- Gargalo na transferência de informações entre CPU e GPGPU.
- Performance penalizada no caso da GPGPU compartilhada.