Computação Paralela em GPU Usando CUDA

Ricardo Farias
Programa de Engenharia de Sistemas e Computação
COPPE / UFRJ

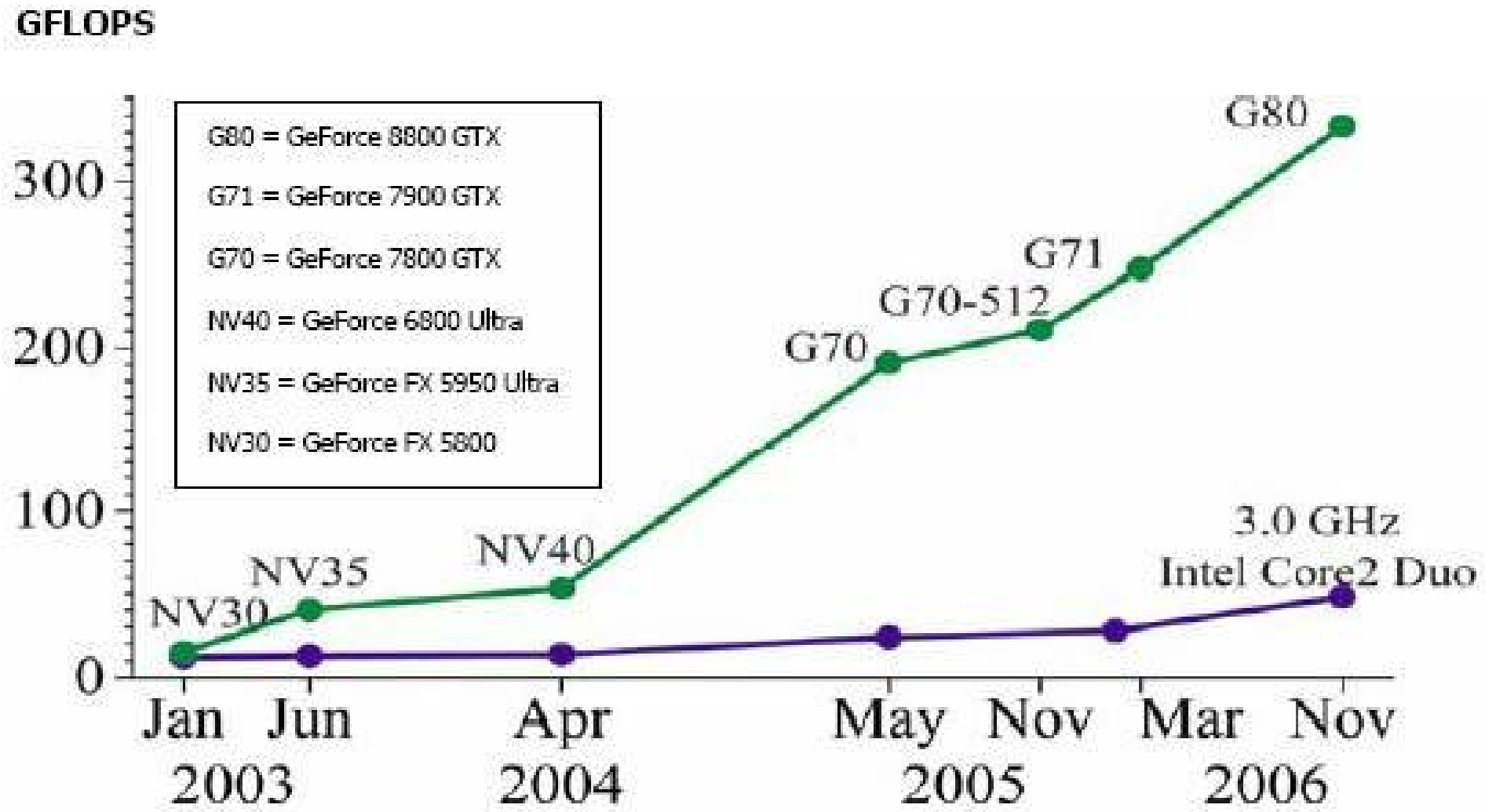
rfarias@cos.ufrj.br

2011

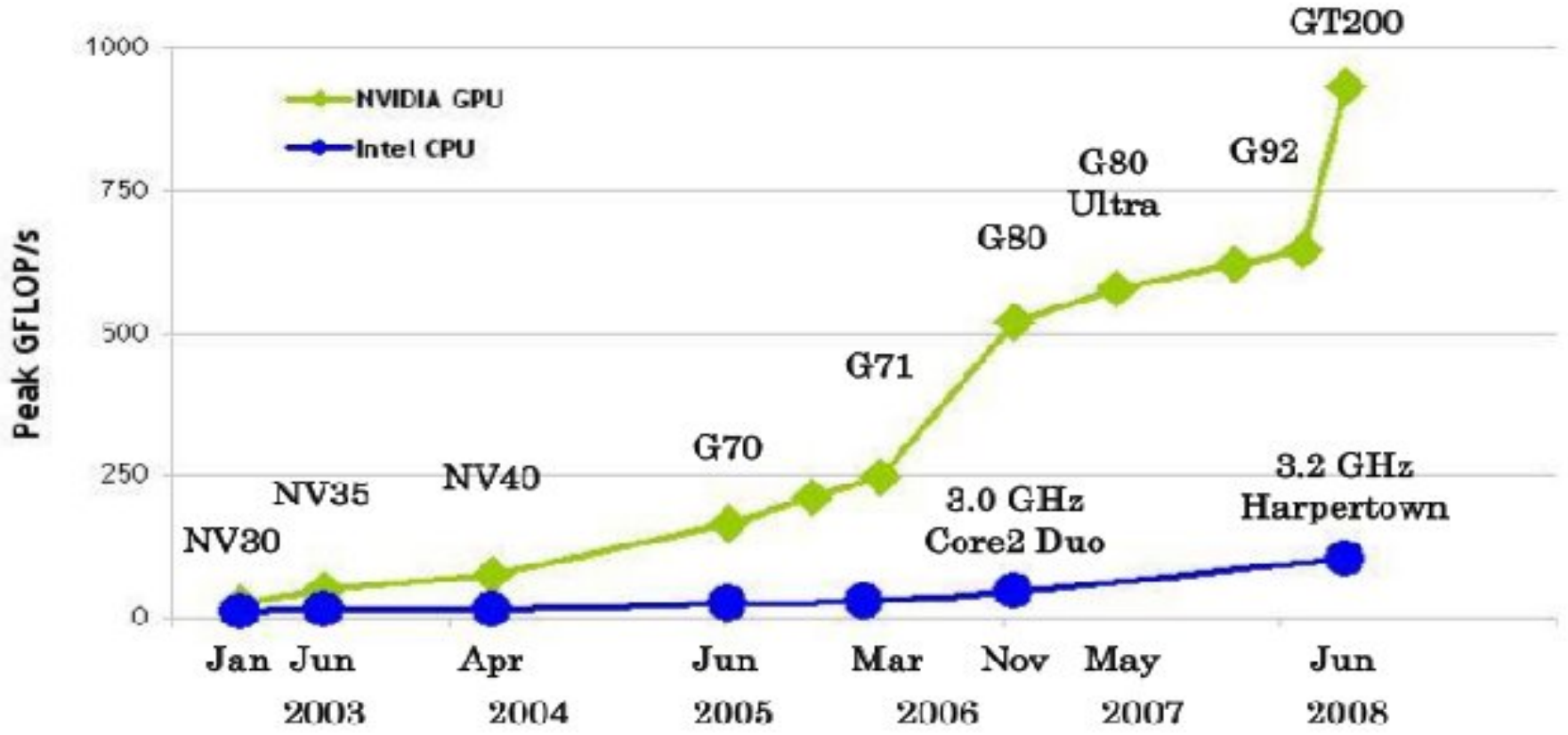
Motivação

- O que é GPGPU?
  - General-Purpose Computing on a Graphics Processing Unit (GPU)
  - Usando Hardware gráfico para computação não-gráfica
- O que é CUDA?
  - Compute Unified Device Architecture
  - Arquitetura de Software para gerenciar programação paralela orientada por dados

Motivação


GFLOPS
G80 = GeForce 8800 GTX
G71 = GeForce 7900 GTX
G70 = GeForce 7800 GTX
NV40 = GeForce 6800 Ultra
NV35 = GeForce FX 5950 Ultra
NV30 = GeForce FX 5800
300
200
100
0
G80
G71
G70-512
G70
NV40
NV35
NV30
3.0 GHz
Intel Core2 Duo
Jan
Jun
Apr
May
Nov
Mar
Nov
2003
2004
2005
2006

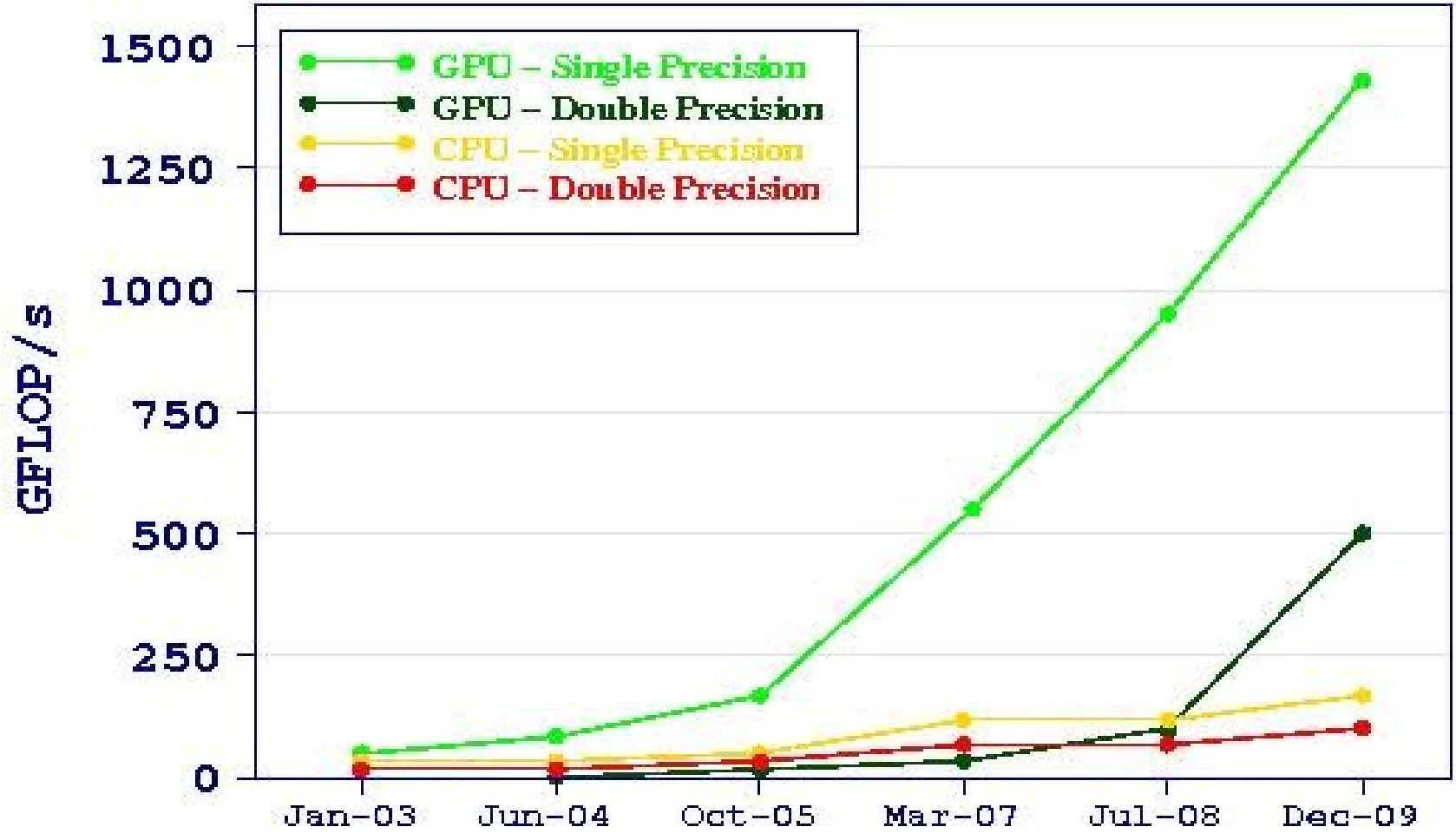
Motivação



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

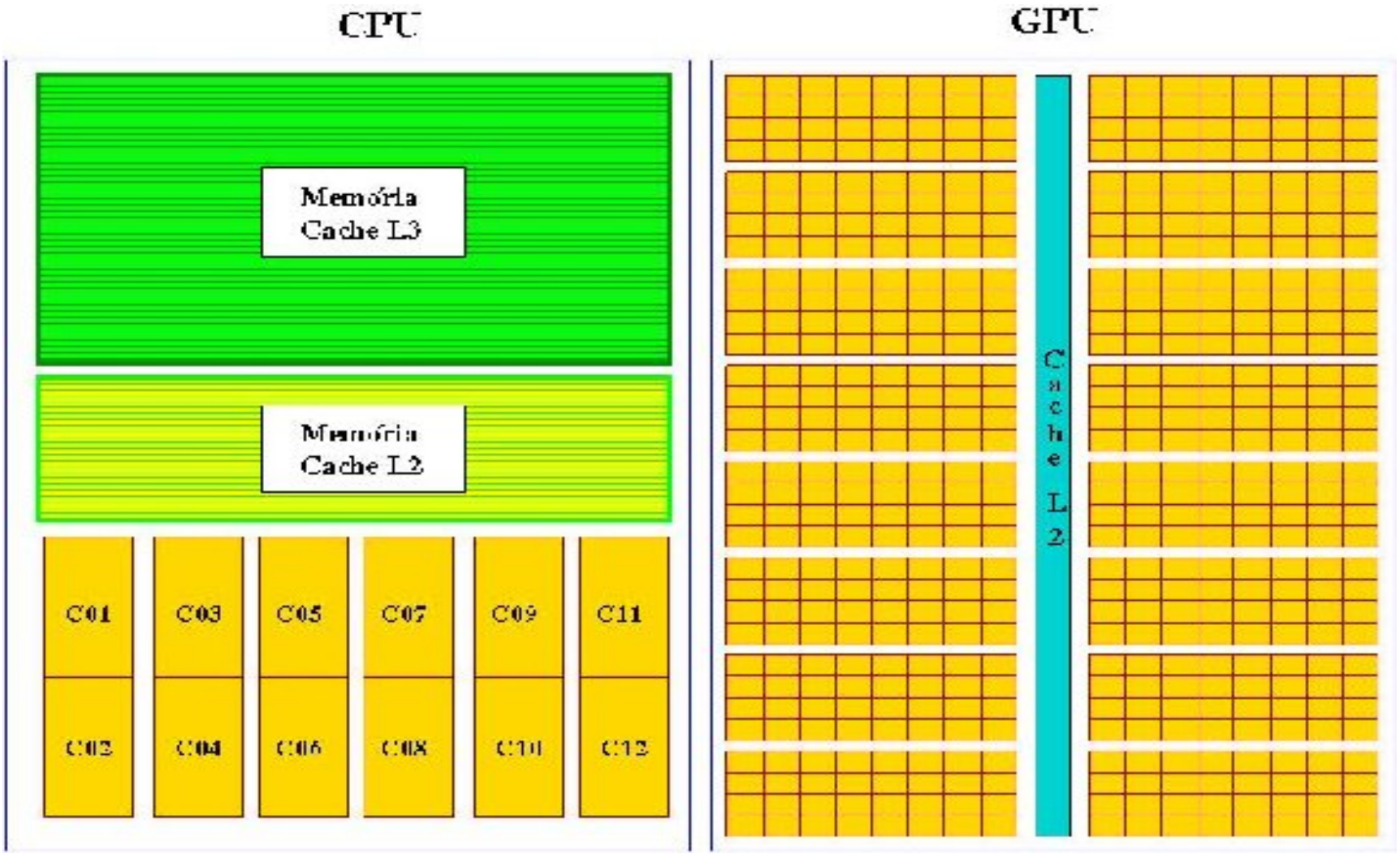
Motivação

- GPU Fermi Tesla
- CPU AMD Opteron 12 cores



Comparação das CPUs/GPUs

- Mais transistores dedicados para processamento de dados



CPU x GPU

- **CPU**
  - Quantidade de Caches (L1, L2, L3)
  - Previsão de Salto
  - Alta-performance (pelas previsões)
- **GPU**
  - Muitas ALUs
  - Memória onboard Rápida
  - Grande quantidade de tarefas paralelas
    - Executa programas em cada fragmento/vertice

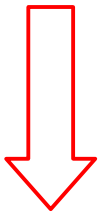
- **GPU**s são indicadas para paralelismo de dados

Arquitetura Fermi

- 3 Bilhões de transistores
- 4 GPC (Grap.Proc.Cluster)
- 16 SM
- 32 Elementos/SM
- 512 Cores
- 6 GB DDR5
- 64 bits float
- 32K Registradores/SM
- 64K Shared L1
- 768K Cache L2

Introdução à Programação CUDA

C U D A



Computer Unified Device Architecture

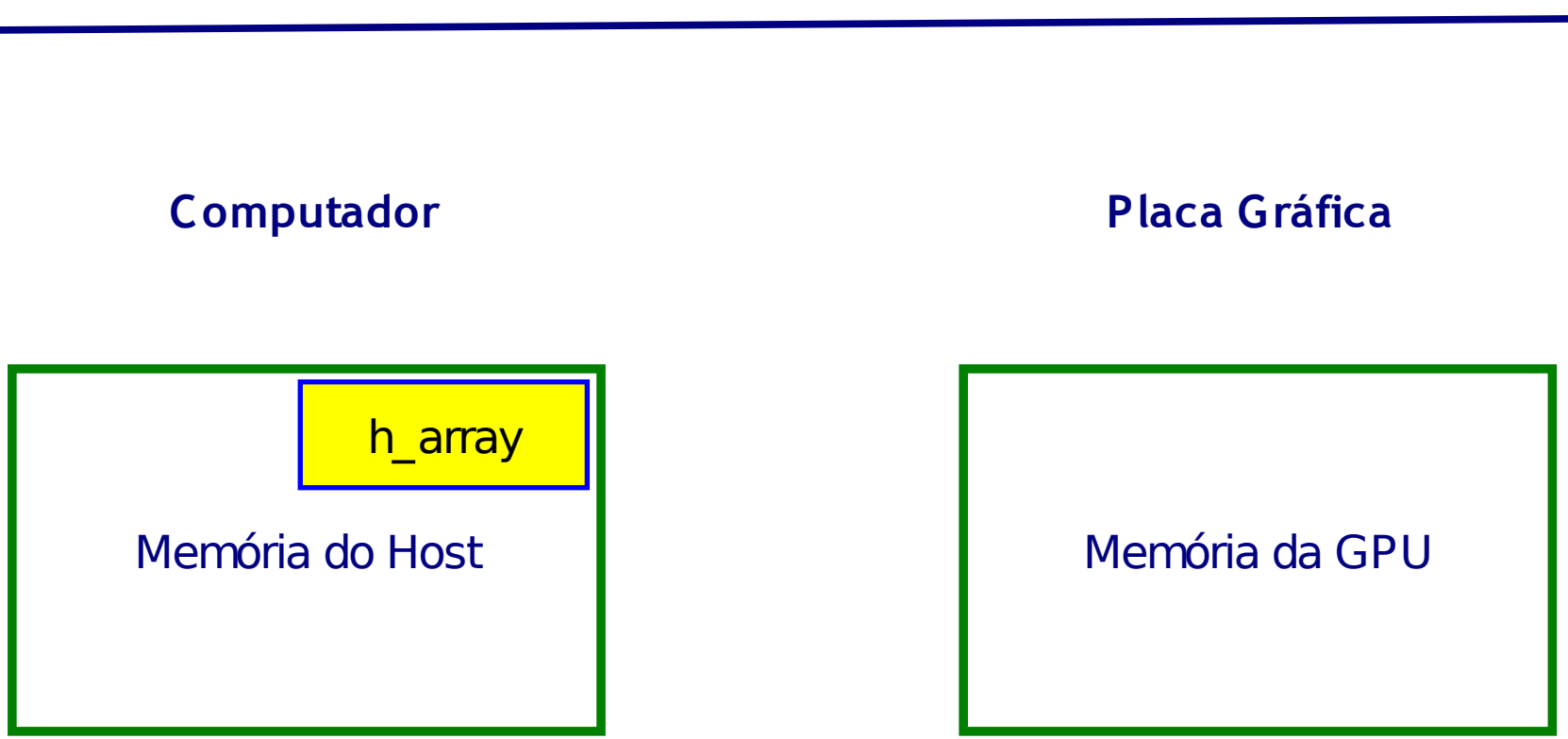
Programação CUDA

Um programa da GPU chama-se ***kernel***.

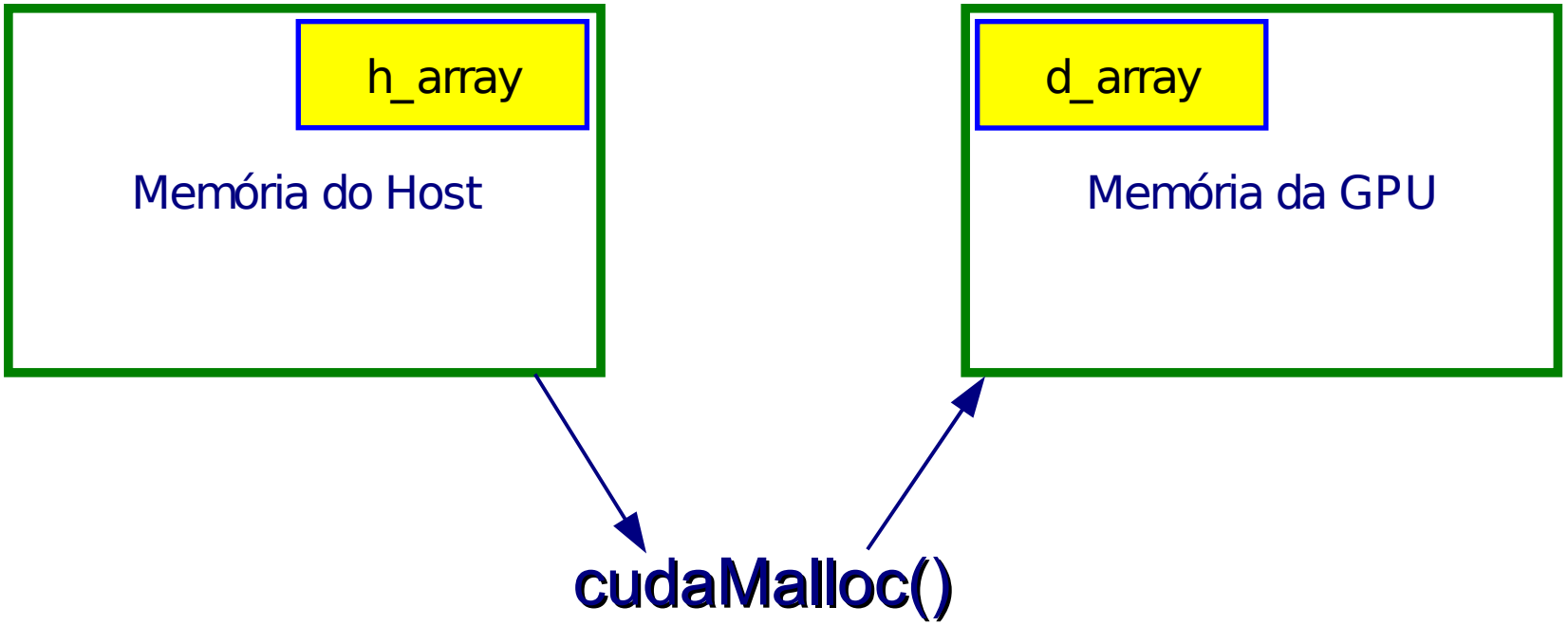
A Programação de um ***kernel*** exige:

- Reservar espaço na memória da placa gráfica.
- Copiar dados para a memória da placa.
- Chamar o código a ser executado na GPU.
- Copiar os resultados de volta da GPU.

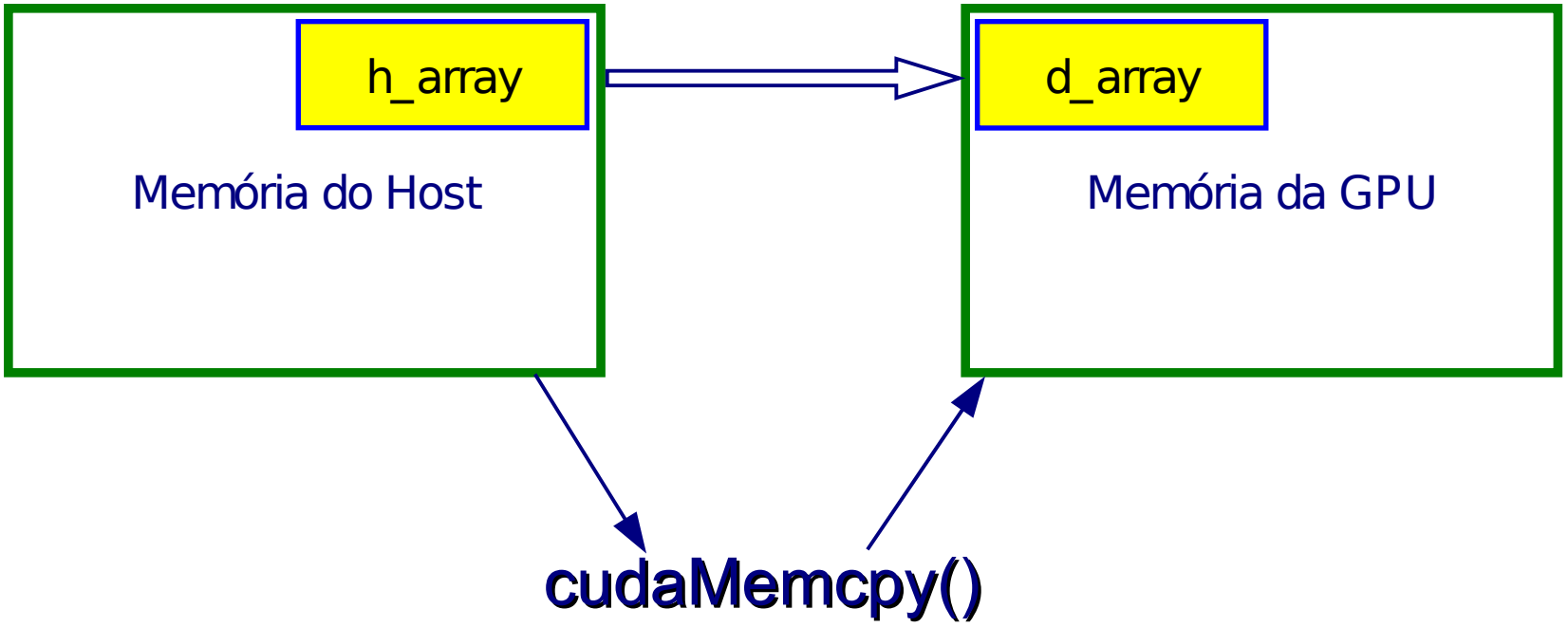
Passos de um Kernel



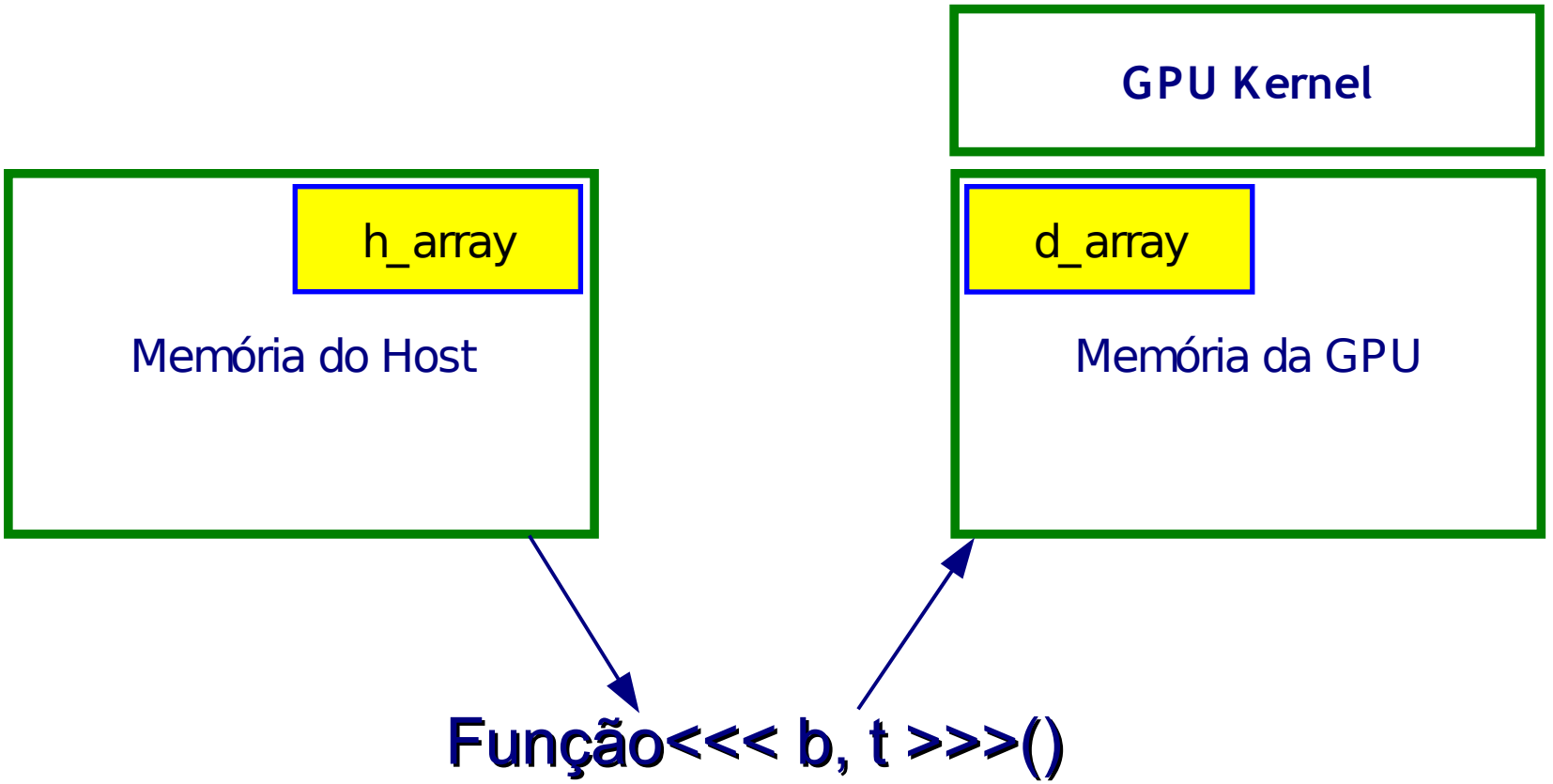
Passos de um Kernel



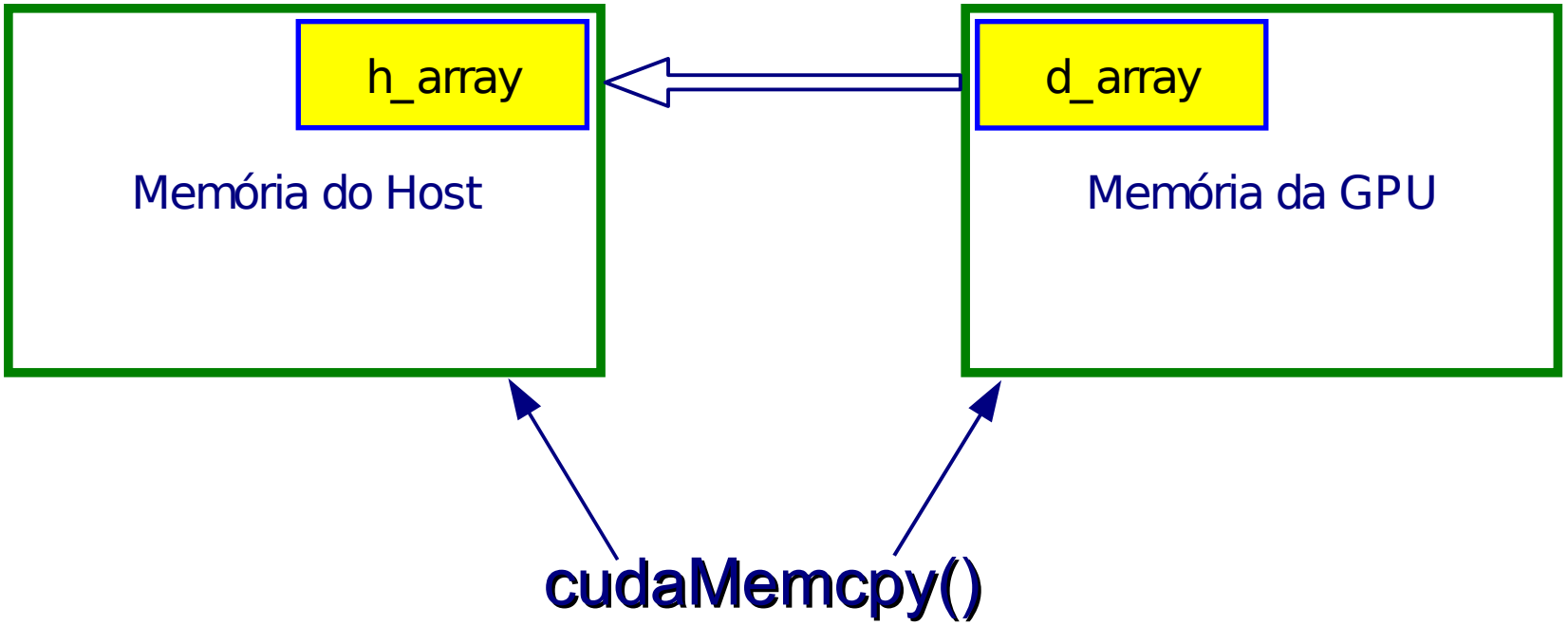
Passos de um Kernel



Passos de um Kernel



Passos de um Kernel



Diretivas CUDA

Tipos de Função e Diretivas de Compilação CUDA:

- __host__ Função executa na CPU
- __global__ Função executa na GPU, chamada pela CPU.
- __device__ Executa na GPU, chamada por outra função GPU

Diretivas CUDA

Tipos de Função e Diretivas de Compilação CUDA:

- __host__ Função executa na CPU
- __global__ Função executa na GPU, chamada pela CPU.
- __device__ Executa na GPU, chamada por outra função GPU

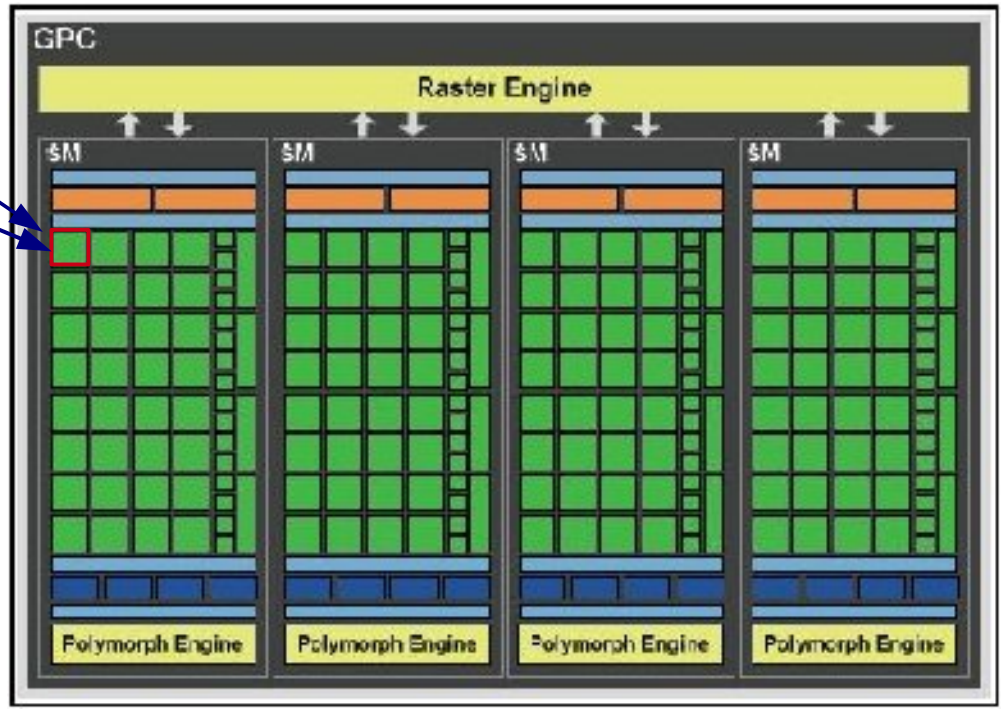
Como executar um kernel?

função<<< numero de blocos, quantidade de threads >>>(argumentos)

Entendendo Blocos e Threads

func<<< 1, 1 >>>( )

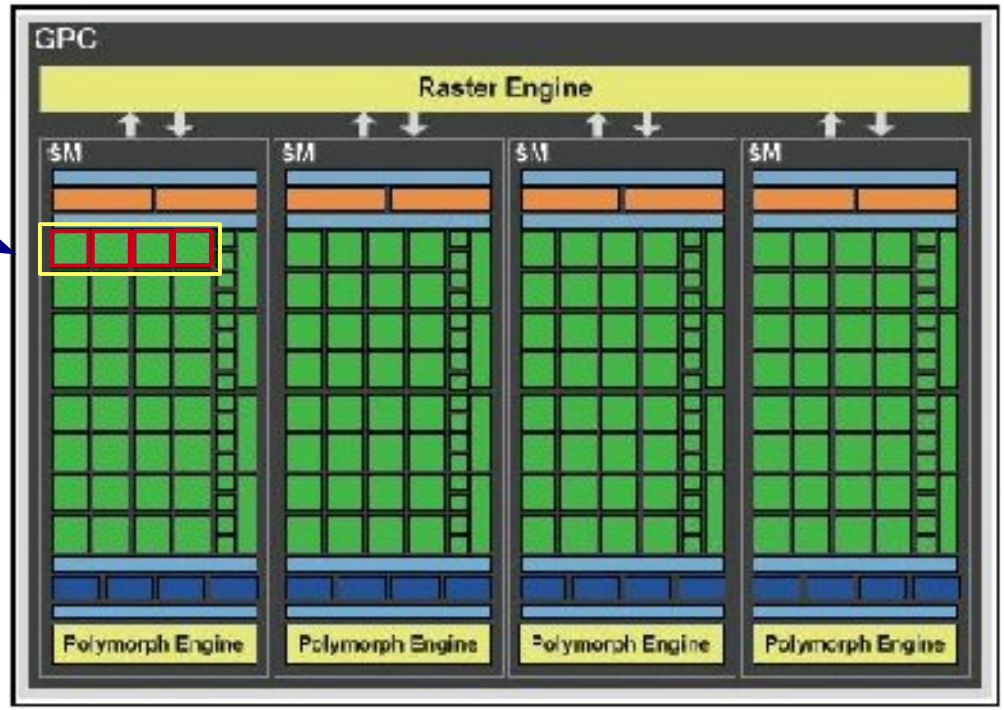
bloco thread



Entendendo Blocos e Threads

func<<< 1, 4 >>>( )

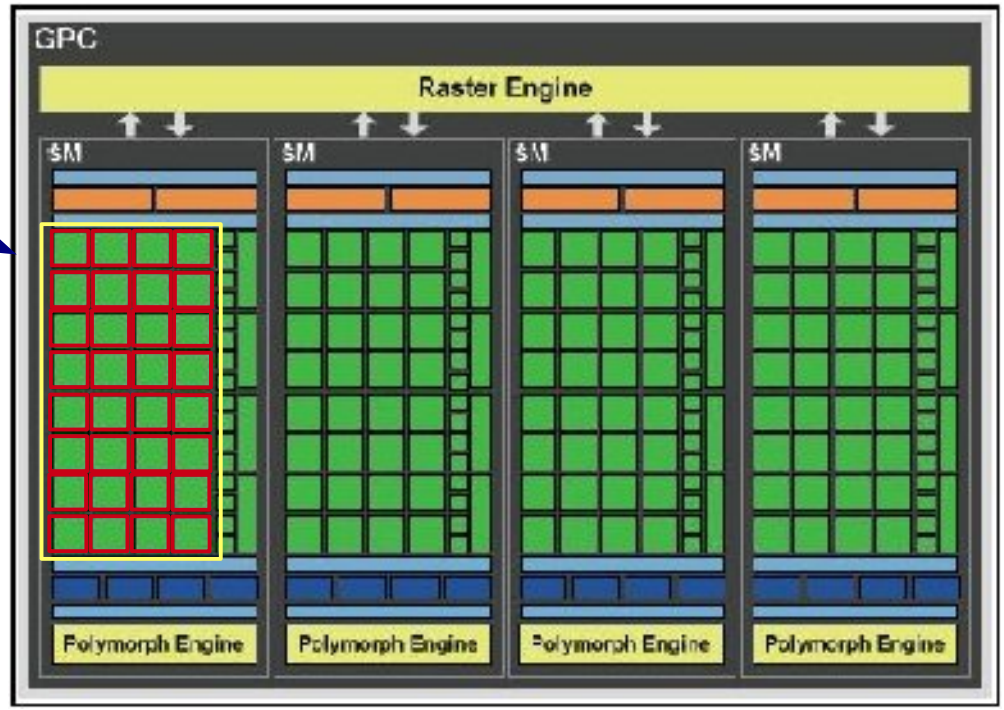
bloco threads



Entendendo Blocos e Threads

func<<< 1, 32 >>>( )

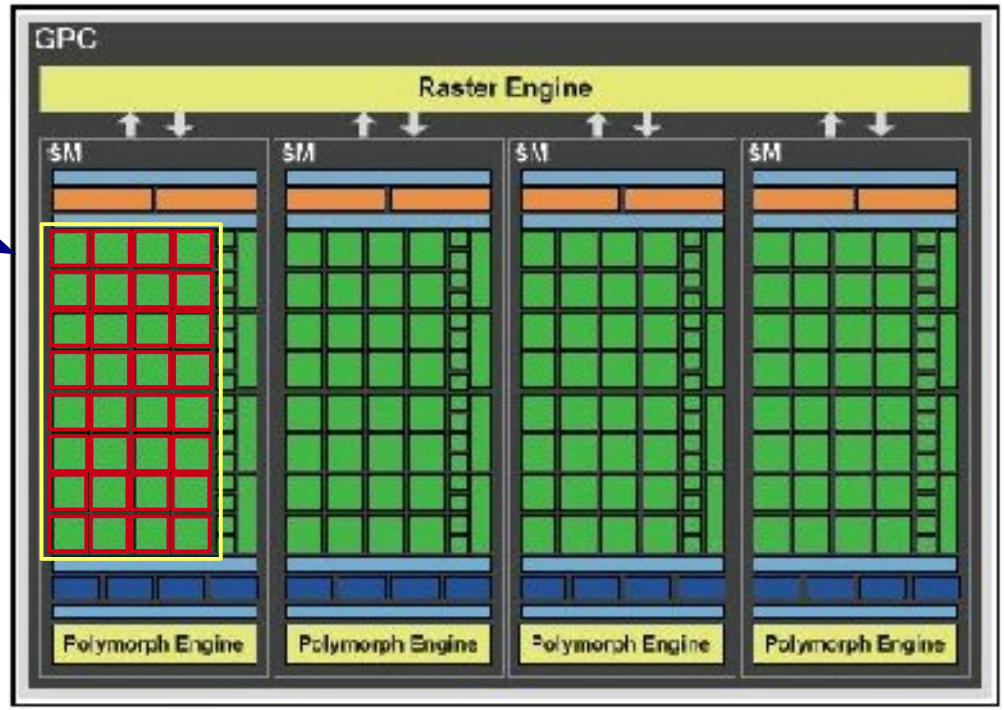
bloco threads



Entendendo Blocos e Threads

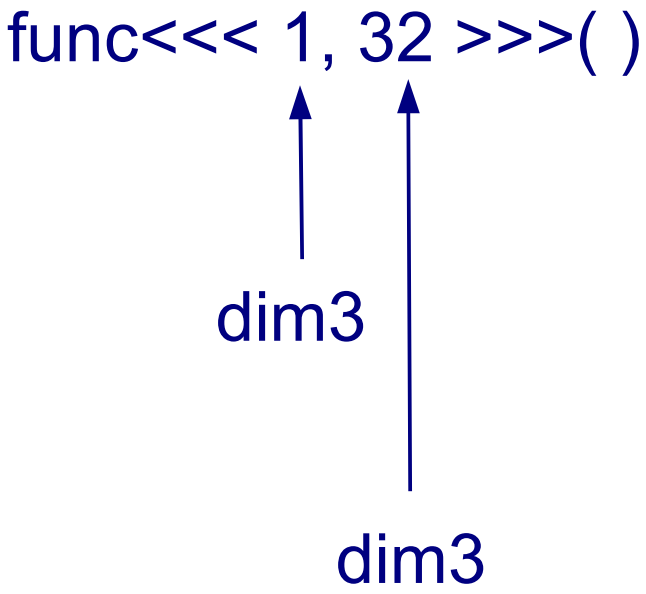
func<<< 1, 32 >>>( )

bloco threads



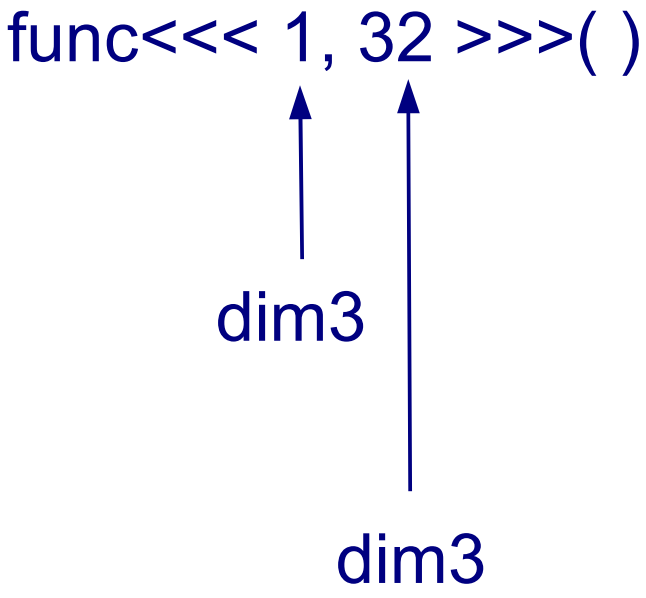
Como uma thread sabe quem é ela?

Entendendo Blocos e Threads



dim3 → (x, y, z)

Entendendo Blocos e Threads



Indices das Threads:
threadIdx.x, threadIdx.y, threadIdx.z

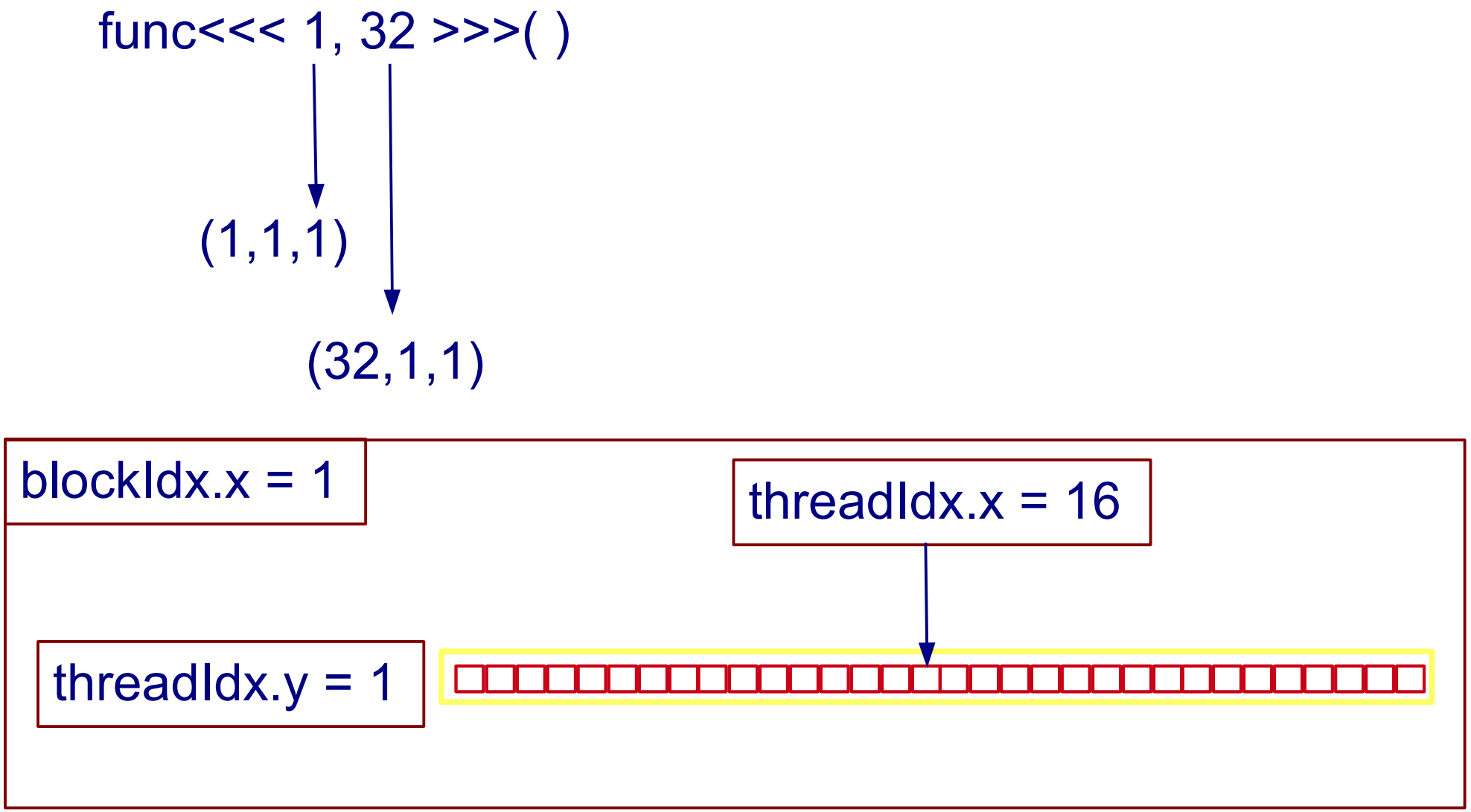
Indices dos Bloco:
blockIdx.x, blockIdx.y, blockIdx,z

Tamanho dos Blocos
blockDim.x, blockDim.y, blockDim.z

Tamanho da Grid:
gridDim.x, gridDim.y, gridDim.z

dim3 → (x, y, z)

Entendendo Blocos e Threads

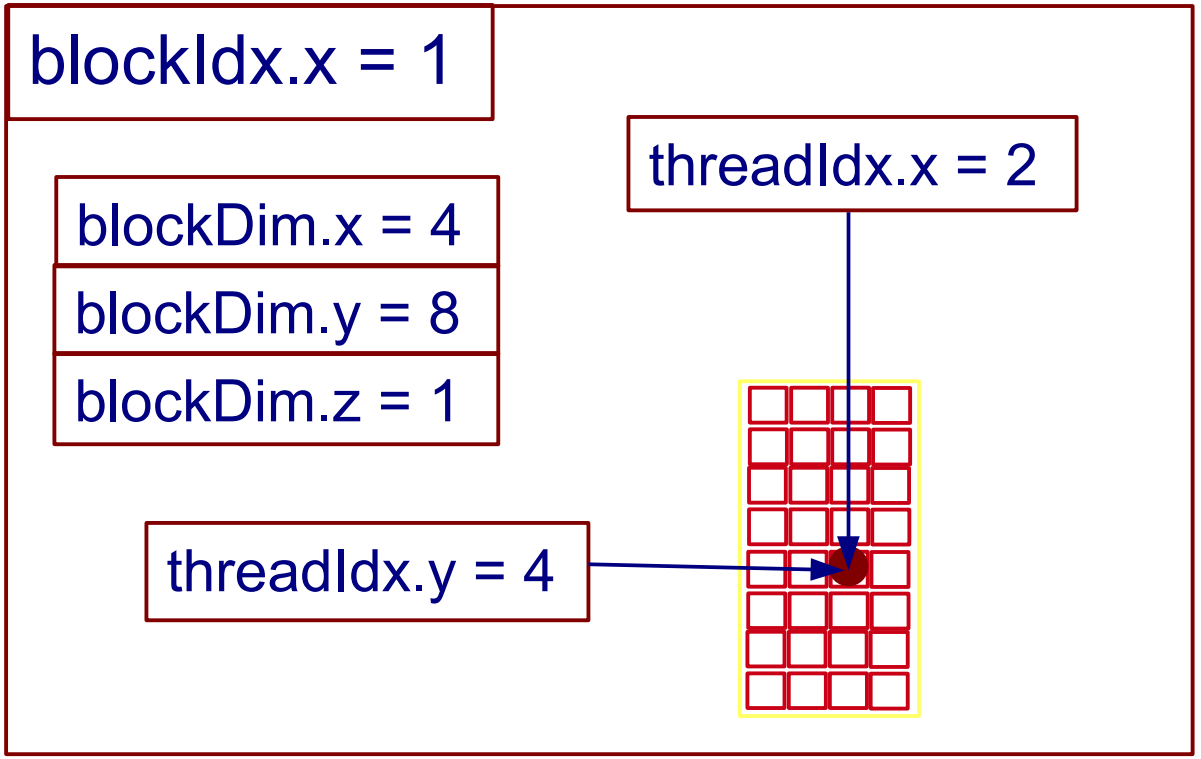


Entendendo Blocos e Threads

```
dim3 bloco( 4, 8 )
func<<< 1, bloco >>>( )
```

(1,1,1)

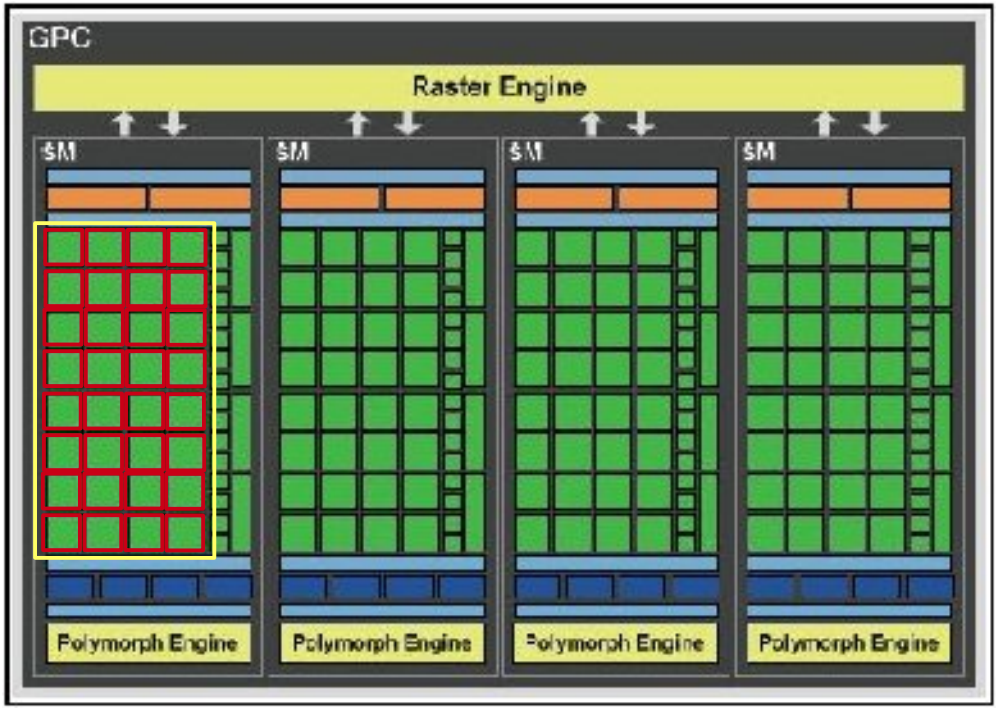
(4,8,1)



Entendendo Blocos e Threads

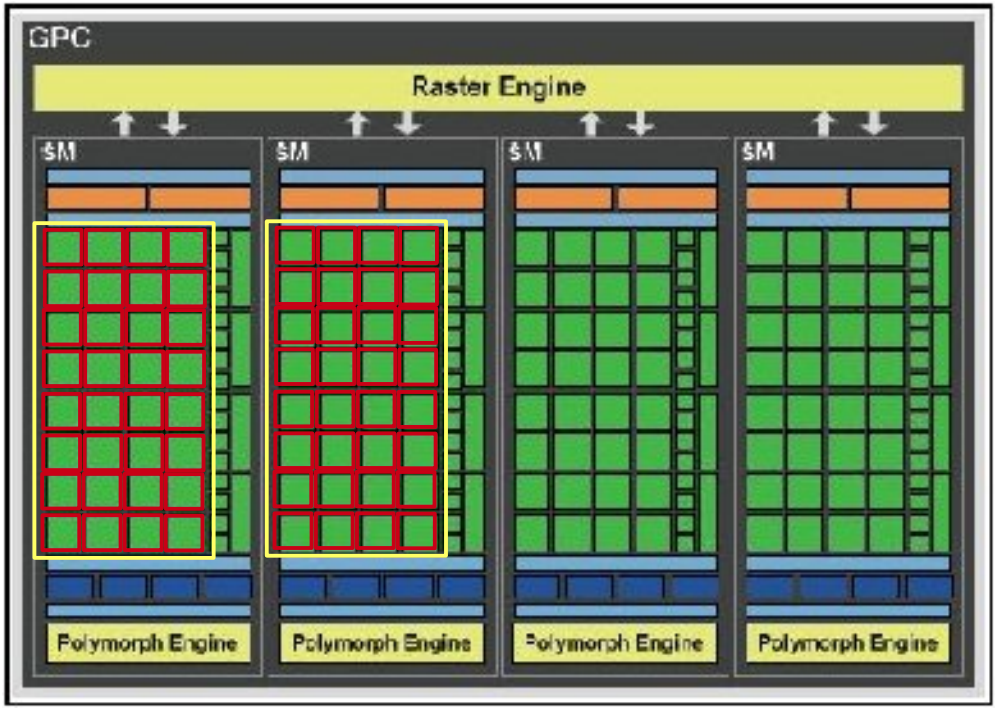
func<<< 1, 32 >>>( )

Executado em 1 warps no mesmo SM



func<<< 2, 32 >>>( )

Executado em 1 warp em diferentes SM



Exemplos Práticos