Insper

SuperComputação

Aula 14 – Programação em CUDA C(++)

2019 – Engenharia

Igor Montagner, Luciano Soares <igorsm1@insper.edu.br>

Aulas passadas

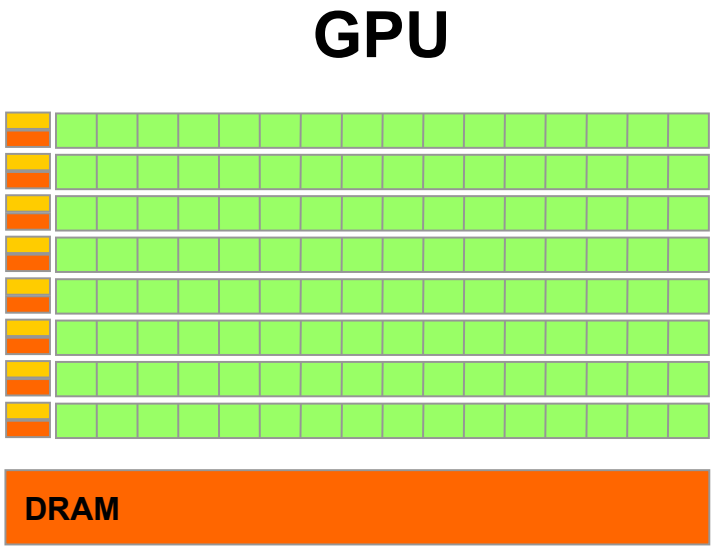
- Instalação do CUDA toolkit
- Compilação de programas para GPU
- Operações usando *thrust*
  - Alocação e transferência de dados
  - Transformações e reduções

Objetivos de aprendizagem

- Programar kernels em CUDA C para dados uni e bidimensionais
- Medir tempo levado por operações na GPU

GPU minimiza throughput

- ALU simples
  - Eficiente energeticamente
  - Alta taxa de transferência
- Cache pequeno
  - Acesso contínuo a RAM
- Controle simples
- Número massivo de threads



CPU vs GPU

- CPUs para partes sequenciais onde uma latência mínima é importante
  - CPUs podem ser 10X mais rápidas que GPUs para código sequencial

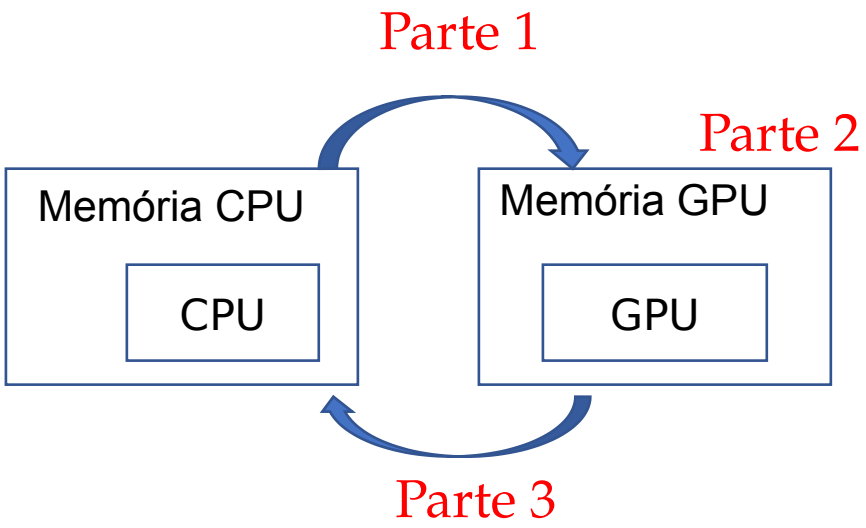
- GPUs para partes paralelas onde a taxa de transferência(throughput) bate a latência menor.
  - GPUs podem ser 10X mais rápidas que as CPUs para código paralelo

Fluxo de um programa

Parte 1: copia dados CPU → GPU

Parte 2: processa dados na GPU

Parte 3: copia resultados GPU → CPU

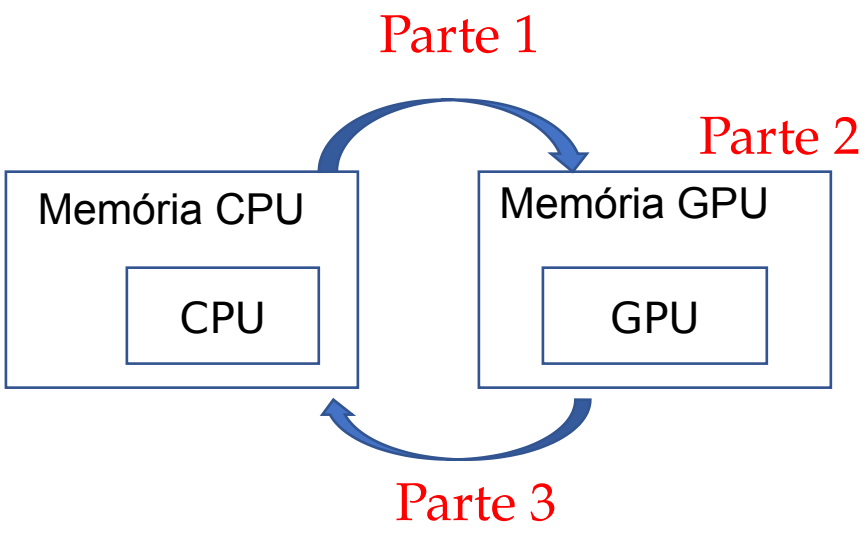


Fluxo de um programa

Parte 1: copia dados CPU → GPU (Thrust)

Parte 2: processa dados na GPU (CUDA C)

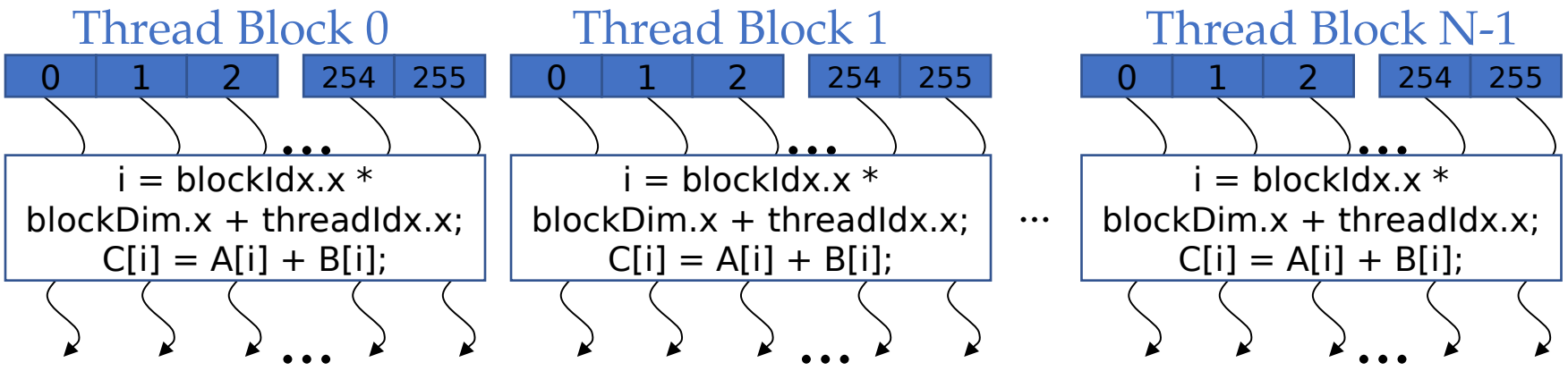
Parte 3: copia resultados GPU → CPU (Thrust)



CUDA C

- API específica para GPUs Nvidia que permite programação paralela massiva
- Baixo nível → expõe detalhes de design das GPUs; permite fine tuning para GPUs específicas
- Função que roda na GPU = kernel
- Funções auxiliares para alocar, transferir e gerenciar dados na memória da GPU
  - Thrust é construída usando essa API

CUDA C – blocos e threads



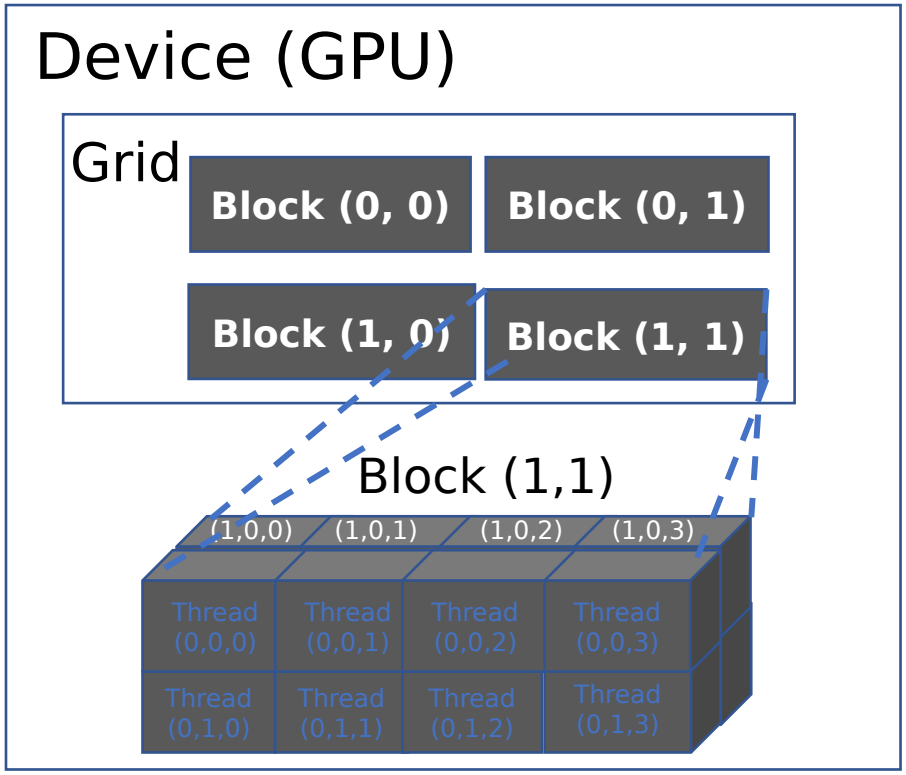
- As threads são divididas em múltiplos blocos (de threads)
  - Threads no mesmo bloco cooperam via **shared memory, atomic operations** e **barrier synchronization**
  - Threads em blocos diferentes não cooperam diretamente

CUDA C – blocos e threads

- Divide um problema grande em blocos de threads
- Cada bloco tem tamanho fixo
- Blocos podem ser executados em qualquer ordem e são independentes uns dos outros
- Sincronização entre blocos não existe
- Blocos podem formar um grid 2D ou 3D
- **Todas as threads rodam o mesmo código**

CUDA C – blocos e threads

- Cada thread usa índices para localizar o item de dado a ser processado:
  - threadIdx: 1D, 2D ou 3D
  - blockIdx: 1D, 2D ou 3D
  - blockDim: 1D, 2D ou 3D
- Endereçamento simplificado ao processar dados multidimensionais
  - Imagens
  - Equações diferenciais no espaço



Insper

CUDA C – exemplo 1

Cada bloco processa
blockDim.x
elementos do vetor

```
// Calcula soma de A + B e salva em C
// Cada thread realiza uma dessas somas

__global__
void vecAddKernel(float* A, float* B, float* C, int n)
{
    int i = threadIdx.x+blockDim.x*blockIdx.x;
    if(i<n) C[i] = A[i] + B[i];
}
```

Cada thread processa
um elemento do vetor

CUDA C – exemplo 1

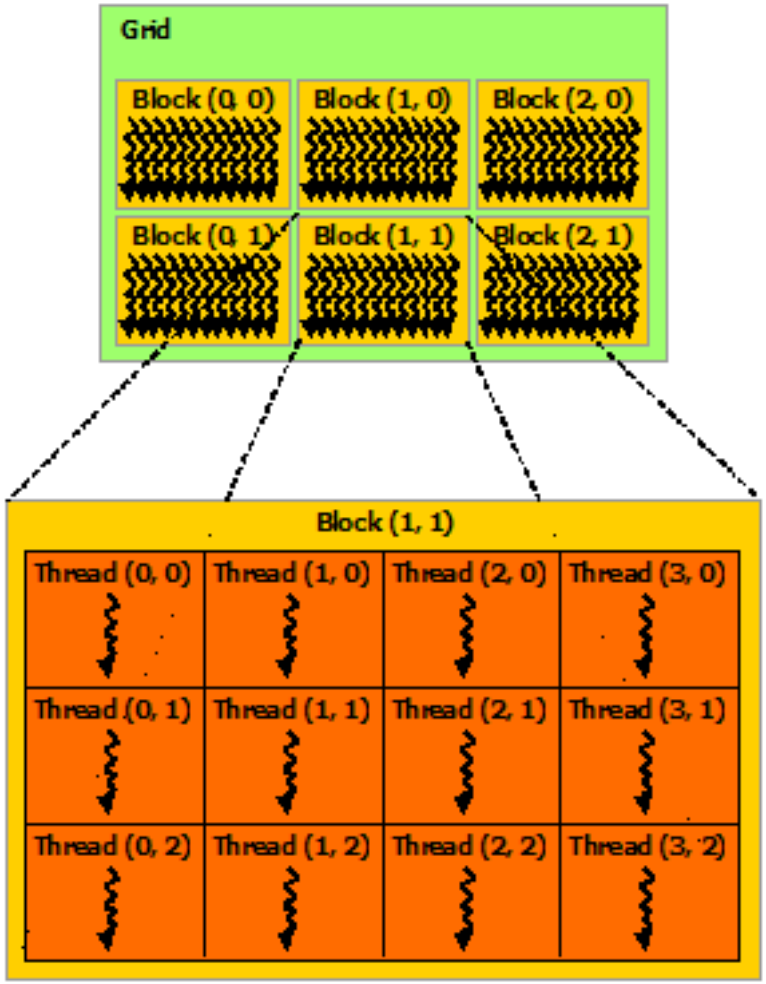
```
void vecAdd(float* h_A, float* h_B, float* h_C, int n)
{
 // d_A, d_B, d_C alocações e cópias omitidas
 // ceil(n/256.0) blocos de 256 threads cada
  vecAddKernel<<<ceil(n/256.0),256>>>(d_A, d_B, d_C, n);
}
```

Número de blocos necessários para processar o vetor

Threads por bloco

CUDA C – exemplo 2

- Matrizes podem ser divididas em blocos 2D
- Cada bloco possui
  - largura *blockDim.x*
  - altura *blockDim.y*
  - posição *blockIdx.x* e *blockIdx.y*
- Cada thread possui posição *threadIdx.x* e *threadIdx.y* dentro do bloco
- Localização do dado segue a mesma lógica do vetor 1D, mas é feita para as duas dimensões



Insper

CUDA C – exemplo 2

```
__global__ void add_one(int *input, int height, int width) {
    int i=blockIdx.y*blockDim.y+threadIdx.y;
    int j=blockIdx.x*blockDim.x+threadIdx.x;

    if (i < height && j < width) {
        input[i * width + j] += 1;
    }
}


// dentro do main
dim3 dimGrid(ceil(nrows/16.0), ceil(ncols/16.0), 1);
dim3 dimBlock(16, 16, 1);

add_one<<<dimGrid,dimBlock>>>(image_raw_pointer, nrows, ncols);
```

Posição do elemento na imagem a partir do grid

Número de quadrados para cobrir imagem

Bloco de tamanho 16 x 16

Atividade para aula

Programação em CUDA C em matrizes.

Insper

www.insper.edu.br