Notas dos slides

APRESENTAÇÃO

O presente conjunto de slides pertence à coleção produzida para a disciplina *Introdução ao Processamento Paralelo e Distribuído* ofertada aos cursos de bacharelado em Ciência da Computação e em Engenharia da Computação pelo Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas.

Os slides disponibilizados complementam as videoaulas produzidas e tratam de pontos específicos da disciplina. Embora tenham sido produzidos para ser assistidos de forma independente, a sequência informada reflete o encadeamento dos assuntos no desenvolvimento do conteúdo programático previsto para a disciplina.

computação
Universidade Federal de Pelotas

universidade federal de pelotas
CDTec
centro de desenvolvimento tecnológico

Notas da videoaula

DESCRIÇÃO

Nesta videoaula são apresentados conceitos relacionados às arquiteturas manycore, apontando suas características, modos de funcionamento e exemplos.

O foco principal é voltado para GPUs e para o modelo de programação CUDA, pelo qual é possível explorá-las através do processamento paralelo.

OBJETIVOS

Nesta videoaula o aluno conhecerá as arquiteturas manycore que podem ser utilizadas para o processamento paralelo, em especial às GPUs e o modelo de programação CUDA.

> **With Moore's law winding down, GPUs, invented by NVIDIA in 1999, came just in time.**
>
> NVidia

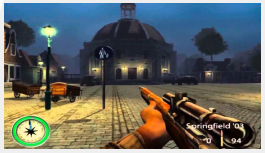
Arquiteturas Manycore

Também chamadas de **aceleradores** ou **arquiteturas heterogêneas**:

- Possuem altíssimo poder computacional (paralelismo);
- Oferecem custo de processamento menor em comparação com CPUs (financeiro e energético);
- Número de cores: centenas ou milhares;
- Exemplos: GPUs (placas de vídeo), Cell BEs (PlayStation 3), FPGAs, Intel Xeon Phi;
- **Tendencia atual em HPC:** 10 dos 15 computadores no topo da lista Top 500 usam aceleradores (2019).

Processamento em placas de vídeo?

- Evolução dos gráficos 3D para jogos passou a exigir um grande poder computacional;
- **GPU** (Graphics Processing Units): é o processador da placa de vídeo:
  - Responsável por processar os pixels exibidos na tela (muitas operações);
  - Possui muitos processadores de para executar tais operações;
- **GPGPU** (General Purpose Graphics Processing Units): uso de GPUs para o processamento de propósito geral, ou seja, para o processamento que não seja gráfico.



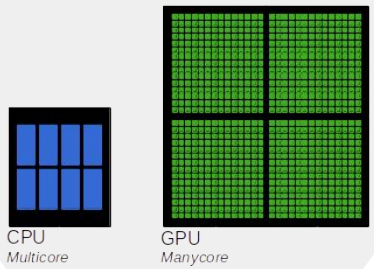
Arquitetura das GPUs

GPUs e CPUs possuem arquiteturas distintas:

- **CPU:** múltiplos *cores* com **grande** capacidade, tanto em velocidade (clock) quanto em complexidade de operações;
- **GPU**: milhares de *cores* de **pequena** capacidade, tanto em velocidade (clock) quanto em complexidade de operações;
  - Nos cores da GPU predomina a Unidade Lógica Aritmética (ALU);

Exemplo:

- Intel Core i7-9700F: **8 cores de 3,00 GHz;**
- Nvidia Titan RTX: **4.608 cores de 1,35 GHz;**
- https://www.youtube.com/watch?v=-P28LKWTzrI



GPUs

Questão 1

Como uma GPU pode processar um código mais rápido que uma CPU se a capacidade dos seus *cores* é menor?

Questão 2

Como faço para utilizar uma GPU para executar meu programa?

Questão 3

Eu tenho uma GPU no meu computador. Posso utilizá-la para executar meu programa?

Questão 4

Que tipos de programas podem ser executados de forma eficiênte em GPUs?

Programação para GPUs

CUDA: modelo de programação de propósito geral exclusivo para GPUs Nvidia;

- Extensão da linguagem C para controlar a execução de threads em GPUs;
- Disponibilizado gratuitamente pela Nvidia (atualmente na versão 11): https://developer.nvidia.com/cuda-toolkit

OpenCL: modelo de programação multiplataforma: suporta GPUs (Nvidia, AMD, ARM) e outros aceleradores, como FPGAs;

OpenACC: especificação baseada em diretivas de compilação (como OpenMP) definindo partes de código para execução em paralelo.

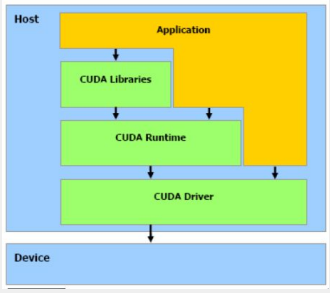
CUDA

Compute **U**nified **D**evice **A**rchitecture;

Oferece suporte a diversos ambientes além de C, como Fortran, OpenCL, Python e Java;

Componentes necessários para executar um programa utilizando CUDA:

- Ambiente CUDA (toolkit) composto por suas **bibliotecas**, **runtime**, **driver da GPU** e **compilador** (nvcc):
  - O `nvcc` necessita de um compilador C padrão (`gcc` ou outro);
- Dispositivo GPU (um ou mais) com suporte a CUDA.
- **Host:** computador - **Device:** GPU



Modelo de execução CUDA

Execução do programa controlada pela CPU:

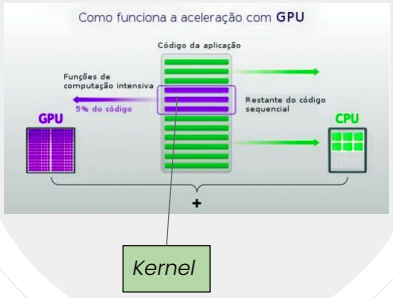
- **Aceleração:** trechos de código (*kernels*) são lançados pela CPU para execução em paralelo na GPU;

A execução do programa CUDA é composta por ciclos:

- CPU, GPU, CPU, GPU, ..., CPU, GPU, CPU;

GPU: somente realiza operações na sua memória interna:

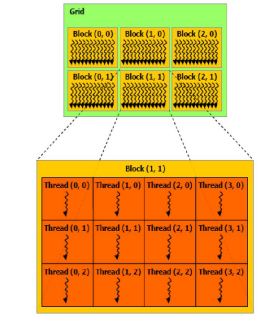
- A memória (Ent./Saída) acessada pelo *kernel* deve estar disponível na GPU;



Execução de aplicações

A execução do *kernel* é composta por:

- **Grade:** agrupa os elementos para executar o *kernel*, ou seja, os blocos;
- **Blocos:** conjunto de threads que executam um *kernel*. Os blocos são executados em paralelo;
- **Thread:** threads que compõe um bloco, sendo executadas simultaneamente em grupos de 32 (*warp*).



Hierarquia de memória para execução do *kernel*

Registradores: acessados individualmente por cada thread:

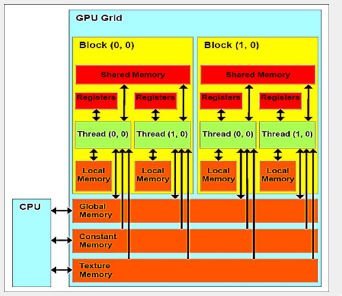
- Tempo de vida: execução da thread;
- Usadas para declarar variáveis locais no *kernel*;

Memória local: mesmo escopo e tempo de vida dos registradores, porém mais lenta:

- Não é física, sendo uma abstração da memória global;

Memória compartilhada: acessada pelas threads do mesmo bloco:

- Tempo de vida: execução das threads do bloco;
- Execuções com cooperação (acumuladores, contadores...) das threads do mesmo bloco;



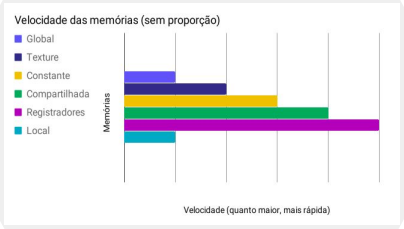
Hierarquia de memória para execução do *kernel*

Memórias global, constante e *texture*: acessadas por todas as threads:

- Tempo de vida: execução do programa;
- Utilizadas para as transferências de memória (CPU->GPU);
- Memórias **constante** e ***texture***: são mais rápidas que a global, porém suportam apenas leitura (threads não podem escrever nelas):
  - A memória **constante** é menor porém mais rápida que a ***texture***;
  - São utilizadas por aplicações muito específicas;

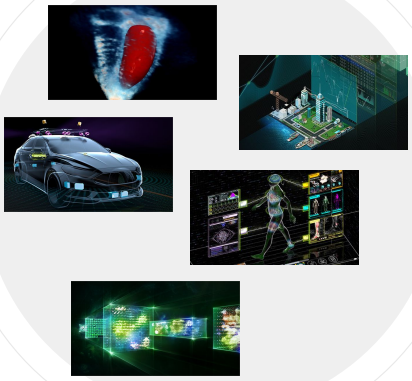
Importante conhecer a hierarquia de memória para implementar soluções eficientes.

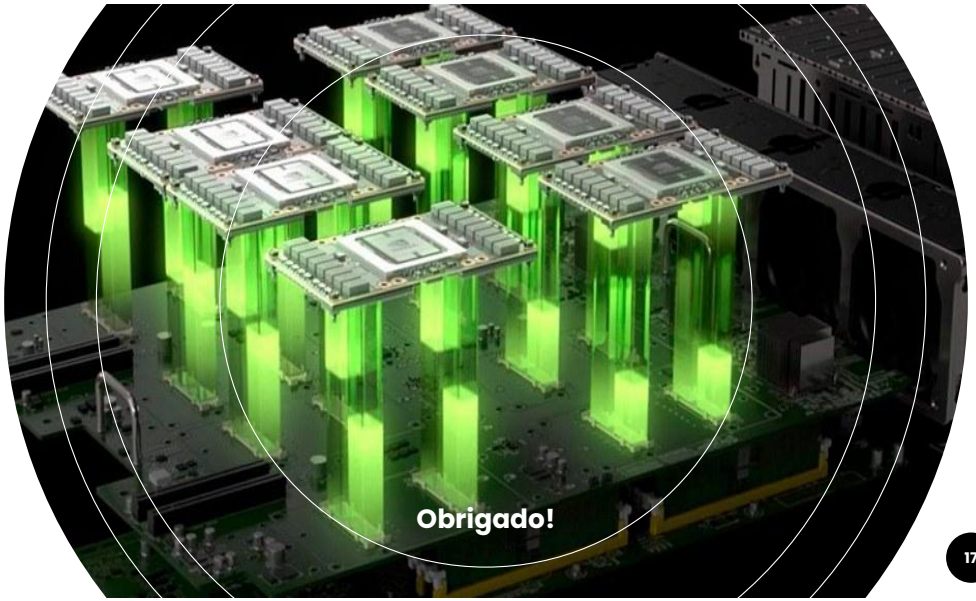
https://www.microway.com/hpc-tech-tips/gpu-memory-types-performance-comparison/



Onde GPUs e CUDA são aplicados?

- Aplicações científicas;
- Aplicações médicas;
- Aplicações de mercado financeiro;
- Aplicações de transportes (machine learning):
  - https://www.youtube.com/watch?v=-96BEoXJMs0
- Jogos;
- Supercomputação;
- Mais em:
  - https://www.nvidia.com/en-us/industries/




Obrigado!