Insper

SuperComputação

Aula 20 – GPU e números aleatórios

2021 – Engenharia

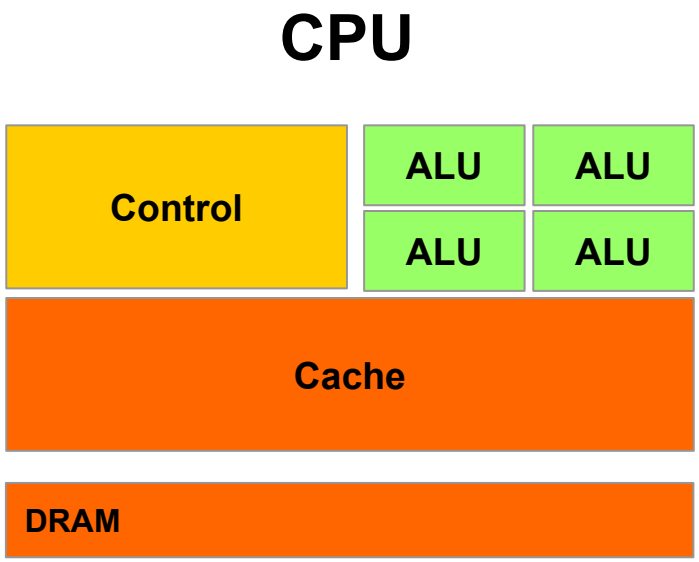
Igor Montagner <igorsm1@insper.edu.br>
Antônio Selvatici <antoniohps1@insper.edu.br>

Objetivos de aprendizagem

- Gerar números aleatórios de forma massivamente paralela em GPU

CPU minimiza latência

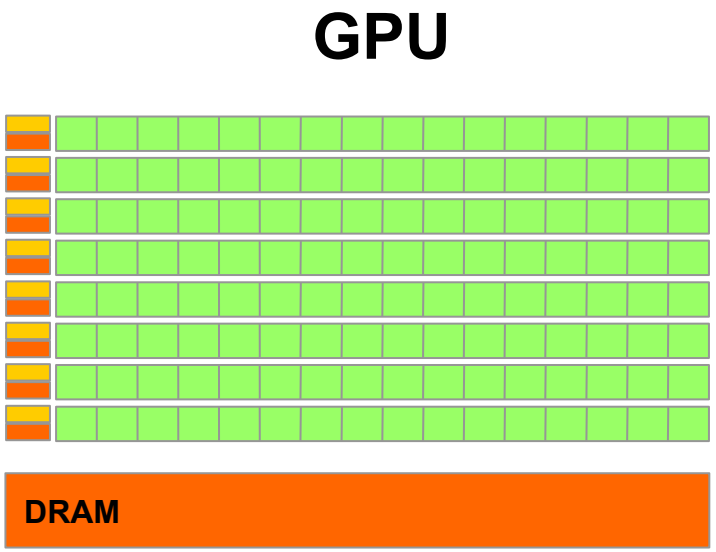
- ALU pontente minimiza latência das operações
- Cache grande:
  - Acelera operações lentas de acesso a RAM
- Controle sofisticado:
  - Branch prediction
  - Data forwarding



Insper

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

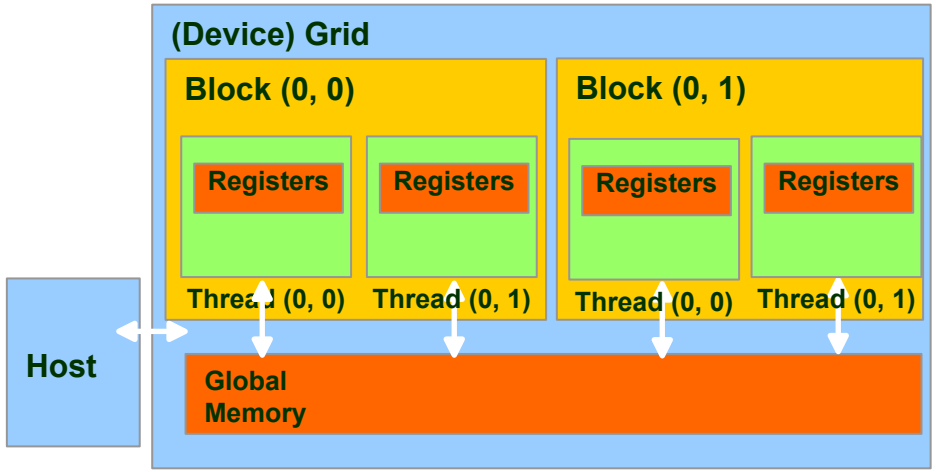
Memória em GPU

Código da GPU (device) pode:

- Cada thread ler e escrever nos registradores
- Ler e escrever na memória global

Código da CPU (host) pode:

- Transferir dados de e para memória global



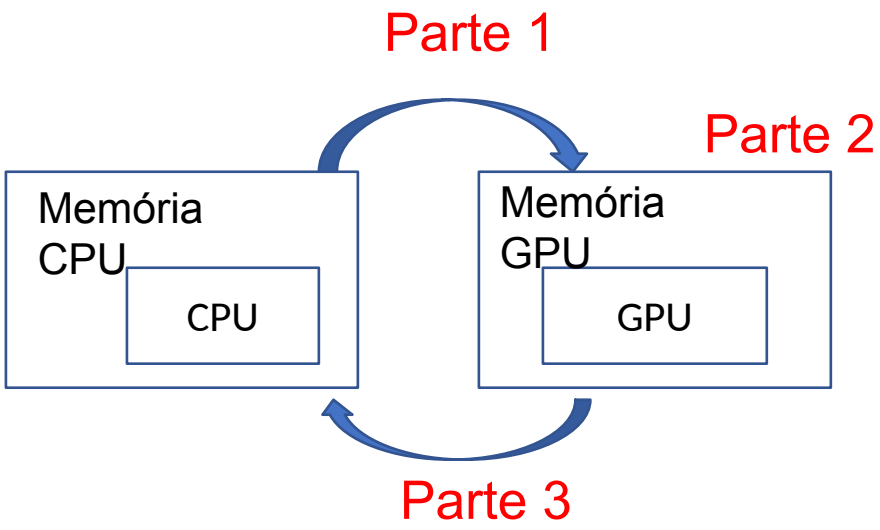
Insper

Fluxo de um programa

Parte 1: copia dados CPU → GPU

Parte 2: processa dados na GPU

Parte 3: copia resultados GPU → CPU



Gerando números (pseudo-)aleatórios

Gerador de números pseudo-aleatórios (**pRNG**): algoritmo determinístico que gera **sequências** de números que parecem aleatórias

1. **Determinístico:** produz sempre a mesma sequência.

2. **Sequências que parecem aleatórias:** não conseguiríamos distinguir uma sequência gerada por um pRNG e uma sequência aleatória de verdade.

Números (pseudo-)aleatórios

Sorteio de números aleatórios

1. **Gerador:** produz bits aleatórios a partir de um parâmetro **seed**. Cada **seed** gera uma sequência diferente de bits.

2. **Distribuição de probabilidade:** gera sequência de números a partir de um conjunto de parâmetros

Números (pseudo-)aleatórios usados na computação paralela

Duas abordagens:

1. **Gerar com código sequencial e distribuir entre as threads:** reduz a fração do tempo de execução que pode ser paralelizado. OK para pequenas sequências.

2. **Gerar nas próprias threads:** cada gerador tem a sequência de tamanho limitado a x/n, onde x é o tamanho da sequência original e n é o número de threads

Geração de números aleatórios em GPU

Como obter sequências de qualidade?

Atividade prática

Geração de números pseudo-aleatórios em GPU (30 minutos)

1. Usando métodos da API thrust
2. Implementar diferentes metodologias para gerar sequências de qualidade, diferenciando do caso da CPU

Atividade prática

Implementação do cálculo de pi com Monte Carlo (30 minutos)

1. Implementação sequencial
2. Implementação paralela em GPU

www.insper.edu.br