Programação para GPUs

Ricardo de la Rocha Ladeira

Introdução

Nas aulas anteriores, foram estudados aspectos sobre Programação de Alto Desempenho, envolvendo conceitos de processos, *threads*, comunicação entre processos e sincronização. Foram vistos conceitos sobre *threading* implícita e uso de bibliotecas como `pthread` e OpenMP em exemplos como a criação de um chat *multithread*, um algoritmo para multiplicação de matrizes e do conjunto de Mandelbrot.

À medida que se avança pela Programação de Alto Desempenho, percebe-se que a busca por eficiência computacional pode abranger diversas abordagens. Quando se trata de enfrentar desafios que demandam não apenas processamento intenso, mas também alto nível de paralelismo, a programação para Unidades de Processamento Gráfico (*Graphical Processing Units* – GPUs) emerge como uma das principais abordagens de exploração.

O paralelismo massivo, o poder de processamento e as aplicações em uma variedade de campos tornam essa abordagem uma peça-chave no estudo da Programação de Alto Desempenho.

GPU

A Unidade de Processamento Gráfico (*Graphical Processing Unit* – GPU) é o componente do computador que renderiza elementos gráficos. A GPU é projetada para lidar eficientemente com cálculos intensivos em paralelo, tornando-a essencial para aplicações que demandam alto desempenho gráfico, como jogos, design gráfico e processamento de vídeo. Sua arquitetura permite processar grandes volumes de dados visuais, contribuindo para uma experiência visual mais fluida e realista em uma variedade de dispositivos eletrônicos.

Além de sua função primária no processamento gráfico, a GPU também passou a ser usada para fazer computação de propósito geral. Sua arquitetura permite lidar com conjuntos complexos de dados e realizar cálculos intensivos de maneira eficiente. Com isso, a GPU

tornou-se uma peça central não apenas na busca por experiências visuais aprimoradas, mas também na aceleração de inovações tecnológicas em campos diversos.

Os principais fabricantes de GPUs são a NVIDIA, conhecida por suas GPUs GeForce (Figura 1), Quadro e Tesla; a AMD (*Advanced Micro Devices*), fabricante de GPUs Radeon; e a Intel, muito conhecida por suas CPUs, mas também pelas GPUs Iris Xe.

Figura 1. NVIDIA GeForce RTX 2080 [1].

Anatomia da GPU

- **Arquitetura Paralela**: Uma GPU possui centenas a milhares de núcleos de processamento, também conhecidos como *CUDA Cores* (no caso da NVIDIA) ou *Stream Processors* (em GPUs em geral). Esses núcleos são responsáveis por realizar cálculos simultâneos, tornando a GPU eficiente em tarefas paralelas.
- **Barramentos de Memória**: A GPU possui sua própria memória de vídeo (VRAM – *Video Random Access Memory*), conectada por barramentos de alta largura de banda. Isso permite que ela acesse rapidamente os dados necessários para processamento gráfico e outras operações paralelas.
- **Hierarquia de Memória**: Cada núcleo de processamento possui registradores locais para armazenar dados temporários, permitindo operações rápidas.
- **Memória Cache**: A GPU possui hierarquias de cache para armazenar dados temporários e reduzir a latência de acesso à memória principal.
- **Memória Global (VRAM)**: É a memória principal da GPU, onde são armazenados dados, texturas, *shaders*[1] e outros recursos necessários para as operações gráficas.

[1] *Shaders* são programas utilizados em GPUs para realizar cálculos relacionados ao processamento gráfico. Eles desempenham um papel fundamental no pipeline gráfico.

- **APIs e Drivers**: A GPU é controlada por APIs (*Application Programming Interfaces* – Interfaces de Programação de Aplicações) como DirectX, OpenGL ou CUDA. *Drivers* específicos garantem a comunicação eficiente entre o sistema operacional, a CPU e a GPU.
- **Pipeline Gráfico**:
    - Entrada/Saída: Os dados de entrada, como geometria 3D e texturas, são processados pela GPU.
    - *Shader* de Vértice: Transforma as coordenadas dos vértices em espaço 3D.
    - Tesselação: Controla a densidade da malha 3D para otimizar a qualidade visual.
    - *Shader* de Geometria: Processa primitivas gráficas, como pontos, linhas e triângulos.
    - Rasterização: Converte primitivas 3D em pixels para exibição na tela [2].
    - *Shader* de Pixel (*Shader* de Fragmento): Calcula a cor e outras propriedades de cada pixel.
    - Saída: A imagem final é enviada para a tela.

Aplicações da Programação com GPUs

Existem várias áreas de aplicação para a programação com GPUs. Entre elas:

- **Gráficos e Jogos**:
    - Desenvolvimento de jogos [3], simulações e gráficos computacionais de alta qualidade.
    - Renderização de gráficos 3D em tempo real [4].
- **Aprendizado de Máquina e Inteligência Artificial**:
    - Treinamento de modelos de *Machine Learning* e *Deep Learning*.
    - Desenvolvimento de algoritmos de processamento de dados em grandes conjuntos de dados.
- **Ciência de Dados**:
    - Análise e visualização de grandes conjuntos de dados.
    - Processamento paralelo para acelerar operações em conjuntos de dados extensos.
- **Renderização de Vídeo**:
    - Processamento rápido de vídeo e renderização de efeitos especiais.
    - Processamento de vídeo em tempo real [5].
- **Medicina e Bioinformática**:

    - Análise de imagens médicas, como endoscopias [6] tomografias [7] e ressonâncias magnéticas [8].
    - Alinhamento de sequências genéticas e análise de *big data* em bioinformática [9].
- **Finanças Quantitativas**:
    - Modelagem e simulação de instrumentos financeiros [10].
    - Análise de riscos e precificação de derivativos [11].
- **Realidade Virtual e Aumentada**:
    - Desenvolvimento de experiências interativas e ambientes virtuais.
    - Renderização de ambientes 3D para aplicações de realidade aumentada.
- **Segurança Computacional**:
    - Aceleração de ataques de força bruta em algoritmos de criptografia.
    - Mineração de criptomoedas através do cálculo de *hashes*.
    - Geração de chaves criptográficas.

A lista não é exaustiva; além disso, há ainda outros exemplos em áreas como Física, Química, Biologia e Engenharia. Isso demonstra o quanto a programação com GPUs continua a se expandir para novas áreas à medida que a tecnologia avança.

Programação

A programação para GPUs tem sido realizada com tecnologias como C, C++, CUDA (*Compute Unified Device Architecture*) e OpenCL (*Open Computing Language*). C e C++ são linguagens de programação tradicionais amplamente utilizadas para desenvolvimento de software em geral. Para programação em GPUs, elas são estendidas com APIs específicas, como o CUDA para GPUs NVIDIA e o OpenCL para dispositivos gráficos de diferentes fabricantes.

CUDA [12], desenvolvido pela NVIDIA, é, na verdade, uma plataforma de computação paralela que permite aos programadores utilizarem GPUs NVIDIA para acelerar tarefas computacionais intensivas. É uma extensão das linguagens C e C++ com adições específicas para a programação em GPUs.

OpenCL [13] é uma estrutura de programação paralela de código aberto que suporta o desenvolvimento de software para uma variedade de dispositivos de processamento, incluindo GPUs. Ele oferece uma abordagem mais genérica para programação paralela em

comparação com o CUDA, sendo suportado por vários fabricantes, como AMD, Intel e NVIDIA.

Resumo

Este texto abordou a programação para GPUs, explorando sua relevância na Programação de Alto Desempenho. Após introduzir conceitos básicos de *threads*, processos e sincronização, destaca-se a GPU como um componente essencial não apenas para tarefas de processamento gráfico, mas também de computação paralela em grande escala, de forma a aparecer como uma alternativa a CPUs. Com arquitetura projetada para paralelismo massivo, a GPU pode ser utilizada tanto para renderização gráfica quanto para cálculos complexos, sendo empregada em áreas como jogos, aprendizado de máquina, processamento de vídeo e bioinformática.

A anatomia das GPUs inclui componentes como núcleos paralelos, memória de vídeo de alta largura de banda (VRAM) e hierarquias de cache, possibilitando o processamento eficiente de grandes volumes de dados. Além disso, tecnologias de programação como CUDA, exclusiva para GPUs NVIDIA, e OpenCL, de código aberto e compatível com múltiplas plataformas, facilitam o desenvolvimento de aplicações intensivas em cálculo que podem tirar proveito do poder de processamento das GPUs. Essas tecnologias transformam as GPUs em ferramentas poderosas para aplicações de diversas áreas.

Exercícios

1) Pesquise e crie um tutorial de uso de GPUs com o Google Colab.
2) Existem alternativas ao Google Colab? Forneça uma lista com alternativas (pagas e/ou gratuitas).

Gabarito

1) Livre.
2) Livre.

Referências

[1] BURNES, Andrew. GeForce RTX Founders Edition Graphics Cards: Cool and Quiet, and Factory Overclocked. August 20, 2018. Disponível em: https://www.nvidia.com/en-us/geforce/news/geforce-rtx-founders-graphics-card-breakdown/. Acesso em: 20 nov. 2023.

[2] FATAHALIAN, Kayvon. How a GPU Works. Disponível em: https://www.cs.cmu.edu/afs/cs/academic/class/15462-f11/www/lec_slides/lec19.pdf. Acesso em: 20 nov. 2023.

[3] NOVAKOSKY, R. Computação através da GPU: ênfase em jogos digitais. 2014. Disponível em: http://ric.cps.sp.gov.br/bitstream/123456789/1152/1/20141S_NOVAKOSKYRoberto_TCCPD1258.pdf. Acesso em: 21 nov. 2023.

[4] RICHMOND, P.; ROMANO, D. Agent based gpu, a real-time 3d simulation and interactive visualisation framework for massive agent based modelling on the gpu. In: Proceedings International Workshop on Supervisualisation. 2008. Disponível em: https://www.researchgate.net/profile/Daniela-Romano-2/publication/228343510_Agent_based_gpu_a_real-time_3d_simulation_and_interactive_visualisation_framework_for_massive_agent_based_modelling_on_the_gpu/links/0c96051b8a1604cd6b000000/Agent-based-gpu-a-real-time-3d-simulation-and-interactive-visualisation-framework-for-massive-agent-based-modelling-on-the-gpu.pdf. Acesso em: 21 nov. 2023.

[5] RATHORE, M. M. et al. Real-time video processing for traffic control in smart city using Hadoop ecosystem with GPUs. Soft Computing, v. 22, p. 1533-1544, 2018.

[6] FONSECA, F. X., FERNANDES, J. M., OLIVEIRA, I. C., & CAMPOS, G. Análise paralela de imagem endoscópica com recurso a GPU. In 3o INForum Informatics Symposium. 2011.

[7] TAVARES, R. S. Reconstrução de imagens por tomografia por impedância elétrica utilizando recozimento simulado massivamente paralelizado. 2016. Tese de Doutorado. Universidade de São Paulo.

[8] DANTAS, Thales Henrique. Reconstrução de imagens de ressonância magnética acelerada por placas de processamento gráfico. Dissertação de Mestrado. 2014. Disponível em: https://repositorio.unb.br/bitstream/10482/20195/1/2015_ThalesHenriqueDantas.pdf. Acesso em: 20 nov. 2023.

[9] NOBILE, M. S. et al. Graphics processing units in bioinformatics, computational biology and systems biology. Briefings in bioinformatics, v. 18, n. 5, p. 870-885, 2017.

[10] SOUZA, T. T. P. Simulações financeiras em GPU. Tese de Doutorado. Universidade de São Paulo. 2013. Disponível em:

https://www.teses.usp.br/teses/disponiveis/45/45134/tde-23052013-234703/publico/mestrado.pdf. Acesso em: 21 nov. 2023.
[11] ISHIKAWA, F. T. M. Faculdade de Engenharia Mecânica. Tese de Doutorado. Universidade Estadual de Campinas.2014. Disponível em: http://www.fem.unicamp.br/~lotavio/tgs/2014_Acelera%C3%A7%C3%A3oC%C3%A1lculosFinanceirosUtilizandoGPUs_TG_FernandoIshikawa.pdf. Acesso em: 21 nov. 2023.
[12] NVIDIA. NVIDIA CUDA. NVIDIA Docs Hub. Disponível em: https://docs.nvidia.com/cuda/doc/index.html. Acesso em: 21 nov. 2023.
[13] THE KHRONOS GROUP. OpenCL. Disponível em: https://www.khronos.org/api/opencl. Acesso em: 21 nov. 2023