IME - Instituto de Matemática e Estatística

Comparação de Desempenho entre OpenCL e CUDA

Thiago de Gouveia Nunes
Orientador: Prof. Marcel P. Jackowski

Introdução

GPGPU (General-purpose computing on graphics processing units) é utilizar uma GPU (Graphic Processing Unit) ao invéz de uma CPU (Central Processing Unit) para realizar as computações de uma aplicação.

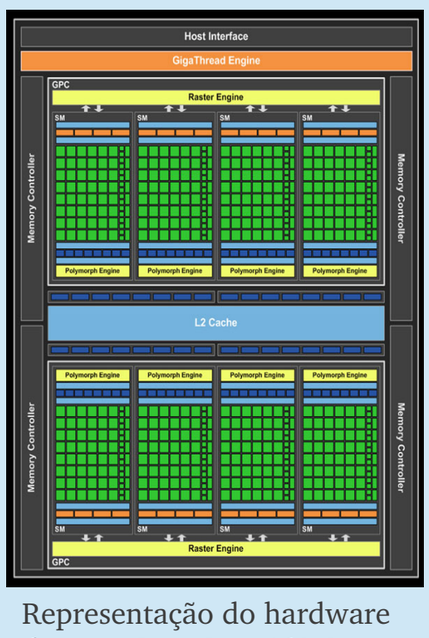
Existem 2 linguagens fortes no mercado para programação GPGPU.
Uma delas é o CUDA (Compute Unified Device Architecture), desenvolvida pela NVIDIA para placas NVIDIA, e a outra é o OpenCL (Open Computing Language), uma alternativa open source desenvolvida em conjunto por várias empresas, e aprovada pelo grupo Khronos.

GPGPU nasceu do hardware diferenciado da GPU, que permite que problemas que são altamente paralelizaveis e com o minimo de dependencia entre suas threads sejam executados em uma fração de tempo do que eles seriam numa CPU.

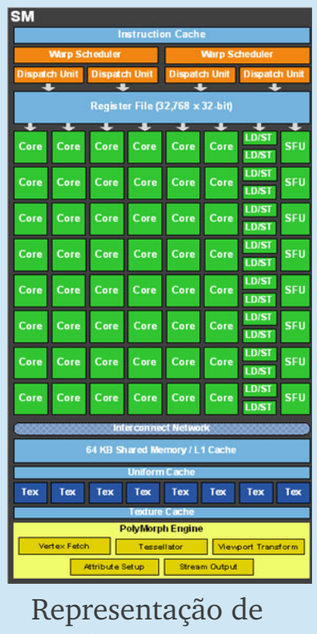
O hardware da GPU é composto de um bloco de memória principal, um escalonador principal, unidades de processamento gráfico e vários multiprocessadores especiais, chamados de *Streaming Multiprocessors*.

As GPUs seguem o paradigma de execução *SIMD* (Single Instruction, Multiple Data), onde cada thread na GPU executa o mesmo código, mas com dados diferentes. Chamamos de *Kernel* o código que será executado na GPU. Cada thread é uma instância do *Kernel*.

As threads são escalonadas para os processadores dentro dos SM, e lá elas são executadas em conjunto, ou seja, dado um instante de tempo elas estão executando a mesma instrução. Esse é o principal motivo que define ogrupo de problemas ótimo para uma GPU.



Representação do hardware de uma GPU GTX 460 SE.



Representação de um Streaming Multiprocessor

Medida de desempenho

Para qualificar as duas linguagens, foram montados 2 tipos de kernels.

- O primeiro é Memory Bound, ou seja, seu tempo de execução é ditado pela velocidade de acesso aos dados. Logo ele foi usado para verificar a velocidade de acesso a memória das linguagens. Esse kernel copia uma matriz 4096x4096 de floats para outra.
- O segundo kernel é Compute Bound, ou seja, seu tempo de execuçaõ é determinado unicamente pela velocidade de processamento. Ele é usado para avaliar o método de execução das duas linguagens. Esse kernel multiplica duas matrizes 1024x1024 de floats e depois guarda o valor numa terceira.

A GPU usada para os testes é uma GTX 460 SE da NVIDIA.

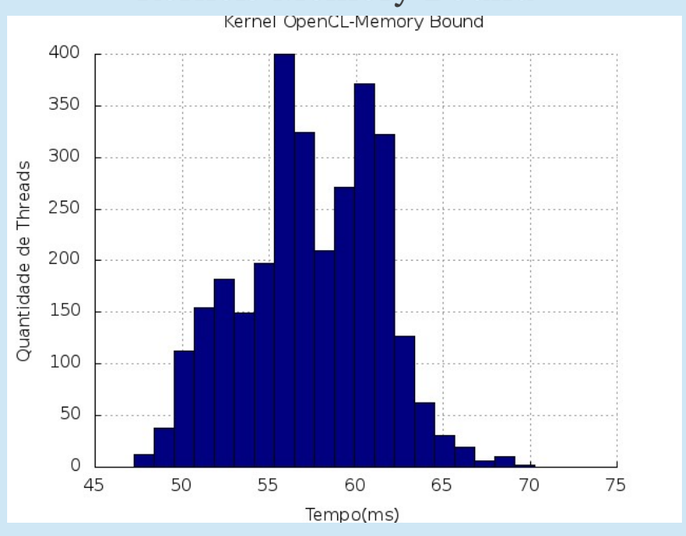
Resultados dos testes

Para colher resultados dos testes cada um dos kernels foi executado 3.000 vezes.

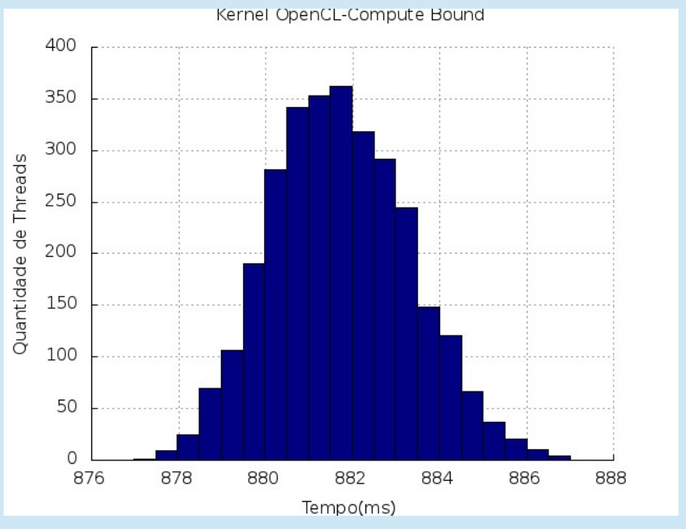
O CUDA foi, em média, 15 vezes mais rápido que o OpenCL tanto para o kernel Compute Bound como para o kernel Memory Bound.

Os histogramas abaixo mostram o tempo de execução dos kernels:

Kernels Memory Bound



Kernels Compute Bound



Por que essa diferença?

Uma das causas da diferença de desempenho entre as linguagens é o paradigma usado por elas ao abstrair a execução das threads na GPU.

O OpenCL foi desenvolvido para rodar em sistema hibrídos, em que tanto GPUs e CPUs são usados, e por essa abrangência ele paga com desempenho. Ele não reconhece várias funcionalidades que as placas NVIDIA oferecem, como por exemplo o uso de ponto flutuante de precisão dupla, enquanto o CUDA tem acesso a essas funcionalidades.

A outra causa está na compilação das linguagens para a máquina virtual das GPUs NVIDIA, a PTX (Parallel Thread Execution). Essa máquina virtual é usada para garantir a compatibilidade de um mesmo programa para GPUs diferentes.

É possível compilar os kernels para um arquivo .ptx, que depois será executado nessa máquina virtual.

Ao comparar esses dois arquivos, notou-se que os kernels do CUDA fazem menos acesso a memória global da GPU do que os do OpenCL. Isso faz uma grande diferença, já que essa memória deve ser acessada por todas as threads executando dentro da GPU, gerando um acesso sequêncial para esse acesso, que acaba diminuindo muito a velocidade de execução das threads.