Programação Paralela em GPU com o uso da Integração CUDA e MATLAB

Gesiel Rios Lopes[1], **Paulo Sérgio L. de Souza**[1]

[1]Instituto de Ciências Matemática e de Computação (ICMC)
Universidade de São Paulo (USP)
Avenida Trabalhador São-carlense, 400 – Centro
CEP: 13566-590 – São Carlos - SP - Brasil

gesielrios@usp.br, pssouza@icmc.usp.br

Resumo. *Viabilizar soluções computacionais que reduzam o tempo de processamento e forneçam respostas cada vez mais precisas é um dos grandes desafios da Ciência da Computação atual. Nesse sentido, a computação paralela surge como uma alternativa. Este trabalho avalia o uso do plugin PCT disponibilizado pela MathWorks® que realiza a integração CUDA-MATLAB através da resolução de dois problemas, o custo para resolução de grandes sistemas lineares e a resolução de equação de onda de segunda ordem através de métodos espectrais. Os resultados obtidos indicam um ganho na eficiência computacional cerca de 5x a 10x em comparação à implementação sequencial em uma CPU.*

1. Introdução

O uso da computação paralela vem sendo cada vez mais necessário para o processamento de grandes volumes de dados, encontrados em problemas de diversas áreas da ciência. Assim, se faz extremamente necessária a busca pelo aumento do desempenho computacional à medida que o volume de dados aumenta. Não menos importante, a utilização de um bom modelo de programação também se torna necessário para viabilizar o acesso e a computação dos dados. Modelos de programação como *Compute Unified Device Architecture* (CUDA) [Nvidia 2011] permitem que aplicações possam ser desenvolvidas mais facilmente na GPU.

Empresas como a Intel® e AMD® melhoraram significativamente o desempenho da CPU, aumentando o número de núcleos. No entanto, por razões técnicas, um chip CPU contém apenas um número reduzido de núcleos. Nos últimos anos, as GPUs tem recebido cada vez mais a atenção, já que o desempenho de processamento de ponto flutuante por segundo (*Flops*, do inglês *FLoating-point Operations Per Second*) tem se mostrado mais eficiente do que uma CPU, além disso, o seu consumo de energia e custos são bem menores do que aqueles dos sistemas à base de CPU [Belleman et al. 2008].

No entanto, tirar proveito dos benefícios da programação de uma GPU não é uma tarefa simples já que é necessário utilizar uma interface de programação padrão (API, do inglês *Application Programming Interface*) de gráficos, como *OpenGL* e *DirectX*, além do aprendizado de linguagens de programação específicas. A MathWorks® com o objetivo de ajudar os desenvolvedores e com o intuito de tornar a programação mais simples, desenvolveu um *plugin*, o *Parallel Computing Toolbox* (PCT), capaz de fazer a integração entre CUDA e o MATLAB. O uso do MATLAB para a computação em GPU

pode permitir que aplicações sejam aceleradas mais facilmente e de forma mais eficiente [Little and Moler 2013].

Este trabalho visa avaliar os benefícios obtidos por meio do uso do *plugin* PCT disponibilizado pela MathWorks®. O restante deste trabalho está organizado da seguinte maneira. Na Seção 2 temos os conceitos relacionados à programação de GPUs com o MATLAB e na Seção 3 temos a descrição dos experimentos realizados para avaliar o desempenho da integração CUDA-MATLAB, bem como os resultados comparativos encontrados nos experimentos realizados entre CPU e GPU. Por fim, a Seção 4 destaca as conclusões do trabalho.

2. Integração CUDA e MATLAB

O PCT permite a resolução de problemas computacionais através do uso intensivo dos processadores *multicore*, GPUs e *clusters*, disponibilizando uma forma mais eficiente para acelerar códigos desenvolvidos no MATLAB [Little and Moler 2013].

[Little and Moler 2013] destaca as seguintes características como sendo as principais do PCT:

- Paralelização de *loops* `for`;
- Suporte a CUDA em GPUs NVIDIA®;
- Uso do poder de processamento total dos *desktop multicore* através de workers locais;
- Suporte à computação em *clusters* e *grid* através do MATLAB *Distributed Computing Server*.
- Execução interativa e em *batch* de aplicações.

As aplicações desenvolvidas a serem executadas em GPUs NVIDIA® utilizando o PCT, são em geral, mais simples e rápidas do que utilizar a linguagem CUDA-C, isso ocorre porque os aspectos de exploração de paralelismo são realizados implicitamente pelo próprio PCT, além de otimizar a execução do código nos núcleos da GPU. Entretanto, a organização e o número de *threads* a serem executadas nos núcleos da GPU não podem ser gerenciados manualmente pelo programador [Little and Moler 2013].

3. Resultados

3.1. Ambiente de testes

Para avaliar os benefícios do ganho computacional obtido por meio do uso da programação paralela em GPU utilizando a integração CUDA-MATLAB através do *plugin* PCT disponibilizado pela MathWorks® foram realizados dois experimentos, o custo para resolução de grandes sistemas lineares e a resolução de equação de onda de segunda ordem através de métodos espectrais. Todos os testes foram realizados em um computador com processador Intel Core i5 3570 3,40GHz, 16 GB de memória RAM, com placa de vídeo NVIDIA® GeForce GTX 970, dispondo de 4 GB de memória e 1664 núcleos de processamento operando a 1.23GHz. Foi utilizado o MATLAB versão 2016a, juntamente com o PCT, onde os algoritmos foram implementados e executados.

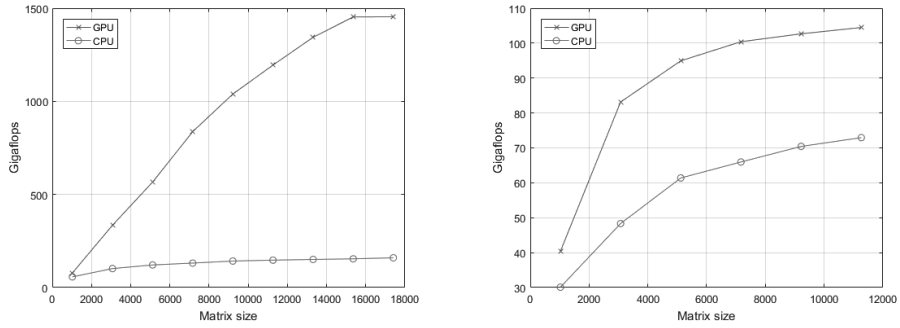
3.2. Sistemas de Equações Lineares

Neste experimento, foi analisada a resolução sistemas de equações lineares na GPU. Na resolução de um sistema linear, ou seja, na determinação de X, utilizamos mais frequentemente o operador barra invertida ($\backslash$), conhecido como $mldivide$, isto é:

$$X = A \setminus B \tag{1}$$

Na resolução da Equação 1, queremos analisar o desempenho do operador barra invertida ($\backslash$) e não o custo de transferência de dados entre a CPU e GPU, muito menos o tempo que leva para criar uma matriz ou outros parâmetros. Portanto, separa-se a geração de dados da resolução do sistema linear e mede-se apenas o tempo que leva para resolvê-lo para diferentes tamanhos de matriz.

Na Figura 1(a) é exibido o desempenho da CPU e GPU da solução da Equação 1 para precisão simples e na Figura 1(b) é exibido o desempenho para precisão dupla. Pode observar que o número de operações de ponto flutuante por segundo na GPU se torna mais significativo em relação à CPU, à medida que aumenta-se o tamanho da matriz, tornando o desempenho da GPU mais eficiente em comparação à CPU.



(a) Precisão simples (b) Precisão dupla

Figura 1. Desempenho da solução da Equação 1 (a) precisão simples e (b) precisão dupla.

3.3. Equação de uma onda

Neste experimento resolve-se uma equação de onde de segunda ordem:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{2}$$

com limite $u = 0$. O algoritmo utilizado para resolver a Equação 2 combina um método espectral no espaço e um método de diferença finita central de segunda ordem no tempo. Especificamente, aplica o método espectral de Chebyshev, que usa polinômios de Chebyshev como funções de base [Trefethen 2000].

Na Figura 2 os resultados obtidos no qual mede-se a quantidade de tempo necessário para executar 50 etapas de tempo para tamanhos de grade de 64, 128, 512, 1024 e 2048, com escalas diferentes (linear e logarítmica), tanto para CPU e GPU, considerando inclusive o custo de transferência de dados entre a CPU e GPU.

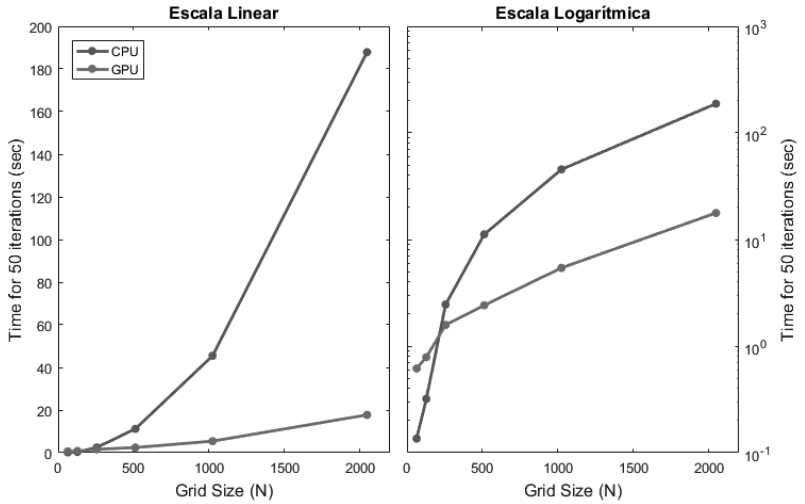


Figura 2. Comparação CPU e GPU para solução da equação de onda.

Os dois gráficos apresentados na Figura 2 são para os mesmos dados de referência e nota-se que a GPU mostra um desempenho muito melhor à medida que aumenta o tamanho dos dados. Para tamanhos de dados pequenos, a sobrecarga de transferência de dados da memória da CPU para a memória da GPU e vice-versa torna-se significativa e CPU mostra um melhor desempenho.

4. Conclusão

Este trabalho apresentou o uso da integração entre CUDA e o MATLAB através do textit-Parallel Computing Toolbox (PCT), *plugin* desenvolvido pela MathWorks®, permitindo que aplicações sejam aceleradas mais facilmente. O PCT disponibiliza uma forma mais eficiente para acelerar códigos desenvolvidos na linguagem MATLAB, executando-os em uma GPU. Os ganhos produzidos pelo PCT foram observados através da resolução de grandes sistemas lineares e de uma equação de onda de segunda ordem através de métodos espectrais, constatando uma superioridade da GPU em relação ao tempo computacional gasto para realização da mesma tarefa em uma CPU, obtendo um *speedup* da GPU em torno de 5x a 10x em comparação a implementação executada na CPU, demonstrando assim seu potencial na exploração do paralelismo e desempenho computacional.

Referências

Belleman, R. G., Bédorf, J., and Zwart, S. F. P. (2008). High performance direct gravitational n-body simulations on graphics processing units ii: An implementation in cuda. *New Astronomy*, 13(2):103–112.

Little, J. and Moler, C. (2013). Matlab gpu computing support for nvidia cuda-enabled gpus.

Nvidia, C. (2011). Nvidia cuda c programming guide. *Nvidia Corporation*, 120(18):8.

Trefethen, L. N. (2000). *Spectral methods in MATLAB*, volume 10. Siam.