Análise da Escalabilidade de um Programa Acelerado por GPU

Weverson dos Santos Gomes[1]

[1]DTEC - Departamento de Tecnologia
Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana, BA – Brasil

weversondsg@gmail.com

1. Introdução

Muitas aplicações são primeiramente programadas para rodar em um único núcleo. No campo da computação de alto desempenho, diferentes estratégias podem ser adotadas para acelerar cômputos quando diferentes recursos computacionais estão disponíveis.

A resolução de problemas computacionalmente densos, necessita de um tempo longo para a obtenção do resultado final. Uma técnica utilizada para acelerar o cômputo destes tipos de problemas é por meio da computação acelerada por GPU, que permite que trechos computacionalmente densos do código sejam executados pela placa de vídeo. Uma maneira de se conseguir isso é utilizando a biblioteca OpenCL.

O objetivo deste trabalho é analisar a escalabilidade de um programa acelerado por GPU para um problema computacionalmente denso. A análise será feita variando o tamanho do problema.

2. Fundamentação Teórica

Algumas questões importantes a se avaliar quando se trabalha com computação de alto desempenho são: É melhor deixar a aplicação serial ou paralela? Se é melhor paralelizar, até que ponto vale a pena aumentar o número de processadores? A minha aplicação está fazendo uso eficiente dos recursos disponíveis? A seguir veremos algumas métricas que podem ajudar a responder estas perguntas. Veremos também alguns conceitos importantes quando falamos sobre computação paralela com GPU.

2.1. Speed Up

Uma métrica importante para avaliar o desempenho de uma aplicação é o speedup.

O speedup pode ser definido como a relação entre o tempo gasto para executar uma tarefa com um único processador e o tempo gasto com N processadores. Representa o ganho efetivo em tempo para um dado aumento no número de unidades de processamento [Lantz 2015]. É dado como:

$$S = t1 / tN \quad (1)$$

2.2. OpenCL

O OpenCL (Open Computing Language) é o padrão aberto e isento de royalties para programação paralela multi-plataforma de diversos processadores encontrados em computadores pessoais, servidores, dispositivos móveis e plataformas embutidas. O OpenCL

melhora a velocidade e a capacidade de resposta de um amplo espectro de aplicações em inúmeras categorias de mercado, incluindo títulos de jogos e entretenimento, software científico e médico, ferramentas criativas profissionais, processamento visual e treinamento e inferência de redes neurais [Khronos 2017].

2.3. Computação acelerada por placas de vídeo

Computação acelerada por placas de vídeo é o uso de uma unidade de processamento gráfico (GPU, Graphics Processing Unit) juntamente com uma CPU para acelerar aplicativos de aprendizado profundo, análise e engenharia. Os aceleradores de placas de vídeo agora potencializam data centers de eficiência energética em laboratórios governamentais, universidades, corporações e empresas de médio e grande portes em todo o mundo. Elas desempenham um papel fundamental na aceleração de aplicativos em plataformas que variam de inteligência artificial até carros, drones e robôs [NVIDIA 2017].

2.3.1. Como as placas de vídeo aceleram os aplicativos de software

A computação acelerada por placas de vídeo libera porções do aplicativo com uso intenso de computação para a placa de vídeo, enquanto o restante do código ainda é executado na CPU, como mostra a figura 1. De uma perspectiva do usuário, os aplicativos simplesmente são executados muito mais rápido [NVIDIA 2017].

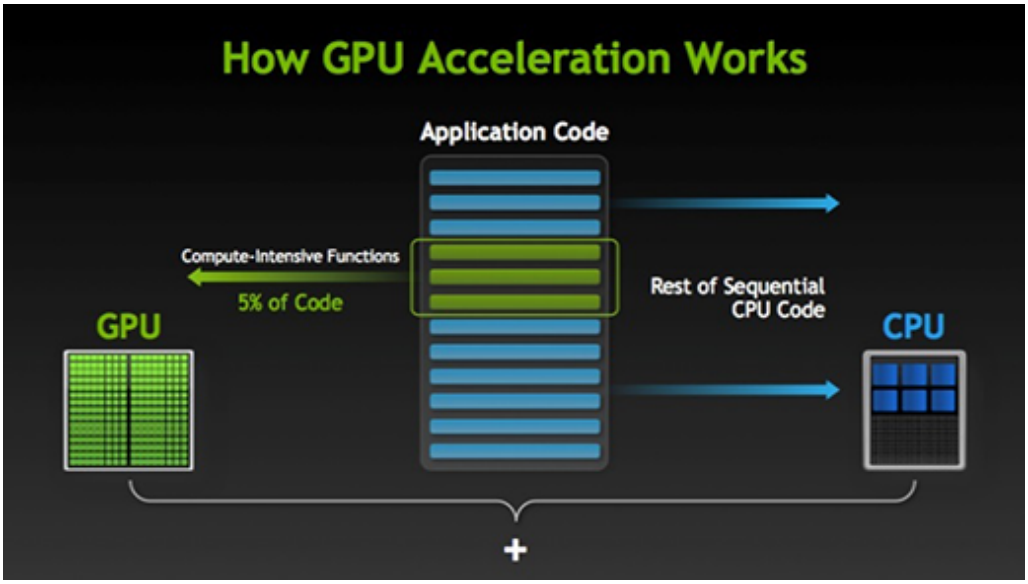

Figura 1. Como funciona um programa acelerado por GPU. Fonte: NVIDIA.

2.3.2. Desempenho da placa de vídeo versus da CPU

Uma forma fácil de entender a diferença entre uma placa de vídeo e uma CPU é comparar o modo como elas processam as tarefas. Uma CPU tem alguns núcleos otimizados para o processamento serial sequencial, enquanto uma placa de vídeo tem uma arquitetura paralela gigantesca que consiste em milhares de núcleos menores e mais eficientes criados

para lidar com múltiplas tarefas simultaneamente [NVIDIA 2017], como mostra a figura 2.

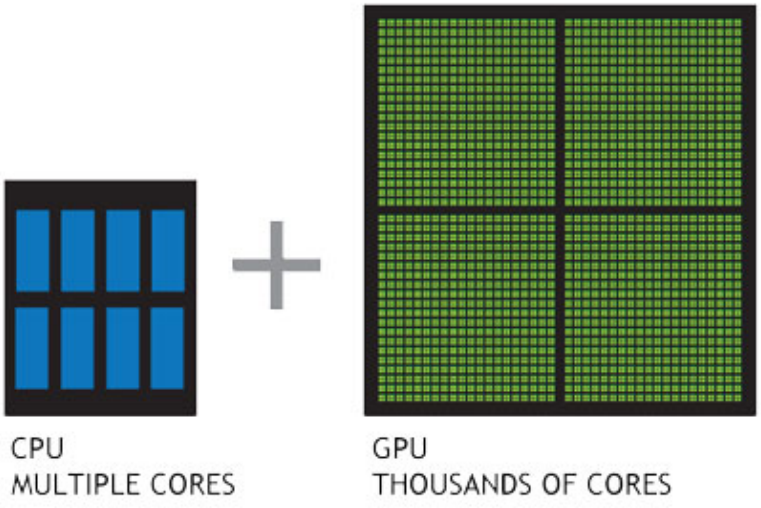

Figura 2. As placas de vídeo têm milhares de núcleos para processar cargas de trabalho paralelas de modo eficiente. Fonte: NVIDIA.

3. Metodologia

3.1. Implementação

Ambos os códigos foram implementados em linguagem C.

Primeiramente foi criado o código serial. A partir deste código foi selecionado o trecho mais computacionalmente denso, o qual consiste de um trecho de código que envolve 5 laços ”for”aninhados. Este trecho de código foi movido para um outro arquivo texto e, quando executado, o programa o envia para a GPU, desse modo a parte mais computacionalmente densa passa a ser executada fora da CPU. Seguindo o modelo mostrado na figura 1.

O código executado pela GPU faz uso de uma estrutura tridimensional na qual cada dimensão corresponde a um laço ”for”do programa serial. Essa estrutura tridimensional possui 17 mil células e cada uma executa os outros dois laços ”for”restantes.

3.2. Compilação

O programa serial foi compilado no gcc utilizando o argumento de otimização -O3, priorizando a redução do tempo de execução em relação ao uso de memória pelo programa gerado e ao tempo de compilação. Usou-se também o argumento gcc -Wall, habilitando todos os alertas do compilador para melhorar a qualidade do código e obter melhor desempenho. O programa acelerado por GPU foi compilado no nvcc, o compilador da NVIDIA.

3.3. Validação

Devido ao grande número de resultados, mais de 1 bilhão, a validação dos testes foi feita por amostragem, comparando os resultados do programa serial com os resultados do programa acelerado pela GPU.

3.4. Execução

O programa foi executado em um computador com sistema operacional Linux, 8GB de memória RAM, processador Intel Core i5 e GPU NVIDIA GT 710m com 96 núcleos CUDA.

Os dados iniciais para o cômputo são lidos de um arquivo texto. Neste arquivo cada linha contém novos dados de entrada. Para cada linha lida são realizadas 1.468.800.000 chamadas a uma função dX, a qual internamente possui um laço ”for”que executa 20 vezes. Isto resulta em um total de 29.376.000.000 de operações para cada linha lida. A equação calculada é:

$$x(t) = 2 * (A\sin(wt) - B\cos(wt)) + Et + \sum_{n=1}^{N} F_n e^{-n\gamma t} + G$$

Figura 3. Equação calculada dentro da função dX

Para cada linha de arquivo foram analisados os tempos de execução do programa utilizando 10, 20, 30 e 40 linhas do arquivo de entrada. Os tempos totais de execução são colhidos por meio do comando ”time”do Linux, inserido junto ao comando ”./”, que é usado para executar o arquivo. O programa recebe um único parâmetro, que se refere ao número de linhas que devem ser lidas do arquivo de entrada. Desse modo o comando para executar o programa via linha de comando é:

time ./rendezvous N

onde rendezvous é o nome do programa e N é o número de linhas do arquivo.

4. Resultados

As tabelas 1 e 2 mostram os os tempos de execução do programa serial e do programa acelerado por GPU, para 10, 20, 30 e 40 linhas de arquivo.

Tabela 1. Tempos de execução do programa serial por número de linhas de arquivo

Quantidade de linhas	10	20	30	40
Tempo de execução	7m48s	15m42s	23m16s	31m19s

Tabela 2. Tempos de execução do programa acelerado por GPU por número de linhas de arquivo

Quantidade de linhas	10	20	30	40
Tempo de execução	25,72s	50,64s	1m15s	1m40s

Com estes dados em mãos podemos obter o speedup do programa à medida que aumentamos o número de linhas.

A figura 4 mostra o speed up obtido para 10, 20, 30 e 40 linhas de arquivo.

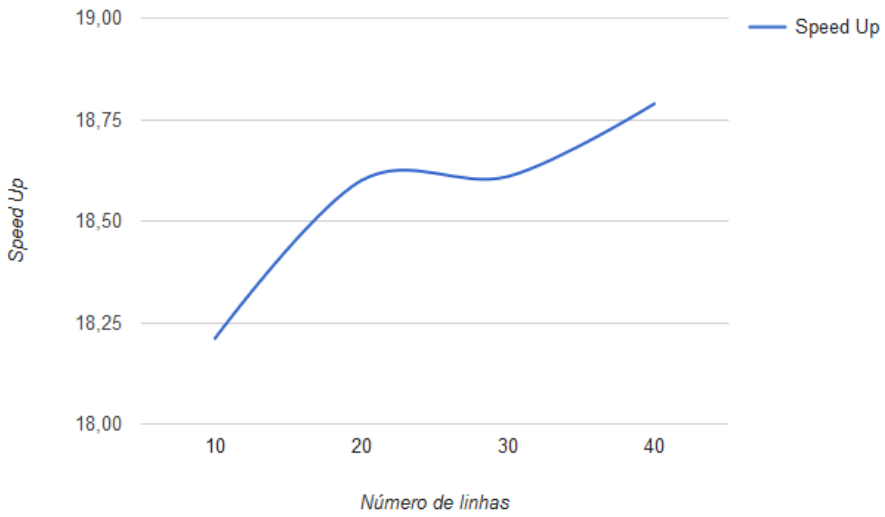

Figura 4. Curva de speed up

5. Considerações Finais

Neste trabalho foi feita uma análise comparativa do desempenho de dois programas, um programa serial e outro acelerado por GPU. Tal análise foi feita a partir do tempo necessário para a execução dos programas variando a quantidade de linhas lidas do arquivo de entrada.

Na análise foi possível observar que o speed up sofre pouca variação com o aumento do problema a ser resolvido, o que é um bom resultado e significa que a eficiência do uso dos recursos disponíveis é mantida.

Referências

Khronos (2017). The open standard for parallel programming of heterogeneous systems. Khronos Group.

Lantz, S. (2015). Scalability. Workshop: High Performance Computing on Stampede. Disponível em: https://www.cac.cornell.edu/education/training/StampedeJan2015/Scalability.pdf.

NVIDIA (2017). O que é computação acelerada por placas de vídeo? NVIDIA. Disponível em: http://www.nvidia.com.br/object/what-is-gpu-computing-br.html.