Uma Proposta para Caracterização do Consumo Energético de GPUs Programadas com OpenACC

Renan Moretto Bernardi, Guilherme Piêgas Koslovski

[1]LabP2D – Universidade do Estado de Santa Catarina (UDESC)

Resumo. *OpenACC facilitou a programação de GPUs e ganhou popularidade nos últimos anos. O presente trabalho propõe a caracterização do consumo energético de GPUs programadas com OpenACC, identificando o relacionamento entre desempenho e consumo. Para atingir o objetivo, um plano de experimentação é apresentado, bem como resultados preliminares indicando a influência no consumo energético oriunda dos parâmetros de compilação.*

1. Introdução

Com a evolução da tecnologia e o aumento da quantidade de dados gerados e consumidos pelos usuários, surge a necessidade de uma melhor utilização e desempenho dos recursos disponíveis. Nesse contexto, é criada a unidade de processamento gráfico (GPU, *Graphics Processing Unit*) com o intuito de processar primitivas gráficas em tempo real. Com capacidade de processamento de um único dispositivo podendo chegar a TeraFlops por segundo, a GPU fornece desempenho sem precedentes no processamento de partes intensivas de aplicações [Nesi et al. 2018]. Também utilizada em cálculos complexos de matemática e geometria devido a sua capacidade de processar vetores e matrizes, a GPU passou a ser utilizada para diferentes aplicações como processamento de imagens, inteligência artificial, cálculo numérico e visão computacional.

Em paralelo com o desenvolvimento tecnológico, proteger o meio ambiente e reduzir o consumo de energia é uma das responsabilidades requeridas dessa e das próximas gerações de computadores. Por este motivo, a computação verde é um movimento que ganhou destaque, tanto de empresas de desenvolvimento de *hardware* e *software* quanto de iniciativas governamentais. De acordo com o relatório da Empresa de Pesquisa Energética o consumo energético médio *per capita* do Brasil em 2016, é de 2.228 kWh/hab, já o supercomputador Summit, constituído por 2.282.544 cores, 27.648 GPUs e classificado como o supercomputador com maior desempenho da atualidade, tem eficiência energética de 1,39 TFLOPS/kW e com capacidade total de processamento podendo chegar a 122.300 TFLOPS, sendo que seu consumo energético equivale ao consumo de 14.300 habitantes aproximadamente [Top500.org 2018].

2. Revisão de Literatura e Motivação

Inicialmente a Seção 2.1 revisa os conceitos sobre GPU e OpenACC, enquanto a Seção 2.2 define o problema e apresenta alguns trabalhos relacionados.

2.1. GPU e OpenACC

As GPUs possuem arquitetura *Single Instruction Multiple Data* (SIMD), uma das classes da taxonomia de Flynn [Flynn 1972], na qual a mesma instrução é aplicada a diversos

dados. A arquitetura da GPU consiste de três tipos de memória, global, constante e compartilhada e processadores integrados aos dispositivos, os *Streaming Processors* (SP) e os *Streaming Multiprocessors* (SM), conectando-se aos outros componentes a partir da interface *PCI-Express* [Cook 2013]. Um SP, também conhecido como bloco na arquitetura da NVIDIA é responsável processamento de uma quantidade de *threads*, esta quantidade conhecida como *warp*, é responsável por executar e calcular sub-partes do problema a ser resolvido. Já no caso do SM, este tem como função o processamento total de um ou mais blocos e na arquitetura NVIDIA é conhecido como grade.

O modelo de programação, CUDA, faz uso das GPUs da NVIDIA, e foi desenvolvido para suportar várias linguagens de programação como C, C++ e Fortran. O CUDA permite que o programador defina funções a serem executadas em GPU, conhecidas como *kernels*, que tem como requisito a quantidade de *warps*, blocos e grades. Estes valores são definidos pelo desenvolvedor e devem ser baseados no problema alvo e nas especificações da GPU utilizada [Nesi et al. 2018], uma tarefa não trivial.

Por sua vez, o OpenACC é o modelo de programação que faz uso de diretivas de compiladores de alto nível, desenvolvido para abranger uma amplitude de arquiteturas, desenhado como uma plataforma independente de linguagem para a programação. É possível desenvolver apenas um único código fonte e este pode ser executado em uma variedade de dispositivos [NVIDIA Corporation 2018]. Com o OpenACC foi possível atenuar a complexidade de programação para a utilização de GPUs, permitindo ao desenvolvedor ignorar o paralelismo e abstraindo a transferência de dados do hospedeiro (CPU) para o dispositivo (GPU), focando no desenvolvimento do código sequencial e então adicionando as diretrizes de OpenACC em partes onde julga-se necessário.

Para a compilação de um código utilizando as diretivas de OpenACC são utilizados diversas *flags* de compilação. A *flag -acc* indica ao compilador a inclusão da biblioteca de *runtime* e habilita o reconhecimento dos *pragmas*, palavra reservada que indica uma região de paralelização, é possível a utilização da *flag -fast*, que habilita a otimização *inline*, que faz a remoção do *overhead* na chamada de processamento.

2.2. Consumo Energético de GPUs

A grande demanda de poder computacional influenciou no desenvolvimento das GPUs, tornando-as otimizadas para um alto desempenho, mesmo ao custo de um maior consumo energético. Deste modo, para a utilização de GPUs é necessário a realização de análises com relação ao custo benefício, verificando a linearidade e proporcionalidade do aumento no consumo energético com a redução no tempo de execução. As aplicações sendo executadas são também variáveis importantes na análise de custo benefício, no qual o consumo está diretamente relacionado com as rotinas sendo executadas.

Na literatura especializada, uma proposta utilizou *benchmarks* para avaliar o esforço de paralelismo, desempenho e consumo de energia [Memeti et al. 2017], comparando *benchmarks* baseados em OpenMP, OpenCL, OpenACC e CUDA. As conclusões da pesquisa mostram que o esforço de desenvolvimento é menor utilizando OpenACC se comparado a OpenCL e CUDA e que um menor tempo de execução das aplicações resultam em um menor consumo energético. Ainda, um estudo investigou a utilização de *Dynamic Voltage and Frequency* (DVFS) alterando a voltagem e frequência do processador durante o processamento com o intuito de economizar energia [Mei et al. 2013].

Dentre as diferentes abordagens analisadas, diminuir a voltagem dos cores da GPU pode reduzir o consumo de energia, assim como escalar a frequência de memória, mas encontrar as configurações ideais para o DVFS em GPU não é uma tarefa trivial.

3. Plano de Testes e Resultados Preliminares

Para a realização dos testes necessários para a caracterização do consumo energético de GPUs programadas com OpenACC, são definidos dois cenários. O primeiro compreende a execução e análise de diferentes implementações de multiplicações de matrizes em OpenACC, utilizando diferentes diretivas e métodos de implementação, juntamento com diferentes *flags* de compilação, e CUDA, utilizando diferentes métodos de otimização de multiplicação de matrizes. É fato que o processamento matricial é essencial para algoritmos desenvolvidos para GPU. Em um segundo momento, os *benchmarks* SPECC ACCEL e *Edinburgh Parallel Computing Centre* serão executados para representar múltiplas cargas de processamento. Especificamente para os testes utilizando diferentes implementações de multiplicação de matrizes é necessário uma análise para a escolha da melhor carga de trabalho, já que a escolha inadequada pode afetar possíveis conclusões no caso de estudo.

3.1. Ambiente de Experimentação

O computador utilizado para a realização das experimentações possui as seguintes configurações: Placa Mãe LENOVO Sharkbay, processador Intel i7 4770 3.4GHz, Core/*threads* 4/8, RAM DIMM 16GB 1600 MHz DDR3, NVIDIA Titan Xp 12GB, sistema operacional Ubuntu 17.04.

Para análise, são coletadas duas métricas: o tempo de execução para diferentes tipos de algoritmos juntamente com seu consumo energético. Para a comparação do tempo de execução, a coleta é feita em microssegundos e com os dados coletados é calculado a média de 10 execuções, enquanto para a análise do consumo energético são utilizados os dados coletados com o dispositivo de coleta WattsUp Pro.

3.2. Resultados Parciais

Nos testes realizados com OpenACC, as matrizes multiplicadas são de tamanho 1000×1000. As matrizes foram declaradas como variáveis *Double* e *Float* e a execução do algoritmo com as diretivas básicas, tem como diferencial a utilização das *flags -acc* e *-fast*. Os valores das matrizes foram definidos aleatoriamente.

Tabela 1. Resultados Parciais

Algoritmo	**Tempo de Execução** (μs)	**Consumo Total de Energia** (W)
Double -acc	4287,45	4811
Double -fast	3130,26	2497
Float -acc	2186,38	2537
Float -fast	712,82	698

Na Tabela 1 observa-se que as execuções com variáveis *Double* tem um tempo de execução superior comparado com a declaração com *Float*, devido a diferente capacidade de processamento em GPU de variáveis *Double* e *Float*. A utilização da *flag* de

compilação *-fast* apresenta um melhor desempenho em relação a *flag -acc* para ambos os tipos de variáveis. Nota-se também que o consumo total na utilização da *flag -fast*, é menor que utilizando a *flag -acc*. É possível perceber também que apesar do maior tempo de execução utilizando *Double -fast* o consumo total de energia é menor do que usando *Float -acc*, no qual mostra que a utilização de *flags* de otimização tem significância na redução do consumo energético.

4. Considerações e Trabalhos Futuros

Com o intuito de acelerar o processo de resolução de problemas complexos, as GPUs e ferramentas de programação como CUDA e OpenACC tem sido amplamente utilizadas. O OpenACC, através de suas diretivas de programação, permite que desenvolvedores inexperientes com programação paralela possam otimizar seus algoritmos, executando-os em GPU. Com a utilização adequada de diferentes diretivas e *flags* de compilação foi possível diminuir tanto o tempo de execução, quanto o consumo de energia gasto na execução.

A continuação para o trabalho tem enfoque no desenvolvimento dos algoritmos de otimização em CUDA e OpenACC, bem como a realização de testes e análise de resultados. Ainda, a execução dos testes com *benchmarks* será o foco na próxima etapa da pesquisa.

Agradecimentos: LabP2D, UDESC, FAPESC e NVIDIA.

Referências

Cook, S. (2013). *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Flynn, M. J. (1972). Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960.

Mei, X., Yung, L. S., Zhao, K., and Chu, X. (2013). A measurement study of gpu dvfs on energy conservation. In *Proceedings of the Workshop on Power-Aware Computing and Systems*, HotPower '13, pages 10:1–10:5, New York, NY, USA. ACM.

Memeti, S., Li, L., Pllana, S., Kolodziej, J., and Kessler, C. W. (2017). Benchmarking opencl, openacc, openmp, and CUDA: programming productivity, performance, and energy consumption. *CoRR*, abs/1704.05316.

Nesi, L. L., Pillon, M. A., de Assunção, M. D., and Koslovski, G. P. (2018). GPU-accelerated algorithms for allocating virtual infrastructure in cloud data centers. In *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2018)*.

NVIDIA Corporation (2018). *OpenACC Getting Started Guide*. NVIDIA Corporation, https://www.pgroup.com/resources/docs/18.5/pdf/openacc18_gs.pdf, 2018 edition.

Top500.org (2018). Top500 supercomputer. Acessado em: ago. 2018.