Custo energético em computação de alto desempenho

Paulo N. M. dos Anjos[1], **Alvaro L. Fazenda**[1]

[1]Campus São José dos Campos – Universidade Federal de São Paulo (UNIFESP)
São José dos Campos – SP – Brazil

{paulo.anjos,alvaro.fazenda}@unifesp.br

Abstract. *The development of new algorithms usually focuses on improving performance, with little consideration given to environmental impact and energy cost associated with their execution. However, this topic has been receiving increased attention recently. This work aims to objectively demonstrate the energy consumption of commonly found programming patterns in computationally intensive scientific programs. The energy consumption will be measured using software through the RAPL interface, while other performance measures will utilize the number of operations executed and elapsed time. Variations will be made in the number of threads used and compilation options to identify the impacts these changes have on energy efficiency.*

Resumo. *O desenvolvimento de novos algoritmos geralmente tem foco na melhoria de desempenho, sendo raramente dada qualquer importância ao impacto ambiental e custo energético em função de sua execução, porém, esse tópico vem ganhando maior atenção recentemente. Este trabalho tem como objetivo mostrar de forma objetiva o consumo energético de padrões de programação comumente presentes em programas científicos demandantes de processamento de alto desempenho. A medida do consumo energético será feita por softwares por meio da interface RAPL, e as demais medidas de desempenho utilizarão a quantidade de operações executadas e o tempo decorrido. Serão feitas variações na quantidade de threads utilizadas e nas opções de compilação, buscando identificar quais os impactos que estas mudanças causam na eficiência energética.*

1. Introdução

Um dos problemas que vem ganhando tração mundialmente é o aumento da produção de aparelhos elétricos e o desenvolvimento de novas tecnologias sem dar a devida importância ao impacto ambiental causado. De acordo com um estudo de 2015, estima-se gastos em torno de 250 bilhões de dólares ao ano para manter os computadores ligados pelo globo e que, deste total, apenas 15% era utilizado para computação [Anwar et al. 2013] e o resto apenas para mantê-los ligados, porém inativos [Salama 2020].

Logo, é perceptível o descaso com os recursos utilizados para a manutenção deste estado, onde há um desperdício que poderia ser facilmente evitado em várias etapas, desde a concepção, produção, uso e descarte, e que devemos dar a devida importância, buscando descobrir formas de otimizar esses processos, reduzindo o impacto ambiental.

A computação verde (*green computing*) surge como um meio para enfrentar os desafios ambientais associados ao crescente consumo de energia, incluindo a área de processamento de alto desempenho. A lista Green500[1] é um exemplo de busca por desenvolvimento de software e hardware buscando o melhor custo e benefício entre desempenho e consumo energético. Na edição de junho de 2022, o supercomputador denominado Frontier (*Hewlett Packard Enterprise Frontier* ou OLCF-5, localizado no *Oak Ridge Leadership Computing Facility* nos EUA) ocupava tanto a primeira posição em desempenho no Top500[2] quanto a segunda posição no Green500.

2. Objetivo e metodologia

O objetivo deste trabalho é medir desempenho e custo energético de códigos-fonte que representam padrões comuns em soluções numéricas aplicadas à Computação Científica, sendo, consequentemente, muito utilizados em Processamento de Alto Desempenho. Os experimentos serão conduzidos variando os níveis de otimização do compilador e a quantidade de *threads* utilizadas de forma concorrente, para identificar quais os efeitos dessas variações para a eficiência energética, bem como as melhores combinações possíveis de opções em relação às métricas utilizadas. Como métricas, serão usadas o tempo de processamento, o custo energético total e relativo por unidade de tempo, além da métrica *MFLOPs/Watt* (milhões de operações numéricas com números reais de ponto flutuante por segundo, sobre o custo energético em Watt).

2.1. Padrões de programação

Foram selecionados diferentes técnicas e padrões de programação comumente utilizadas em programas adaptados ao uso de recursos de processamento de alto desempenho, que se assemelham aos programas científicos que demandam muitos recursos computacionais de processamento, sendo eles:

- *Map*: valores de uma coleção de entrada sofrem uma alteração de acordo com uma função e são individualmente e independentemente mapeados para uma saída específica [University of California 2021];
- *Reduction*: padrão para reduzir um conjunto de entradas, normalmente utilizando-se de operadores associativos e comutativos, onde a ordem não mudará o resultado [University of California 2021];
- *Stencil*: generalização do *Map*, na qual a função ainda mapeia a entrada para uma saída, porém levando em consideração a vizinhança da entrada [Malony 2014].

2.2. Métricas de desempenho e eficiência energética

A principal métrica será o desempenho por *watt*, onde o desempenho será medido em FLOPS (*floating point operations per second*), e *watt* a unidade padrão de energia em *joules* por segundo. As métricas para avaliar a eficiência energética dos programas serão baseadas na quantidade de operações que são realizadas e a energia necessária para executá-las, o qual é também utilizado pela bem conhecida lista *Green500*.

O custo energético pelo processador será obtido através de uma funcionalidade chamada de RAPL (*Running Average Power Limit*, em português, Limite de Energia

[1]https://www.top500.org/lists/green500/
[2]https://www.top500.org/lists/top500/

Médio de Execução). Uma interface desenvolvida pela Intel que utiliza MSRs (*Model-Specific Registers*, em português, registros de modelo específico) para armazenar o uso e gasto energético de vários domínios relacionados ao processador [Pandruvada 2014], como pacote total *cores* (pp0), *igpu* (pp1), memórias. O RAPL não é um medidor analógico, mas um modelo matemático por *software*, que estima o consumo energético do *hardware* por meio de contadores de desempenho e modelos de *I/O*, porém, as medidas obtidas são muito próximas dos valores reais [Rotem et al. 2012].

3. Desenvolvimento e resultados

A partir da fundamentação apresentada foram implementados programas em linguagem C++ para os padrões citados, utilizando programação aderente ao padrão OpenMP[3] para paralelizar, o qual também foi instrumentado para realizar a medição do tempo de execução e adquirir os valores dos registradores do RAPL, de forma a medir o custo energético do sistema durante a execução.

Vários testes foram realizados em diferentes configurações de otimização de compilação, bem como alterando a quantidade de *threads* que serão utilizadas no código paralelo em um computador utilizando um processador i7-9750H (6 núcleos e 12 *threads*) e dois pentes de memória KF432S20IB/8, sendo executado no sistema operacional Linux Mint 21.1 e compilados utilizando o g++ 11.3.0. Os resultados obtidos para os algoritmos testados com um *array* preenchido por valores reais de ponto flutuante de dupla precisão (*double*) com tamanho 100x100, com 5 milhões de iterações, estão presentes nas Figuras 1 e 2. As citadas Figuras possibilitam analisar a eficiência paralela e energética do algoritmo e sua escalabilidade. Em alguns casos, é notada uma perda de eficiência paralela e energética com a adição de mais *threads*, como ocorre ao utilizar-se da tecnologia de *hyperthreading*, em um processador com 6 núcleos (*cores*) reais e até 12 *threads* com codinome i7-9750H (Intel). Tais resultados sugerem que as perdas podem estar relacionadas a ocorrência de falta de afinidade de dados em memória cache (*cache miss*), trocas frequentes de contexto, alteração dinâmica na frequência do processador, entre outras, as quais serão analisados futuramente.

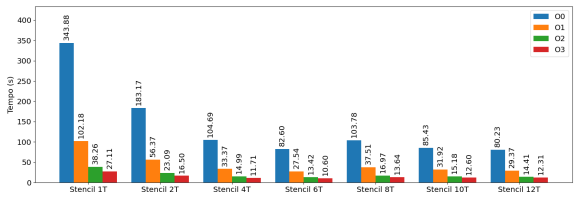

Figura 1. Tempo (s) x Algoritmo-threads para Stencil

Para os algoritmos testados, apenas o denominado *Reduction* apresentou ganhos em todas as configurações, enquanto os outros mostraram poucos ganhos com o aumento do número de *threads* em algumas configurações. Ao analisar a Figura 1 é possível perceber que a redução na eficiência energética presente na Figura 2, provém da baixa eficiência paralela e consequente baixa escalabilidade do algoritmo. As razões desse comportamento precisam ser melhor investigados em trabalhos futuros. O uso de opções mais agressiva de otimização do compilador mostrou-se benéfica para uma melhor eficiência energética.

[3]https://www.openmp.org/

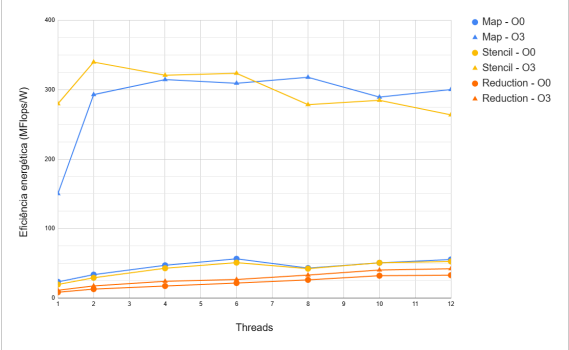

Figura 2. Eficiência energética (MFlops/W) x Threads

Por fim, este trabalho ressalta a importância da conscientização e adoção de práticas de computação verde na área de processamento de alto desempenho. A busca por soluções energeticamente mais eficiente contribui não apenas para a redução do impacto ambiental, mas também para a economia de recursos e para a viabilidade econômica a longo prazo dessas aplicações. A computação verde representa um passo importante na direção de um futuro mais sustentável e consciente. Como trabalhos futuros, destaca-se a expansão destas análises para outros padrões, outros processadores, além de testes com *benchmarks* conhecidos, bem como investigar as razões da baixa eficiência paralela encontrada em alguns testes.

Este trabalho foi parcialmente financiado pelo Projeto #2019/26702-8, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

Anwar, M., Qadri, S., and Sattar, A. (2013). Green computing and energy consumption issues in the modern age. *IOSR Journal of Computer Engineering*, 12(6):91–98.

Malony, A. D. (2014). Stencil pattern. Disponível em: `https://ipcc.cs.uoregon.edu/lectures/lecture-8-stencil.pdf`. Acesso em: 04/01/2022.

Pandruvada, S. (2014). Running average power limit. Disponível em: `https://01.org/blogs/2014/running-average-power-limit-%E2%80%93-rapl`. Acesso em: 06/01/2022.

Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., and Rajwan, D. (2012). Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro*, 32(2):20–27.

Salama, M. (2020). Green computing, a contribution to save the environment. Disponível em: `https://www.lancaster.ac.uk/data-science-of-the-natural-environment/blogs/green-computing-a-contribution-to-save-the-environment`. Acesso em: 06/01/2022.

University of California (2021). Parallel computation patterns (reduction). Disponível em: `https://www.cs.ucr.edu/~amazl001/teaching/cs147/S21/slides/07-Reduction.pdf`. Acesso em: 04/01/2022.