# Comparação de Consumo de Energia entre CPU e GPU em Computadores de Uso Doméstico

**Guilherme C. Salvati**[1], **Rafael Andreola**[1], **Samuel F. Ferrigo**[1]

[1]Universidade de Caxias do Sul (UCS)
Caxias do Sul – RS – Brasil

{gcsalvati, randreola2, sfferrig}@ucs.br

***Resumo.*** *Este trabalho avalia o desempenho e o consumo energético da multiplicação de matrizes quadradas utilizando paralelismo em CPU (OMP) e paralelismo heterogêneo CPU-GPU (CUDA) em computadores domésticos. A partir do algoritmo clássico com técnica de blocagem, analisaram-se os impactos em cada tipo de arquitetura. Experimentos foram realizados em três equipamentos domésticos, medindo-se tempo de execução e potência média do sistema para estimar a energia incremental. Os resultados mostram superioridade consistente da GPU tanto em desempenho quanto em energia total consumida.*

## 1. Introdução

A multiplicação de matrizes é um núcleo computacional fundamental, amplamente utilizado em computação científica, simulações numéricas, processamento de dados e aplicações de aprendizado de máquina. O algoritmo clássico para multiplicar duas matrizes quadradas $N \times N$ possui complexidade $O(N^3)$, o que resulta em alto tempo de execução e, consequentemente, em elevado consumo energético, especialmente quando executado em arquiteturas sequenciais [Abuzaid et al. 2015].

Com a evolução das arquiteturas de computadores, o uso de processamento paralelo tornou-se essencial para reduzir o tempo de execução dessas operações, aplicando técnicas de otimização como blocagem (*tiling*) e vetorização de Instrução Única sobre Múltiplos Dados (SIMD). Além disso, diferentes tecnologias de *hardware* possibilitam distintas formas de paralelismo, com benefícios próprios de desempenho e eficiência energética.

Grande parte dos estudos comparativos entre CPU e GPU concentra-se em ambientes de alto desempenho (HPC), frequentemente utilizando *hardware* dedicado. Entretanto, uma parcela significativa do desenvolvimento e da experimentação em computação científica é realizada em computadores pessoais equipados com *hardware* de mercado, como processadores multicore e GPUs tradicionais. Nesse contexto, torna-se relevante avaliar se os ganhos de desempenho e eficiência energética observados em ambientes dedicados também se mantêm consistentes em máquinas de uso geral.

Neste trabalho são analisadas as diferenças de desempenho, em termos de tempo de execução e consumo médio de energia, utilizando dois paradigmas de desenvolvimento paralelo: paralelismo em nível de CPU, com OMP, e paralelismo via GPU, com a plataforma CUDA da NVIDIA. Os experimentos foram conduzidos em três computadores pessoais com diferentes configurações de *hardware*. O objetivo é obter um resultado

empírico sobre a eficiência energética desses dois paradigmas em um contexto de *hardware* doméstico contemporâneo.

## 2. Referencial teórico

Embora o algoritmo clássico de multiplicação de duas matrizes quadradas $N \times N$ apresente complexidade $O(N^3)$, na prática o tempo de execução e a energia consumida não são determinados apenas pela contagem de operações, mas também pelo custo de movimentação de dados ao longo da hierarquia de memória. Assim, mesmo mantendo o mesmo problema e o mesmo paralelismo disponível, a forma como os dados são acessados e reutilizados pode alterar significativamente o desempenho e a eficiência energética, especialmente em cargas de trabalho com repetidas multiplicações e grande volume de dados [Abuzaid et al. 2015, Ibrahim 2020].

Em arquiteturas modernas, a hierarquia de memória (registradores, múltiplos níveis de *cache*, TLB e memória principal) e suas limitações de largura de banda e latência tornam a localidade de dados um fator crítico. Estratégias como *tiling* são frequentemente empregadas para aumentar a reutilização de dados em níveis mais rápidos da hierarquia e reduzir faltas de *cache* [Goto and van de Geijn 2008]. Em CPUs, o paralelismo combina múltiplos núcleos e vetorização SIMD, enquanto GPUs exploram execução massivamente paralela e memórias *on-chip* para aumentar a concorrência e a taxa de processamento [Chetlur et al. 2008]. Mesmo em computadores pessoais equipados com *hardware* de mercado, esses princípios arquiteturais continuam determinando o comportamento de desempenho e consumo energético.

Sob a ótica energética, otimizações que diminuem o tempo podem reduzir energia total, mas frequentemente elevam a potência instantânea. O resultado final depende do equilíbrio entre maior *throughput*, custos de memória e o comportamento do subsistema de execução vetorial e *multicore* [Goto and van de Geijn 2008, Ren and Suda 2020].

Assim, a análise comparativa entre CPU e GPU deve considerar simultaneamente tempo de execução e energia total consumida. Como a energia pode ser interpretada como a potência integrada ao longo do tempo, reduções no tempo de execução tendem a reduzir a energia total, desde que o aumento de potência média não seja proporcionalmente maior [Ibrahim 2020].

## 3. Método Experimental

Para realizar a comparação de tempo de execução e consumo de energia, foram desenvolvidas duas implementações do algoritmo clássico de multiplicação de matrizes quadradas utilizando *tiling*, com o objetivo de melhorar a localidade de dados e reduzir o impacto de faltas de *cache*. A primeira implementação explora paralelismo em nível de CPU por meio de OMP, enquanto a segunda utiliza paralelismo heterogêneo CPU-GPU com a plataforma CUDA. Os experimentos foram realizados em três computadores pessoais com diferentes configurações de *hardware*, conforme apresentado na Tabela 1.

Para obtenção dos tempos, foram realizadas três execuções do algoritmo para os tamanhos de matriz $2000x2000$, $3000x3000$ e $4000x4000$, cada uma trinta vezes para obtenção do tempo médio de execução em cada computador. A medição de consumo energético foi realizada utilizando um wattímetro externo modelo Xisfix FE-83[1],

[1]Maiores informações sobre o equipamento disponíveis em: https://docs.xisfix.com.br/fe-83/manual.pdf

**Tabela 1. Computadores Pessoais Utilizados**

| | PC1 | PC2 | PC3 |
|---|---|---|---|
| CPU | AMD Ryzen 5 1600 6-Core | AMD Ryzen 5 5600 6-Core | AMD Ryzen 9 7900X 12-Core |
| GPU | NVIDIA GeForce GTX 1060 6GB | NVIDIA GeForce RTX 3060 8GB | NVIDIA GeForce RTX 4060 8GB |
| RAM | 2 x 16 GB DDR4 1666 MHz | 2 x 16 GB DDR4 2600 MHz | 2 x 32 GB DDR5 2600 MHz |
| Sistema Operacional | Windows 10 Pro 64-bit | Windows 11 Pro 64-bit | Windows 11 Pro 64-bit |

registrando-se a potência média do sistema em repouso e durante a execução de cada versão do algoritmo. Para reduzir variações instantâneas, foram obtidas dez leituras de potência, espaçadas em intervalos de 10 segundos. A potência incremental foi calculada como a diferença entre a potência em execução e a potência em repouso, definida por $\Delta P = P_{\text{exec}} - P_{\text{rep}}$. A energia estimada foi obtida pelo produto entre a potência incremental e o tempo total de execução, estimada por $E = \Delta P \cdot t$, em joules.

## 4. Resultados

Realizando as operações conforme descrito na metodologia obtiveram-se os valores médios apresentados na Tabela 2[2]. Observa-se que, para todos os tamanhos de matriz e em todas as configurações testadas, a implementação em GPU apresentou menor tempo de execução quando comparada à versão paralela em CPU.

**Tabela 2. Resultados Obtidos**

| | PC1 | | PC2 | | PC3 | |
|---|---|---|---|---|---|---|
| | CPU OMP | GPU CUDA | CPU OMP | GPU CUDA | CPU OMP | GPU CUDA |
| Tempo (s) $2000x2000$ | 0.219 ±0.0136 | 0.033 ±0.0001 | 0.095 ±0.0054 | 0.018 ±0.0002 | 0.050 ±0.0047 | 0.014 ±0.0003 |
| Tempo (s) $3000x3000$ | 0.705 ±0.0150 | 0.091 ±0.0042 | 0.332 ±0.0132 | 0.059 ±0.0012 | 0.173 ±0.0072 | 0.051 ±0.0005 |
| Tempo (s) $4000x4000$ | 1.652 ±0.0501 | 0.215 ±0.0075 | 0.744 ±0.0165 | 0.132 ±0.0028 | 0.363 ±0.0072 | 0.118 ±0.0022 |
| Potência em uso (W) | 152.45 | 162.70 | 162.80 | 214.45 | 189.70 | 206.90 |
| Potência em repouso (W) | 82.66 | | 96,04 | | 122,82 | |

Como exemplo, no PC1 obteve-se $P_{\text{rep}} = 82{,}66$ W, $P_{\text{OMP}} = 152{,}45$ W e $P_{\text{CUDA}} = 162{,}70$ W, resultando em $\Delta P_{\text{OMP}} = 69{,}79$ W e $\Delta P_{\text{CUDA}} = 80{,}04$ W; os tempos totais foram $t_{\text{OMP}} = 0{,}219+0{,}705+1{,}652 = 2{,}576$ s e $t_{\text{CUDA}} = 0{,}033+0{,}091+0{,}215 = 0{,}339$ s, com energias $E_{\text{OMP}} \approx 80{,}40 \cdot 2{,}533 \approx 179{,}71$ J e $E_{\text{CUDA}} \approx 89{,}10 \cdot 0{,}351 \approx 27{,}17$ J.

[2]Resultados e algoritmos disponíveis em: https://github.com/Rafael-Andreola/matrix_processing_comparison_PCPD

Os resultados consolidados indicam que, embora a potência incremental da GPU seja, em alguns casos, superior à da CPU, a redução significativa no tempo de execução proporcionada pelo paralelismo massivo resulta em menor energia total consumida. O *speedup*[3] obtido foi de 7,6x no PC1, 5,6x no PC2 e 3,0x no PC3. Em termos energéticos, a GPU apresentou redução aproximada de 6,6x, 3,2x e 2,5x, respectivamente.

Além disso, observa-se que o aumento do *speedup* não implica necessariamente redução proporcional do consumo energético, uma vez que a energia total depende tanto do tempo de execução quanto da potência incremental do sistema. Em algumas configurações, a GPU apresentou potência superior à da CPU, mas compensou esse aumento por meio de redução significativa no tempo total. Isso demonstra que a eficiência energética decorre predominantemente da maior taxa de processamento, e não da diminuição da potência instantânea.

Esses resultados mostram que, mesmo em computadores pessoais equipados com *hardware* domésticos, a combinação de paralelismo massivo e blocagem mantém vantagem consistente tanto em desempenho quanto em eficiência energética.

## 5. Conclusão

Os resultados mostraram que, para o mesmo algoritmo de multiplicação de matrizes, o uso de GPU via CUDA entregou ganhos consistentes de desempenho em relação à execução paralela em CPU via OMP, com *speedup* de até 7,6x, além de apresentar redução de consumo energético de até 6,6x. Observa-se, contudo, que a potência incremental da GPU não foi necessariamente menor, reforçando que a economia de energia veio principalmente da redução do tempo total de execução, e não de menor potência. Como trabalho futuro, pretende-se avaliar de forma mais aprofundada a relação entre taxa de aceleração e redução de energia total nesse tipo de *hardware*, avaliando em quais condições o aumento de desempenho é acompanhado por maior eficiência energética, uma vez que a potência incremental pode crescer significativamente em algumas configurações.

## Referências


Abuzaid, F., Hadjis, S., Zhang, C., and Ré, C. (2015). Caffe con troll: Shallow ideas to speed up deep learning. *DanaC'15: Proceedings of the Fourth Workshop on Data analytics in the Cloud*.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B. C., and Shelhamer, E. (2008). cudnn: Efficient primitives for deep learning.

Goto, K. and van de Geijn, R. A. (2008). Anatomy of high-performance matrix multiplication. *ACM Transactions on Mathematical Software*.

Ibrahim, K. Z. (2020). Code development of high-performance applications for power-efficient architectures. In Ahmad, I. and Ranka, S., editors, *Handbook of Energy-Aware and Green Computing - Two Volume Set*. Chapman & Hall.

Ren, D. Q. and Suda, R. (2020). Energy-aware simd algorithm design on gpu and multicore architectures. In Ahmad, I. and Ranka, S., editors, *Handbook of Energy-Aware and Green Computing - Two Volume Set*. Chapman & Hall.


[3]Neste caso, considerou-se como tempo de referência para realização do cálculo de *speedup* a execução paralelizada da CPU em cada um dos cenários.