Paralelização de uma Aplicação de Simulação de Escoamento de Fluidos em Plano Inclinado*

Mariana Rodrigues Padilha†, Claudio Schepke

[1]Laboratório de Estudos Avançados em Computação (LEA)
Universidade Federal do Pampa (UNIPAMPA) – Alegrete – RS – Brasil

{marianarp.aluno,claudioschepke}@unipampa.edu.br

Resumo. *Este trabalho apresenta a refatoração e paralelização de um solver CFD para execução eficiente em arquiteturas heterogêneas. Foram avaliados diferentes modelos de programação paralela, incluindo OpenMP, OpenACC e CUDA. Os resultados indicam que abordagens baseadas em diretivas apresentam bom desempenho em ambientes multi-core, enquanto implementações em GPU oferecem ganhos expressivos para domínios computacionais maiores.*

1. Introdução

Este trabalho aborda a simulação do escoamento de fluido em um plano inclinado contendo uma região de meio poroso, um problema relevante em aplicações ambientais, como infiltração no solo e escoamento superficial[Liritzis 2023]. Computacionalmente, a heterogeneidade do domínio e as condições de contorno impactam o desempenho das abordagens paralelas.

A modelagem é baseada nas equações de Navier–Stokes[Kundu et al. 2002] e Darcy–Forchheimer[Cornelissen 2016]. Essas características aumentam a complexidade computacional e impactam diretamente o desempenho das abordagens paralelas., resolvidas numericamente pelo método dos volumes finitos em malhas estruturadas. O custo computacional é dominado por operações de estêncil, caracterizadas por alta demanda de memória e grande número de iterações.

Com a crescente adoção de arquiteturas heterogêneas, torna-se essencial adaptar códigos científicos para explorar CPUs multi-core e GPUs [Silva et al. 2023]. Neste contexto, este trabalho investiga a paralelização de um solver CFD utilizando OpenMP, OpenACC e CUDA, com o objetivo de analisar o desempenho dessas abordagens.

2. Metodologia

A aplicação realiza a simulação bidimensional do escoamento em um domínio composto por duas regiões: fluxo livre (superior) e meio poroso (inferior) [Schepke et al. 2025]. A discretização espacial é feita pelo método dos volumes finitos em malha estruturada[Ferziger et al. 2020], com avanço temporal iterativo até a convergência. As equações resolvidas são baseadas em operações de estêncil, incluindo quantidade de movimento, correção de pressão, energia e transporte escalar.

*Trabalho parcialmente financiado pelo Edital CNPq: Projeto 135963/2023-0.
†Bolsista PROBIC/FAPERGS 2025

Os experimentos foram conduzidos em uma arquitetura heterogênea cujas principais características são: CPU i9-13900HX (8P+16E) e GPU RTX 4060 com 8 GB de memória. Para a avaliação de desempenho, foram considerados três tamanhos de domínio: Case 1 (51 × 84), Case 2 (102 × 169) e Case 3 (153 × 254), com simulação no intervalo de 0 a 1 segundo e passo de tempo $\Delta t = 0.005$ s.

Em cada passo de tempo, são realizadas iterações internas até que os resíduos das equações resolvidas pelas rotinas sejam inferiores a 10^{-4}. Caso esse critério não seja atingido após 800.000 iterações, a execução é interrompida, indicando problemas de convergência numérica.

3. Implementação Paralela

Após a análise do código original, foi realizada uma refatoração para melhorar a organização e facilitar a paralelização, incluindo reorganização das estruturas de dados, remoção de chamadas em laços internos e separação entre pontos internos e de contorno. A paralelização foi realizada com OpenMP, OpenACC e CUDA. As implementações foram compiladas com NVCC (CUDA) e compiladores do NVIDIA HPC SDK (OpenMP e OpenACC).

Em OpenMP, foram avaliadas diferentes configurações de threads, além do uso de `do concurrent` para explorar paralelismo implícito.

Nas implementações em GPU, os dados são transferidos da CPU, antes da execução, e copiados de volta ao final para análise. Com OpenACC utiliza-se diretivas para paralelizar regiões do código e gerenciar a movimentação de dados, enquanto CUDA permite o controle explícito da execução na GPU. Na versão em CUDA, foi adotada uma configuração de blocos de 16×16 threads, com mapeamento bidimensional (uma thread por célula da malha) e organização em 27 kernels, correspondentes às diferentes etapas do solver, buscando um equilíbrio entre ocupação e eficiência de acesso à memória.

4. Resultados Experimentais

Os experimentos foram realizados em uma arquitetura heterogênea contendo CPU multicore e GPU NVIDIA, conforme descrito anteriormente. Para avaliar o desempenho das diferentes implementações, foram considerados três tamanhos de domínio computacional: Case 1, Case 2 e Case 3, conforme apresentado na Figura 1.

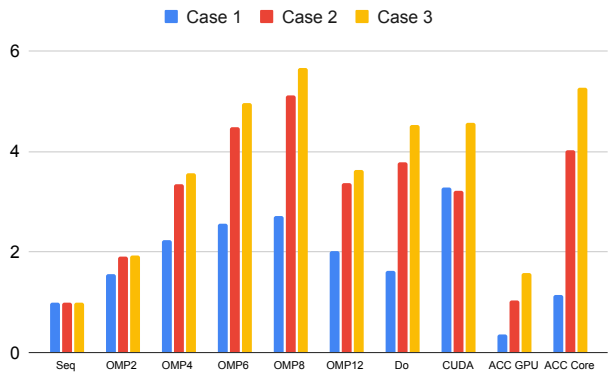

Figura 1. Speedup obtido para os casos 1, 2 e 3 nas implementações sequencial, OpenMP, Do Concurrent, CUDA e OpenACC.

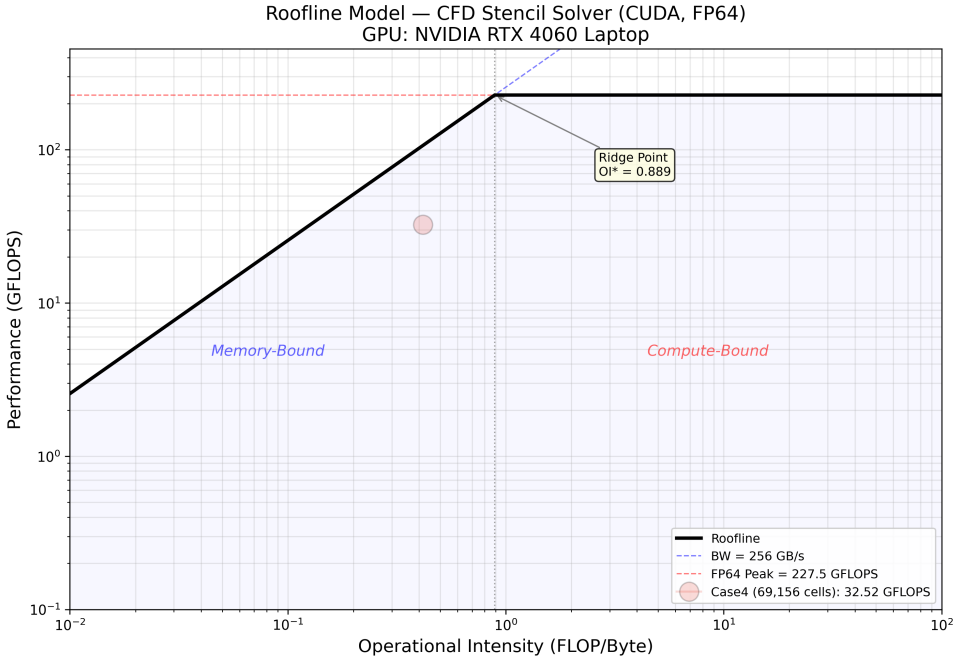

Figura 2. Modelo *Roofline* para a implementação em CUDA em GPU

Tabela 1. Distribuição do tempo de execução entre os principais kernels CUDA.

Kernel	Tempo total (s)	Porcentagem
solve_U	2200.7	34.77%
solve_V	2249.3	35.53%
solve_P	255.0	4.03%
solve_Z	793.0	12.53%
solve_C	832.2	13.15%
Tempo total de Kernel	6330.3	100%
Tempo total de execução	6453.0	—
Overhead	122.8	1.90%

A paralelização com OpenMP melhora o desempenho em relação à versão sequencial; porém, apresenta saturação a partir de 8 threads devido à limitação de largura de banda de memória.

As implementações em GPU obtêm melhores resultados para domínios maiores, com destaque para o $Case3$. Entre as abordagens avaliadas, CUDA apresenta o melhor desempenho devido ao maior controle sobre o paralelismo e ao acesso à memória.

A análise pelo modelo *Roofline* na Figura 2 indica uma intensidade operacional de aproximadamente 0,42 FLOP/Byte, caracterizando a aplicação como *memory-bound*. Considerando os limites da arquitetura, o desempenho máximo teórico é de cerca de 106,8 GFLOP/s, enquanto a implementação em CUDA atingiu aproximadamente 32,5 GFLOP/s, correspondendo a cerca de 30% desse limite.

A Tabela 1 apresenta a distribuição do tempo de execução dos kernels CUDA, com as rotinas principais concentrando mais de 70% do custo computacional. Esse com-

portamento está associado ao padrão de acesso à memória típico de operações de estêncil, incluindo acessos parcialmente não coalescidos. O uso de memória unificada simplifica o gerenciamento de dados, mas introduz *overhead* devido à migração de páginas. De forma geral, os resultados indicam que o desempenho é limitado pelo acesso à memória, sugerindo a necessidade de otimizações focadas na redução do tráfego e na melhoria da localidade dos dados.

O código-fonte da implementação está disponível publicamente para fins de reprodutibilidade em: https://anonymous.4open.science/r/simulacao-de-meios-porosos-636E

5. Conclusão

Este trabalho apresentou a refatoração e paralelização de uma aplicação de dinâmica dos fluidos computacional baseada em operações de estêncil. Diferentes modelos de programação paralela foram avaliados com o objetivo de analisar seu impacto no desempenho da aplicação em arquiteturas modernas.

Os resultados demonstraram que abordagens baseadas em diretivas, como OpenMP e OpenACC, permitem obter ganhos de desempenho relevantes com esforço de implementação relativamente reduzido. Por outro lado, a programação explícita utilizando CUDA oferece maior controle sobre o paralelismo da aplicação, possibilitando melhor exploração da arquitetura da GPU em determinados cenários.

Como trabalhos futuros, pretende-se avaliar domínios computacionais maiores e investigar técnicas adicionais de otimização relacionadas ao acesso à memória e à organização das estruturas de dados. Além disso, pretende-se explorar novas abordagens de paralelização que permitam melhorar ainda mais a escalabilidade da aplicação em arquiteturas heterogêneas.

Referências

Cornelissen, P. (2016). *Coupled Free-Flow and Porous Media Flow: a Numerical and Experimental Investigation*. PhD thesis, Faculty of Geosciences - Utrecht University.

Ferziger, J. H., Perić, M., and Street, R. L. (2020). *Finite Volume Methods*, pages 81–110. Springer International Publishing, Cham.

Kundu, P., Cohen, I., and Hu, H. (2002). *Fluid Mechanics*. Acad. Press.

Liritzis, I. (2023). Disasters and Climatic Phenomena Today and in the Past. *Proceedings of the European Academy of Sciences and Arts*, 2(1).

Schepke, C., Spigolon, R. A., Rufino, J., Cristaldo, C. F. D. C., and Pizzolato, G. L. (2025). Contributions to accelerating a numerical simulation of free flow parallel to a porous plane. In *2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 226–233.

Silva, H. U. d., Lucca, N., Schepke, C., Oliveira, D. P. d., and Cristaldo, C. F. d. C. (2023). Parallel OpenMP and OpenACC Porous Media Simulation. *The Journal of Supercomputing*, 79(8):8425–8446.