

# Estudo e Análise de Processamento Paralelo em Simulações de Fluidodinâmica Computacional com OpenFOAM

Valesca Moura de Sousa<sup>1</sup>, Fernanda Gonçalves de Oliveira Passos<sup>2</sup>

<sup>1</sup>Instituto de Computação – Universidade Federal Fluminense (UFF) – Niterói – RJ

<sup>2</sup>Escola de Engenharia – Universidade Federal Fluminense (UFF) – Niterói – RJ

Niterói, 24 de fevereiro de 2021.

## RESUMO

*Este artigo tem o objetivo de analisar o desempenho dos métodos paralelos implementados na ferramenta de fluidodinâmica computacional, Open-FOAM. Esta análise é feita com base em dois casos de contextos e complexidades distintas. Os resultados mostram que os métodos paralelos avaliados da ferramenta não escalam e que é necessário um esforço específico de ajuste de parâmetros para haver convergência em casos complexos e paralelos.*

## 1. Introdução

A Fluidodinâmica Computacional (Computational fluid dynamics, CFD) é a análise de sistemas envolvendo escoamento de fluidos, transferência de calor e fenômenos associados [1]. Tal análise é de suma importância em diversas áreas, como física e engenharias e o OpenFOAM [2] é uma ferramenta bastante utilizada para este propósito. No entanto, as simulações CFD apresentam cálculos muito custosos mesmo para os computadores potentes que existem atualmente. Nesse contexto, a computação paralela é empregada e o OpenFOAM possui atualmente versões paralelas e distribuídas para melhorar a eficiência de programas CFD.

O objetivo deste trabalho<sup>1</sup> é analisar o desempenho dos métodos paralelos do OpenFOAM nos quesitos uso de memória e *speed-up*. Além disso, serão avaliados os resultados numéricos a fim de verificar como o particionamento do problema na implementação paralela influencia os resultados da simulação em relação à execução sequencial. Para tal, são utilizados dois experimentos: um monofásico e outro multifásico.

## 2. OpenFOAM e Trabalhos Relacionados

---

<sup>1</sup> Este trabalho recebeu apoio financeiro do Projeto RAVES/UFF/Petrobras.

O OpenFOAM [OpenFOAM 2019] é uma ferramenta CFD de código aberto – motivo pelo qual foi escolhida nesse estudo – implementada em C++. Sua implementação paralela se baseia no particionamento da malha entre os processos de modo a executar o algoritmo serial do solucionador iterativo. A cada iteração, é necessária a comunicação entre os processos para trocarem dados de alguns pontos da malha em comum. Para isso, o MPI – Message Passing Interface – é utilizado. Os métodos de particionamento de malha disponíveis no OpenFOAM e analisados neste trabalho são: 1) **Simple**: decomposição geométrica na qual o domínio é dividido em partes por direção no eixo x, y e z; 2) **Hierarchical**: similar ao simple, porém é possível especificar a ordem da divisão direcional; 3) **Scotch**: busca minimizar o número de limites da malha de cada processo.

Há diversos trabalhos que discutem o paralelismo da ferramenta OpenFOAM. Alguns deles indicam que o gargalo na escalabilidade dos solucionadores OpenFOAM está ligado à massiva comunicação MPI entre processadores relacionada tanto a forma de particionamento da malha [2], quanto ao solucionador iterativo selecionado [3]. Nesse contexto, existem autores que propuseram melhorias através de um novo método de particionamento estático a fim de melhorar desempenho e, também, convergência de fluxos multifásicos [4]. O trabalho em [5] propõe otimizações na comunicação de solucionadores iterativos.

### 3. Metodologia da Análise

Com o intuito de analisar o desempenho do OpenFOAM e a influência de seus métodos paralelos nos resultados numéricos, foram feitos experimentos com múltiplas medições de tempo de execução para cálculo do *speed-up*. Além disso, foi monitorada a utilização de memória durante as execuções seriais e paralelas. Nas simulações, foram avaliados três métodos paralelos da ferramenta: *simple*, *hierarchical* e *scotch*. Posteriormente, foi feita uma análise dos resultados numéricos da execução paralela para avaliar como o particionamento da malha com o método *scotch* pode influenciar os resultados da simulação em relação à execução sequencial. Os resultados analisados referem-se aos valores dos campos e, em ambos os casos, tais valores foram extraídos através do *singleGraph*, utilitário fornecido pela aplicação OpenFOAM, para 3 linhas onde foram observadas maiores movimentações da malha.

Os experimentos foram conduzidos em um cluster composto por 5 nós, cada um com um processador quadcore Intel(R) Core i7 860, com memória principal de 32 GB,

conectados por rede do tipo Gigabit Ethernet. Os casos de teste utilizados foram o PitzDaily, monofásico e permanente e, DamBreak, multifásico e transiente. As configurações aplicadas na preparação de ambos os casos se encontram disponíveis no diretório *tutorials* incluso na instalação do OpenFOAM versão 7. Ademais, as malhas foram refinadas em 16 vezes seus tamanhos originais, a fim de aumentar o processamento.

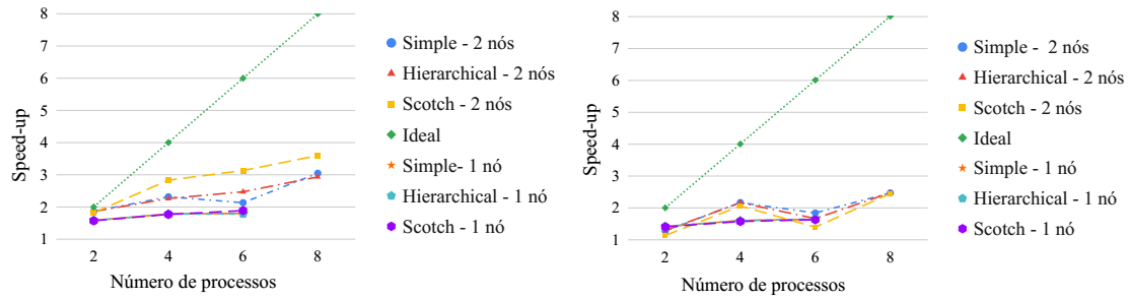
#### 4. Resultados

Para a análise de desempenho, o gráfico da Figura 1 mostra o *speed-up* obtido conforme aumento do número de processos em uma execução realizada em 1 e 2 nós do cluster. À esquerda, temos os valores para o PitzDaily e, à direita, para o DamBreak. De toda forma, percebemos que nenhum dos métodos paralelos utilizados escalam bem. Para o caso bifásico DamBreak, que é mais complexo, os valores de *speed-ups* foram ainda menores.

Além disso, constatou-se que o uso de memória durante a execução da aplicação não é ideal em nenhum dos casos. Por exemplo, uma execução serial do caso DamBreak utiliza cerca de 125 MB. Em uma execução paralela com 2 processos, um dos processos utiliza por volta de 106 MB. Portanto, é possível concluir que a execução paralela utiliza mais memória do que a serial. Isso pode levar a um aumento de *cache misses*.

Para a análise dos resultados numéricos das simulações, observou-se a diferença percentual dos valores obtidos no tempo de simulação dos experimentos em comparação ao valor absoluto numérico para campos como pressão e velocidade. Para tal, comparou-se os resultados de uma execução serial e paralela com 4 processos. No caso PitzDaily as diferenças não são representativas o suficiente tendo como maior valor de diferença relativa 0,1%. Em contrapartida, DamBreak apresenta diferenças consideráveis de até 22,9% em alguns pontos da malha quando comparadas as execuções seriais e paralelas.

**Figura 1.** *Speed-up* calculados sobre a média de tempos de 30 rodadas.



## 5. Conclusão

O presente estudo demonstrou que os métodos paralelos disponíveis na ferramenta OpenFOAM não apresentam boa escalabilidade. Tal problema pode estar associado à latência de comunicação MPI entre máquinas e à alta taxa de *cache misses*, que devem ser analisados futuramente. Pontualmente, e possível afirmar que o método *scotch* apresenta um melhor desempenho que os demais em casos simples, como o PitzDaily.

Em casos complexos, como o DamBreak, o desempenho e corretude numérica de tal método, em relação a simulação serial, não é adequado. Isto mostra que, para casos multifásicos, os parâmetros do método devem ser ajustados especificamente para as execuções paralelas. O estudo destes parâmetros é mais um trabalho futuro.

## 6. Atividades futuras

Analisar a alta taxa de *cache misses*, para verificar uma eventual associação de latência de MPI entre máquinas.

## 7. Referências Bibliográficas

- [1] Versteeg, H. K. and Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Editora Prentice Hall, 2º edition.
- [2] OpenFOAM (2019). The open source CFD toolbox. <http://www.openfoam.com>. Acessado em outubro de 2020.
- [3] Wang, M. et al. (2012). Performance analysis of the graph-partitioning algorithms used in OpenFOAM. *IEEE 5th Int. Conf. on Advanced Comp. Intelligence*, pages 99–104.
- [4] Rivera, O. and Furlinger, K. (2011). Parallel aspects of openfoam with large eddy simulations. In *2011 IEEE Int. Conf. on HPC and Communications*, pages 389–396.
- [5] Xu, L., Wang, M., Xu, X., Cui, C., and Yang, X. (2019). Dynamic mesh re-partitioning considering iterative convergence rate for multiphase flows in

OpenFOAM. *Concurrency and Computation: Practice and Experience*, 31(21):e5412.

- [6] Lin, Z. et al. (2018). Communication optimization for multiphase flow solver in the library of OpenFOAM. *Water*, 10 (10):1461.