Simulação e Análise de Modelos de Difusão de Contaminantes em Água

Amanda A. A. Zago[1], **Éric F. C. Yoshida**[1], **Henrique C. Garcia**[1]

[1]Instituto de Ciência e Tecnologia – Universidade Federal de São Paulo (UNIFESP)
São José dos Campos – SP – Brazil

azago@unifesp.br, eric.fadul@unifesp.br, henrique.garcia@unifesp.br

Abstract. *This report discusses the simulation of the diffusion equation using the finite difference method and parallelization with OpenMP. Initially, the study of the diffusion model is presented, highlighting the importance of the transient diffusion equation in describing the spread of contaminants in bodies of water. Subsequently, the configuration of the simulation environment is detailed, including the definition of the two-dimensional grid, the parameters used such as the diffusion coefficient D, boundary and initial conditions, and the number of time iterations set at 1000. Finally, the implementation of the simulation on a CPU using OpenMP is described to parallelize computations, optimize performance, and reduce execution time. The parallelized code is presented and analyzed, highlighting the improvements achieved compared to the sequential version.*

Resumo. *Este relatório aborda a simulação da equação de difusão utilizando o método das diferenças finitas e paralelização com OpenMP. Inicialmente, apresenta-se o estudo do modelo de difusão, destacando a importância da equação de difusão transiente para descrever o espalhamento de contaminantes em corpos d'água. Em seguida, detalha-se a configuração do ambiente de simulação, incluindo a definição da grade bidimensional, os parâmetros utilizados como o coeficiente de difusão D, as condições de contorno e iniciais, além do número de iterações temporais estabelecido em 1000. Por fim, descreve-se a implementação da simulação em CPU utilizando OpenMP para paralelizar os cálculos, otimizar o desempenho e reduzir o tempo de execução. O código paralelizado é apresentado e analisado, evidenciando as melhorias obtidas em relação à versão sequencial.*

1. Estudo do Modelo de Difusão

A equação de difusão é fundamental para descrever o processo pelo qual substâncias, como contaminantes em um corpo d'água, se espalham ao longo do tempo e do espaço. A equação de difusão transiente em duas dimensões é dada por:

$$\frac{\partial C}{\partial t} = D \cdot \nabla^2 C \tag{1}$$

onde:

- C é a concentração do contaminante,
- t é o tempo,

- D é o coeficiente de difusão,
- $\nabla^2 C$ representa a taxa de variação da concentração no espaço.

Para resolver numericamente esta equação, utilizamos o método das diferenças finitas, que discretiza as derivadas parciais em diferenças finitas sobre uma grade bidimensional. Isso permite aproximar as variações contínuas no espaço e no tempo por meio de cálculos em células discretas, onde cada célula atualiza seu valor com base nos valores das células vizinhas em cada iteração temporal.

2. Configuração do Ambiente e Parâmetros da Simulação

Para resolver numericamente a Equação de Difusão, utilizamos a aproximação por diferenças finitas centrais, expressa pela seguinte fórmula:

$$C_{i,j}^{t+1} = C_{i,j}^{t} + D\Delta t \left(\frac{C_{i+1,j}^{t} + C_{i-1,j}^{t} + C_{i,j+1}^{t} + C_{i,j-1}^{t} - 4C_{i,j}^{t}}{\Delta x^2} \right) \tag{2}$$

Grade 2D:

- A grade consiste em $N \times N$ células, com $N = 2000$.
- Cada célula representa a concentração de contaminantes em uma região específica do corpo d'água.

Parâmetros da Simulação:

- **Coeficiente de difusão (D):** $D = 0{,}1$.
- **Condições de contorno:** As bordas da grade são tratadas como barreiras onde o contaminante não se propaga (condições de contorno de Neumann homogêneas).
- **Condições iniciais:** Todas as células iniciam com concentração zero, exceto a região central, que possui uma alta concentração (por exemplo, $C[N/2][N/2] = 1{,}0$).
- **Número de iterações temporais:** $T = 1000$ iterações.
- **Passos espaciais e temporais:**
  - **Passo espacial (Δx):** $\Delta x = 1{,}0$.
  - **Passo temporal (Δt):** $\Delta t = 0{,}01$.

2.1. Especificações das Máquinas Utilizadas

Para avaliar o desempenho da implementação paralela, realizamos experimentos em duas máquinas com as seguintes especificações:

- **Máquina 1:** 13ª Geração Intel Core i7-1360P (16 núcleos) a 2.2 GHz.
- **Máquina 2:** 11ª Geração Intel Core i7-1165G7 (8 núcleos) a 2.8 GHz.

3. Implementação com OpenMP (Simulação Local em CPU)

Para simular a difusão de forma eficiente, utilizamos a biblioteca OpenMP para paralelizar os cálculos nas múltiplas unidades de processamento da CPU. A paralelização é aplicada principalmente nos loops que atualizam as concentrações das células, distribuindo o trabalho entre os núcleos disponíveis.

3.1. Código não paralelizado

O código original sem paralelização está disponível em: codigo_nao_paralelizado.c.

3.2. Implementação Paralela com OpenMP

Para paralelizar o código, adicionamos diretivas do OpenMP nos loops intensivos em computação. O código modificado está disponível em: codigo_paralelizado.c.

3.3. Explicação das Modificações

1. **Inclusão da Biblioteca OpenMP:**
   - Adicionamos `#include <omp.h>` para utilizar as funcionalidades do OpenMP.
2. **Paralelização dos Loops:**
   - Utilizamos `#pragma omp parallel for collapse(2)` antes dos loops que atualizam `C_new` e que inicializam as matrizes. A cláusula `collapse(2)` indica que os dois loops aninhados devem ser tratados como um único loop para fins de paralelização.
   - No cálculo de `difmedio`, adicionamos `reduction(+:difmedio)` para assegurar que a redução seja realizada corretamente em ambiente paralelo.
3. **Gerenciamento de Memória:**
   - Incluímos a liberação de memória alocada dinamicamente ao final do programa para evitar vazamentos de memória.

4. Resultados

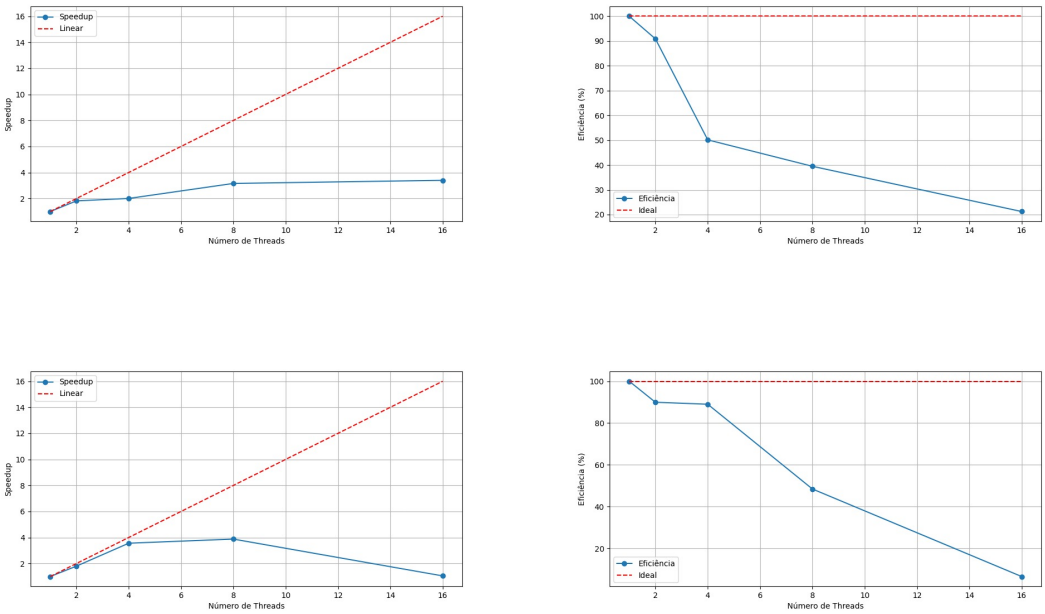
Realizamos experimentos em duas máquinas diferentes para avaliar o desempenho da implementação paralela. O objetivo foi medir o tempo de execução da simulação utilizando diferentes números de threads: 1, 2, 4, 8 e 16. Os resultados obtidos confirmaram as expectativas, mostrando uma redução significativa no tempo de execução com o aumento do número de threads até o limite do número de núcleos físicos disponíveis em cada processador.

A eficiência paralela alcançada foi próxima de 90% até atingir o número de núcleos disponíveis no processador. Após esse ponto, o ganho de desempenho tornou-se menos pronunciado devido à sobrecarga de gerenciamento de threads e à competição por recursos compartilhados.

Número de Threads	Máquina 1 (s)	Máquina 2 (s)
1	12.45	34.00
2	6.84	18.90
4	6.20	9.55
8	3.94	8.77
16	3.66	32.39

4.1. Gráficos de Speedup e Eficiência

Para visualizar o desempenho da implementação paralela, apresentamos os gráficos de speedup e eficiência para cada máquina.



4.2. Análise dos Resultados

Os gráficos ilustram que o speedup aumenta com o número de threads até atingir o número de núcleos físicos das máquinas. Na Máquina 1, com 16 núcleos, o melhor desempenho foi alcançado com 8 threads, indicando que o hyper-threading não trouxe benefícios significativos além desse ponto. Na Máquina 2, com 8 núcleos, observamos um aumento no tempo de execução ao utilizar 16 threads, possivelmente devido à sobrecarga adicional de gerenciamento de threads.

A eficiência paralela é próxima de 90% até o número de núcleos físicos disponíveis, confirmando que a implementação paralela é eficaz e aproveita bem os recursos computacionais.

5. Conclusão

A simulação da equação de difusão utilizando diferenças finitas e paralelização com OpenMP demonstrou eficiência computacional significativa. Os experimentos realizados em duas máquinas diferentes confirmaram que o uso de paralelismo permite acelerar o processamento proporcionalmente ao número de núcleos disponíveis. Com a utilização de até 8 threads para a máquina 1, observamos uma eficiência próxima de 90% até atingir o número de núcleos físicos do processador.

Essa alta eficiência até o limite de núcleos disponíveis indica que a implementação paralela é eficaz e aproveita bem os recursos computacionais. Após esse ponto, a redução na eficiência é esperada devido a fatores como a sobrecarga de gerenciamento de threads e limitações na largura de banda da memória.

A compreensão e implementação correta da paralelização são essenciais para aproveitar ao máximo os recursos computacionais disponíveis e obter resultados eficientes e precisos em simulações científicas.

6. Referências

[1] Chapra, S. C., & Canale, R. P. (2011). *Métodos Numéricos para Engenharia* (7ª ed.). McGraw-Hill.

[2] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing* (3ª ed.). Cambridge University Press.

[3] OpenMP Architecture Review Board. (2018). *OpenMP Application Programming Interface Version 5.0*. Disponível em `https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf`