Análise de Desempenho das Cláusulas collapse e schedule em OpenMP

Estudo de Caso: Simulação da Equação de Difusão 2D

Werbert Arles de Souza Barradas

Universidade Federal do Rio Grande do Norte (UFRN)
Disciplina de Programação Paralela - DCA3703

21 de setembro de 2025

Roteiro

Introdução: O Problema

- Simulações científicas, como a de dinâmica de fluidos, dependem da resolução numérica de equações diferenciais em malhas computacionais.
- O custo computacional está concentrado em **laços aninhados** que iteram sobre essas malhas.
- A paralelização eficiente desses laços é crucial para o desempenho.
- **Estudo de Caso:** A equação da difusão 2D, um modelo que utiliza o padrão de estêncil de cinco pontos, comum em computação científica.

Objetivos e Estratégias Avaliadas

Objetivo

Analisar e comparar o impacto de diferentes estratégias de paralelização de laços aninhados oferecidas pelo OpenMP, com foco nas cláusulas `collapse` e `schedule`.

Versões Analisadas

1. **Serial:** Implementação sequencial para baseline e validação.
2. **Paralelismo Simples:** Uso de `#pragma omp parallel for` apenas no laço mais externo.
3. **Paralelismo Otimizado:** Aplicação de `collapse(2)` para unificar os laços.
4. **Análise de Escalonamento:** Teste das políticas `schedule(static)`, `dynamic` e `guided` sobre a versão otimizada.

O Modelo Físico e a Discretização

Equação da Difusão Vetorial A simulação é governada pela equação:

$$\frac{\partial \vec{v}}{\partial t} = \nu \nabla^2 \vec{v}$$

Discretização Numérica

- Utiliza o método das diferenças finitas em uma grade 2D.
- O operador Laplaciano (∇^2) é aproximado por um **estêncil de cinco pontos**.
- A atualização de cada ponto (i, j) depende de seus quatro vizinhos diretos.

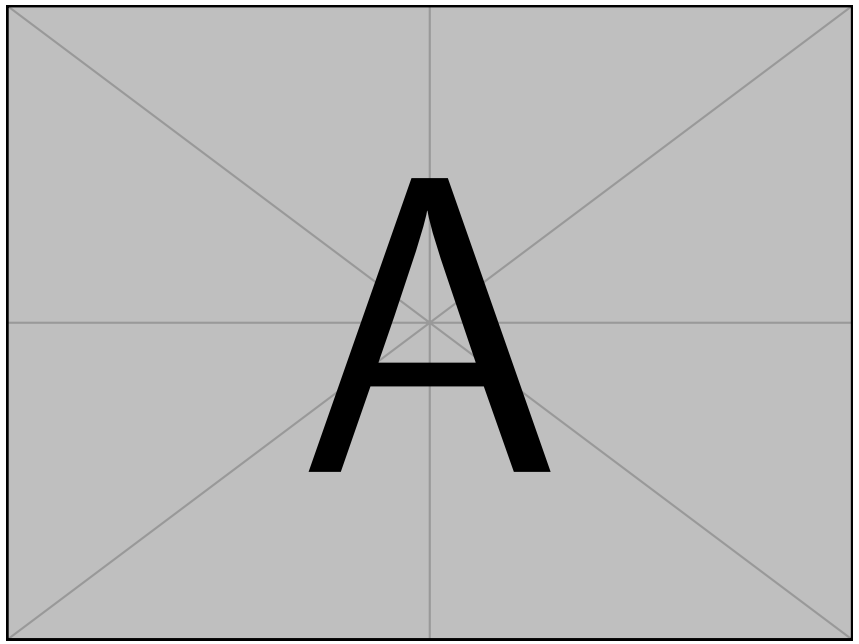

Figura: Validação visual do simulador: a perturbação inicial se espalha e dissipa ao longo do tempo, conforme esperado pelo modelo físico.

Análise de Escalabilidade (Speedup)

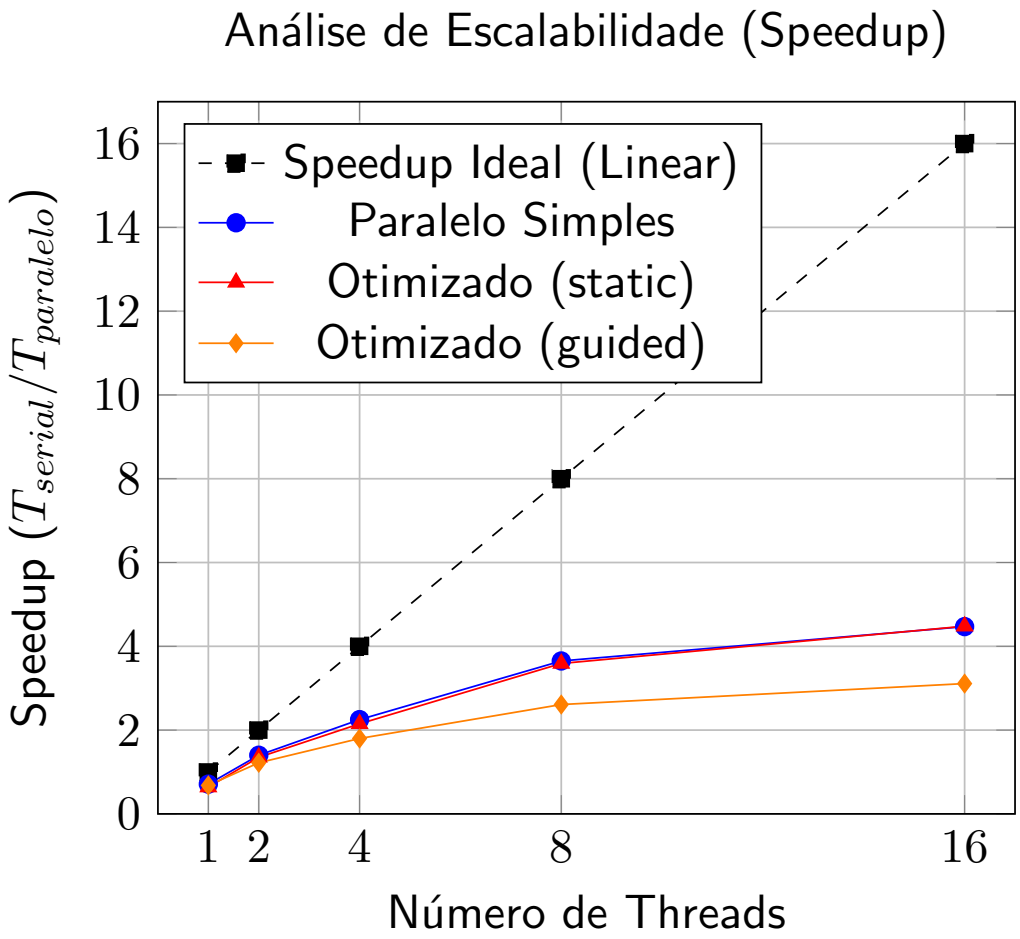

Figura: Gráfico de Speedup vs. Número de Threads. Nenhuma versão atinge a escalabilidade ideal, e as versões *Simples* e *static* têm desempenho quase idêntico.

O Impacto Crítico do Agendamento (schedule)

A Escolha Certa vs. a Escolha Errada

A política de agendamento deve ser compatível com a natureza da carga de trabalho.

`schedule(static)`

- **Resultado:** Melhor desempenho (Speedup de 4.48x).
- **Motivo:** Carga de trabalho perfeitamente uniforme. A divisão do trabalho é feita uma única vez, com overhead mínimo.

`schedule(dynamic)`

- **Resultado:** Desaceleração catastrófica.
- Com 16 threads, foi **110x mais lento** que o código serial.
- **Motivo:** Overhead massivo para agendar cada iteração individualmente, superando o trabalho útil.

Desaceleração Paralela com `schedule(dynamic)`

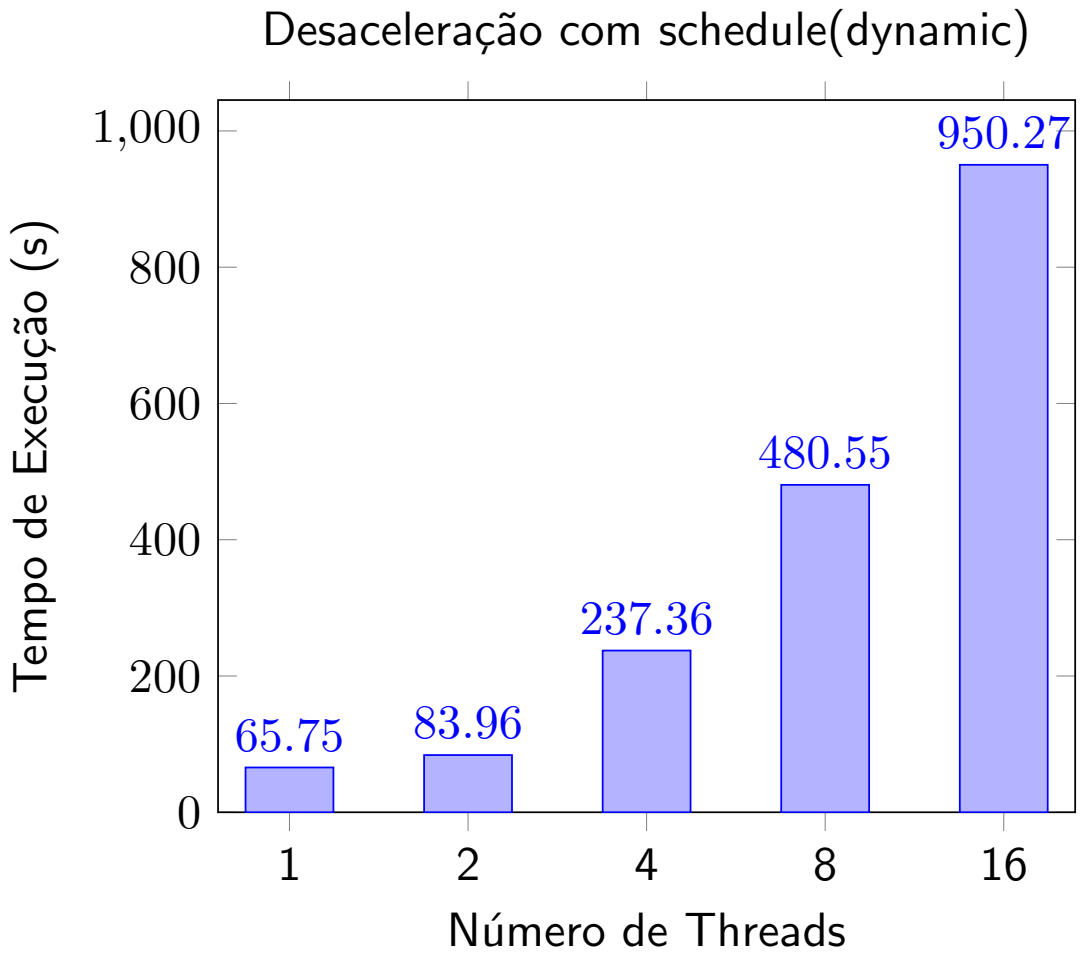

Figura: Aumento drástico do tempo de execução com mais threads, um exemplo clássico de *parallel slowdown* devido ao alto custo do agendamento dinâmico para esta carga de trabalho.

Roteiro: Escolhendo a Estratégia Correta

Passo 1: Identificar Laços Aninhados

Se o gargalo estiver em laços perfeitamente aninhados, a cláusula `collapse` é uma forte candidata a ser usada.

Passo 2: Analisar a Carga de Trabalho

A escolha da política de `schedule` é fundamental e depende do custo de cada iteração.

Cenário 1: Carga Uniforme

- Cada iteração leva o mesmo tempo.
- **Solução:** `schedule(static)`.
- **Justificativa:** Overhead de agendamento mínimo.

Cenário 2: Carga Não-Uniforme

- O tempo por iteração varia.
- **Solução:** `schedule(dynamic)` ou `(guided)`.
- **Justificativa:** Permite balanceamento dinâmico da carga, evitando threads ociosas.

Conclusões

- **Estratégia de Micro-otimização:** A decisão correta de quais laços paralelizar gerou o maior ganho de desempenho, não uma cláusula específica.
- **O `schedule` deve ser adequado à carga:** A comparação entre `static` e `dynamic` mostrou a diferença entre o melhor desempenho e uma desaceleração catastrófica, provando ser o fator mais crítico.
- **O impacto do `collapse` pode ser sutil:** Nesta simulação, com carga uniforme, o `collapse` não trouxe ganho significativo sobre uma paralelização simples, sugerindo que o gargalo era a largura de banda da memória, não o balanceamento de carga.
- **Abstrações exigem compreensão:** As ferramentas do OpenMP são poderosas, mas seu uso inadequado (como `schedule(dynamic)` neste caso) pode ser desastroso.