Relatório: Avaliaćão de escalabilidade

Aluno: Cristovão Lacerda Cronje

1. Introdução

Este relatório descreve a implementação e a análise de desempenho de uma simulação numérica baseada na equação simplificada de Navier-Stokes. A simulação foi desenvolvida em linguagem C, com paralelização utilizando a biblioteca OpenMP, de modo a acelerar os cálculos por meio da execução concorrente em múltiplas threads.

O foco principal deste estudo é a avaliação da escalabilidade do código através de diferentes tamanhos de malha tridimensional (100×50×50, 100×100×50 e 100×100×100) e diferentes esquemas de escalonamento do OpenMP (static, dynamic e guided). O código foi executado no NPAD, utilizando 1 nó da partição amd-512, com --cpus-per-task=128 e tempo de execução limitado a 20 minutos.

Além da análise de desempenho, buscou-se identificar gargalos de escalabilidade, bem como investigar os conceitos de **escalabilidade forte** e **escalabilidade fraca** no contexto das versões otimizadas do código.

2. Objetivos

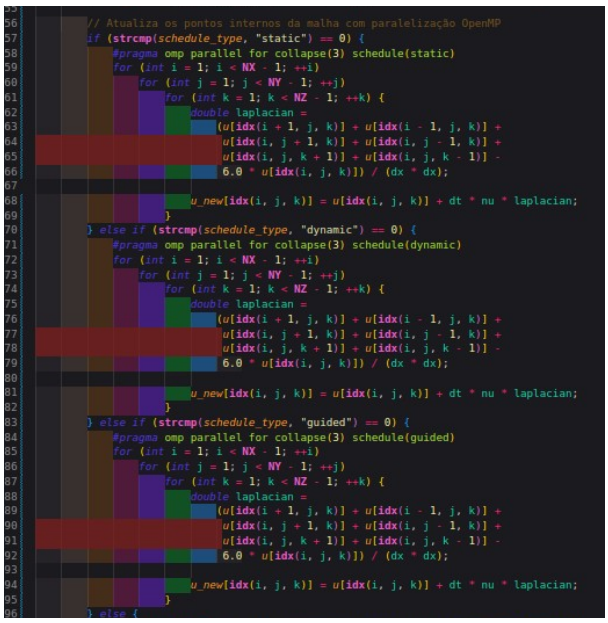
- Implementar uma simulação paralela da difusão viscosa da velocidade de um fluido utilizando OpenMP.
- Coletar o valor do ponto central da malha 3D ao longo das iterações temporais.
- Comparar o desempenho entre diferentes estratégias de agendamento (scheduling) do OpenMP e a versão serial.
- Analisar a escalabilidade e eficiência do código com diferentes tamanhos de malha e número de threads.
- Gerar gráficos que representem a evolução dos valores e o tempo de execução.

3. Metodologia

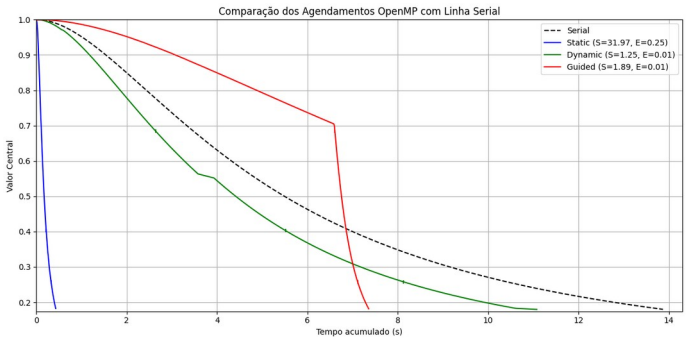
A simulação resolve numericamente a equação de difusão em uma malha tridimensional. O domínio foi discretizado e atualizado iterativamente utilizando um esquema explícito.

A paralelização foi feita aplicando a diretiva #pragma omp parallel for collapse(3) nos três laços aninhados que percorrem os eixos x, y e z da malha. Três estratégias de agendamento (static, dynamic, guided) foram testadas e comparadas com uma versão serial, sem OpenMP.

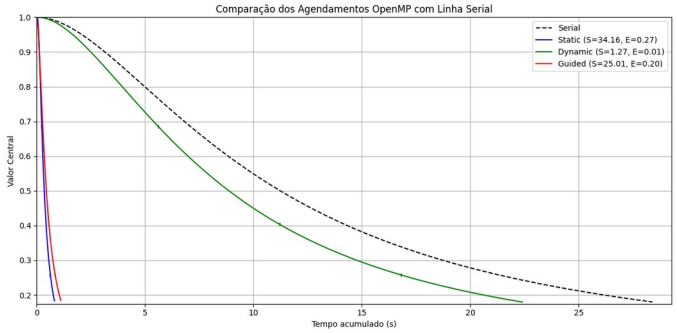
As simulações executadas foram as seguintes, com as configurações de malha:



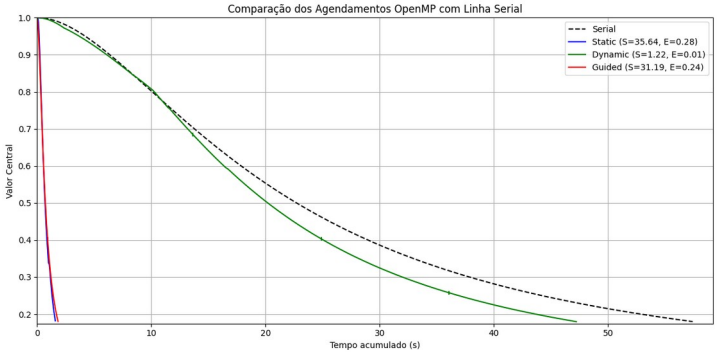
100×50×50(128 threads)



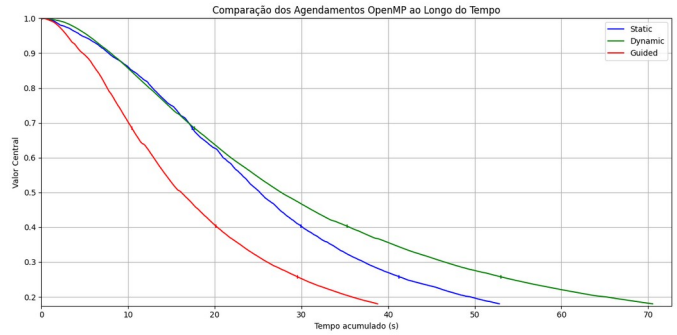
100×100×50(128 threads)



100×100×100(128 threads)



100×100×100 (4 threads, 2 nucleos)*



*Também foi adicionada uma comparação com a execução da atividade 11, demonstrando a redução drástica do tempo com o aumento do número de threads **(escalabilidade forte).**

4. Paralelismo com OpenMP

As diferenças de desempenho observadas entre as estratégias se relacionam diretamente à forma como o OpenMP distribui as iterações entre as threads:

Estratégia	Descrição	Quando Usar	Vantagens / Desvantagens
static	Divide as iterações em blocos fixos entre threads	Iterações de custo uniforme	Baixo overhead, alta eficiência quando a carga é balanceada
dynamic	Threads pegam blocos conforme terminam	Iterações com tempo variável	Melhor balanceamento de carga em casos desbalanceados, porém com alto overhead
guided	Blocos decrescentes dinamicamente(balancear e reduzir overhead)	Iterações que tendem a ficar mais leves com o tempo	Equilíbrio entre balanceamento e overhead, mais eficiente que dynamic em muitos casos

Análise do Desempenho

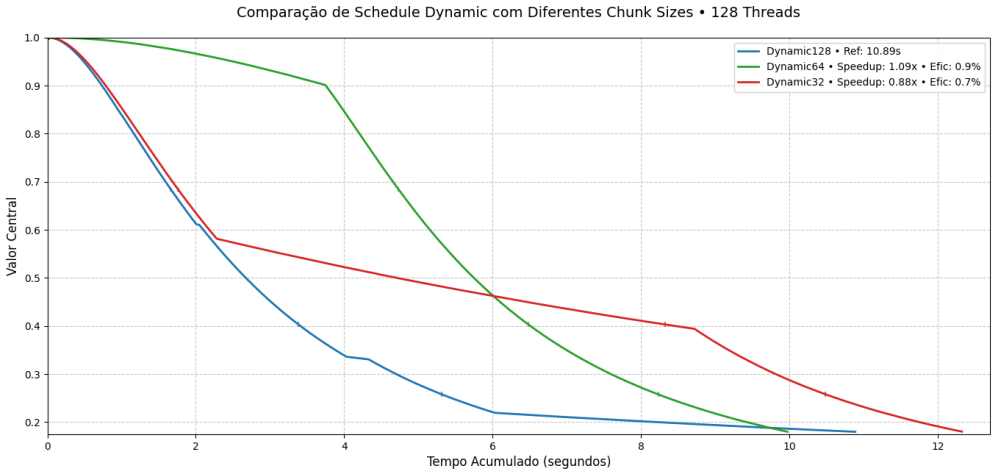
Durante os testes com 128 threads no NPAD (partition=amd-512), os resultados mostraram que a estratégia static apresentou o **melhor desempenho geral**, seguida por guided. **A estratégia dynamic, foi a pior** entre as três estratégias paralelas, inclusive apresentando tempos de execução próximos ao da versão serial (piores tempos absolutos).

Motivo do baixo desempenho de dynamic:
O schedule(dynamic) introduz um **overhead considerável** relacionado ao gerenciamento dinâmico de tarefas: as threads precisam continuamente solicitar novos blocos de trabalho à medida que terminam os anteriores. Esse processo é custoso, especialmente em ambientes com:

- **Grande número de threads (128 no caso)**, onde há maior disputa pelas estruturas internas de controle de tarefas.
- **Carga de trabalho homogênea**, como no caso da simulação da equação de difusão, em que cada iteração realiza praticamente o mesmo número de operações e consome o mesmo tempo. Neste cenário, o balanceamento dinâmico é desnecessário.
- Uso de collapse(3), que aumenta significativamente a quantidade de iterações paralelizáveis (~1 milhão de iterações para uma malha 100x100x100), agravando o custo de agendamento.

Como o desempenho do dynamic poderia ser melhorado:
O desempenho da estratégia dynamic pode ser otimizado com a definição de um **tamanho de bloco apropriado**, reduzindo o número de interações de agendamento entre threads. Por exemplo: #pragma omp parallel for collapse(3) schedule(dynamic, 128).

A definição explícita do tamanho dos blocos (chunk size) reduz o número de requisições dinâmicas e melhora a eficiência do paralelismo. Exemplo executado no NPAD: (malha 150x150x150, chuncksize 32, 64 e 128):



5. **Resultados e Análise**

5.1 Speedup e Eficiência

As métricas utilizadas foram:

- **Speedup** = Tempo serial / Tempo paralelo
- **Eficiência** = Speedup / Número de threads

Os resultados apontam que:

- A eficiência da versão ***static*** decresce à medida que o número de threads aumenta em malhas maiores. Foi a que obteve o melhor compromisso em termos de escalabilidade forte(para a malha 100x100x100)
- A versão ***dynamic*** tende a apresentar uma eficiência razoável em situações de escalabilidade fraca (quando o tamanho da malha é aumentado mantendo-se o número de threads fixo), pois permite um melhor balanceamento de carga entre as threads. No entanto, seu desempenho d**epende da definição adequada do *chunk size*,** que deve ser ajustado conforme o tamanho e a complexidade do problema. Um *chunk size* mal dimensionado pode comprometer a eficiência da execução paralela. Essa comparação pode ser visualizada no gráfico, considerando um problema de tamanho específico.
- A versão ***guided*** obteve o segundo melhor desempenho em termos de escalabilidade forte (mantendo o problema fixo e aumentando as threads).

5.2 Análise

Estratégia	Desempenho Observado	Escalabilidade	Gargalos	Comentários
Serial	Base	-	-	Base para calcular speedup
Static	Rápido para malhas pequenas	Escalabilidade forte limitada	Divisão rígida	Pouco adaptável a variações de carga
Dynamic	Indicado para malhas grandes, especialmente quando há desequilíbrio na carga de trabalho entre as iterações.	Escalabilidade fraca razoável/moderada, devido à capacidade de redistribuir dinamicamente as tarefas.	Possui overhead mais elevado, pois o agendador precisa decidir em tempo de execução qual thread executará qual bloco de iterações.	Altamente adaptável a desequilíbrios, sua eficiência depende de uma escolha adequada do `chunk size`.
Guided	Balanceado	Bom compromisso entre forte e fraca	Menor overhead que dynamic	Estratégia recomendada

6. Conclusão

A simulação evidenciou ganhos expressivos de desempenho com o uso de múltiplas threads. A estratégia ***static*** apresentou o melhor desempenho geral, beneficiando-se de baixo overhead em problemas bem balanceados. A diretiva ***guided*** também se destacou, conciliando bom balanceamento de carga com sobrecarga reduzida. Embora a versão *dynamic* sem definição de *chunk size* tenha tido desempenho apenas ligeiramente superior à versão sequencial, mostrou-se vantajosa em malhas grandes ou cenários com desequilíbrio de carga — especialmente quando o *chunk size* é corretamente ajustado conforme o problema. As análises de escalabilidade revelaram que a escalabilidade forte foi limitada pela sobrecarga e comunicação entre threads em malhas pequenas, enquanto a escalabilidade fraca foi mais bem aproveitada nas versões *dynamic* e *guided*, com ganhos proporcionais ao aumento da carga de trabalho.