UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
IDCA3703 - PROGRAMAÇÃO PARALELA

TAREFA 12 - AVALIAÇÃO DA ESCALABILIDADE
RELATÓRIO DE EXECUÇÃO

ERNANE FERREIRA ROCHA JUNIOR

NATAL/RN, 11 DE MAIO DE 2025

SUMÁRIO

1. INTRODUÇÃO

A simulação de fenômenos físicos complexos, como a difusão de calor ou a solução das equações de Navier-Stokes, frequentemente exige um grande poder computacional devido à natureza tridimensional e iterativa dos cálculos envolvidos. Para tornar tais simulações viáveis em tempo hábil, é essencial explorar técnicas de paralelização, como aquelas providas pela biblioteca OpenMP em arquiteturas multicore.

Este relatório apresenta uma análise de desempenho e escalabilidade de uma implementação paralela em OpenMP para a simulação de difusão 3D de calor, adaptada como um modelo simplificado das equações de Navier-Stokes. A análise foi conduzida utilizando um nó de processamento do supercomputador do Núcleo de Processamento de Alto Desempenho (NPAD) da Universidade Federal do Rio Grande do Norte (UFRN). O objetivo central é avaliar a escalabilidade da aplicação sob dois cenários distintos:

- **Escalabilidade forte**: mantendo o volume de dados fixo e aumentando o número de threads, observa-se o ganho de desempenho obtido;
- **Escalabilidade fraca**: aumentando o volume de dados proporcionalmente ao número de threads, verifica-se se o tempo de execução permanece aproximadamente constante.

Essas duas abordagens permitem identificar gargalos de paralelização, eficiência do agendador de tarefas e limitações estruturais do código. A partir dos resultados, são discutidos os pontos de saturação da escalabilidade e possíveis caminhos para otimização futura.

2. METODOLOGIA

Para avaliar a escalabilidade da implementação paralela do simulador de difusão 3D de calor, foram realizados dois experimentos distintos: um de escalabilidade forte e outro de escalabilidade fraca. Ambos os testes foram executados no ambiente de computação de alto desempenho do NPAD (Núcleo de Processamento de Alto Desempenho da UFRN), utilizando a partição intel-512, com suporte a múltiplas threads via OpenMP.

2.1. Código

A base do código implementa uma simulação iterativa tridimensional do processo de difusão com os seguintes parâmetros fixos:

1. Número de passos de tempo: *NSTEPS = 1000*
2. Constantes físicas: *DT = 0.01*, *DX = 1.0*, *VISC = 0.1*
3. Paralelização: Diretivas *#pragma omp parallel for* com *collapse(3)* e agendamento *schedule(guided, 1024)*, após benchmarking prévio

2.2. Escalabilidade Forte

O experimento de escalabilidade forte mantém o tamanho do problema fixo (*N = 256*) e avalia o tempo de execução com diferentes números de threads. O código é compilado em uma versão com *#define N 256*, e executado com:

```
OMP_NUM_THREADS=1 2 4 8 16 24
```

O script SLURM usado para execução é reproduzido abaixo:

```
#!/bin/bash
#SBATCH --job-name=ERNANE-OPENMP-TASK12
#SBATCH --time=0-0:20
#SBATCH --partition=intel-512
#SBATCH --output=slurm-strong-scalability-%j.out

for threads in 1 2 4 8 16 24
do
  export OMP_NUM_THREADS=$threads
  ./strong-scalability-assessment
done
```

2.3. Escalabilidade Fraca

O experimento de escalabilidade fraca busca manter constante a carga de trabalho por *thread*, aumentando o tamanho do domínio de simulação (*N*) proporcionalmente ao número de *threads*. Para

isso, utilizou-se o volume de referência *N=32* para 1 *thread*, e os valores seguintes foram estimados de modo que $\frac{N^3}{Threads} \approx CTE$.

O código desta versão aceita *N* como argumento de linha de comando:

```
./weak-scalability-assessment <N>
```

O script SLURM correspondente é o seguinte:

```
#!/bin/bash
#SBATCH --job-name=ERNANE-OPENMP-TASK12-WEAK
#SBATCH --time=0-1:00
#SBATCH --partition=intel-512
#SBATCH --output=slurm-weak-scalability-%j.out

for threads in 1 2 4 8 16 24
do
  export OMP_NUM_THREADS=$threads
  case $threads in
    1) N=32 ;; 2) N=40 ;; 4) N=51 ;;
    8) N=64 ;; 16) N=80 ;; 24) N=91 ;;
  esac
  ./weak-scalability-assessment $N
done
```

2.4. Coleta de Dados

Ao final de cada execução, o tempo total da simulação é registrado e impresso no terminal. Os dados coletados foram organizados em arquivos CSV.

2.4.1. Tabela 1 – Escalabilidade Forte (N fixo = 256)

N	NSTEPS	Threads	Time (s)
256	1000	1	206.182
256	1000	2	181.725
256	1000	4	183.169
256	1000	8	183.608
256	1000	16	184.781
256	1000	24	185.823

2.4.2. Tabela 2 – Escalabilidade Fraca (N proporcional)

N	NSTEPS	Threads	Time (s)
32	1000	1	0.610
40	1000	2	1.197
51	1000	4	2.641
64	1000	8	4.764
80	1000	16	8.399
91	1000	24	11.920

3. RESULTADOS

A análise da escalabilidade do código de simulação foi dividida em duas categorias: escalabilidade forte, com tamanho de problema fixo, e escalabilidade fraca, com o tamanho do problema aumentando proporcionalmente ao número de threads. A seguir, discutimos os resultados de ambos os testes com base nos tempos de execução coletados.

3.1. Escalabilidade Forte

Na escalabilidade forte, espera-se que o tempo de execução diminua à medida que mais threads são utilizadas para resolver o mesmo problema. Entretanto, os dados obtidos (Section 2.4.1. - Tabela 1) revelam um ganho limitado de desempenho.

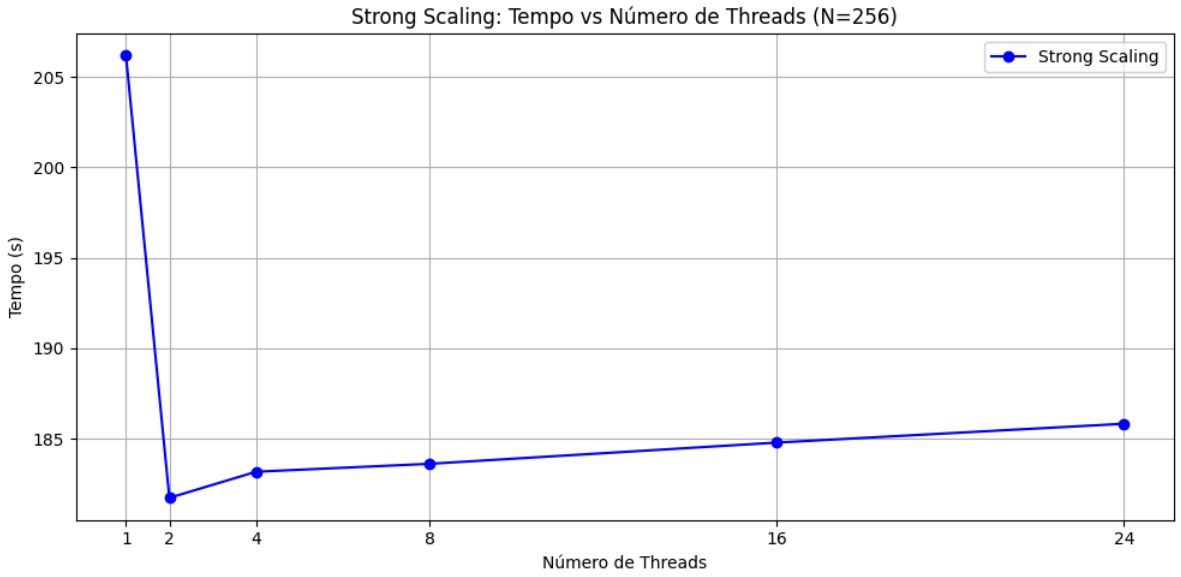

Figure 1: Escalabilidade Forte: Tempo de Execução vs Número de Threads

O tempo total reduz de forma significativa apenas de 1 para 2 *threads*. A partir de 4 *threads*, o desempenho estagna e começa a piorar levemente com o aumento das *threads*. Isso sugere a presença de gargalos de paralelismo, como: saturação de memória/cache (acessos irregulares ao *grid 3D*), sobrecarga de sincronização ou gerenciamento de *threads*, custo elevado de *memcpy* a cada passo e a estrutura dos dados e a granularidade de paralelismo podem estar limitando os ganhos com mais *threads*.

3.2. Escalabilidade Fraca

Na escalabilidade fraca, espera-se que o tempo de execução permaneça aproximadamente constante à medida que o problema cresce proporcionalmente ao número de threads. No entanto, os tempos medidos aumentam linearmente com as threads:

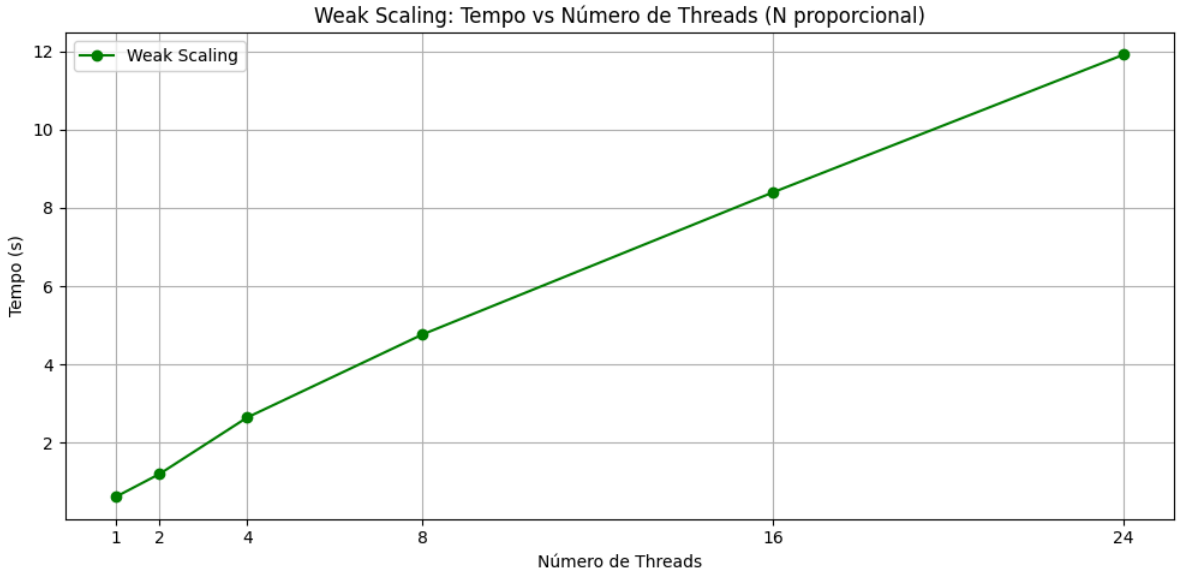

Figure 2: Tempo de Execução vs Threads (N proporcional ao volume por thread)

O tempo cresce de maneira quase linear com o número de *threads*, mesmo com o volume por *thread* mantido constante. Isso indica que o sistema não está escalando eficientemente em função do aumento de carga e comunicação. As possíveis razões podem incluir o aumento da sobrecarga de cache/memória em *grids* maiores, a diminuição do paralelismo efetivo em regiões com pouca variação do valor inicial e o custo de alocação dinâmica em grandes volumes de dados 3D.

4. CONCLUSÃO

A análise de escalabilidade do código de simulação de difusão de calor 3D, utilizando as abordagens de escalabilidade forte e escalabilidade fraca, revelou informações valiosas sobre o desempenho do código em diferentes configurações de *threads* e tamanhos de problema.

A escalabilidade forte apresentou uma tendência de redução do tempo de execução apenas até 2 *threads*, com o desempenho estagnando ou até piorando à medida que mais *threads* eram adicionadas. Isso indica a presença de gargalos de paralelismo, como a limitação de recursos de memória/cache, além de um possível aumento no custo de gerenciamento de *threads* e sincronização, o que impede uma melhoria significativa no desempenho com o aumento de *threads* além de 2. Isso sugere que o problema é limitado pela natureza do algoritmo e pela estrutura dos dados, não aproveitando completamente o paralelismo disponível.

Na análise de escalabilidade fraca, o aumento proporcional do tamanho do problema ao número de *threads* gerou tempos de execução que cresceram de forma quase linear, indicando uma escalabilidade não ideal. Esse comportamento pode ser atribuído a vários fatores, como a sobrecarga de memória e cache em grids maiores e a diminuição do paralelismo efetivo à medida que o tamanho do problema aumenta. Apesar de o volume por *thread* se manter constante, a sobrecarga associada ao aumento do tamanho do problema em relação ao número de *threads* não resultou em um aumento proporcional de desempenho.

Em termos gerais, embora o código apresente algum nível de paralelismo, ele não está tirando total proveito dos recursos disponíveis. O desenvolvimento de soluções mais eficazes para a distribuição de trabalho e acesso à memória será crucial para melhorar tanto a escalabilidade forte quanto a fraca.

5. ANEXOS

- Repositório no Github com o programa desenvolvido: https://github.com/ErnaneJ/parallel-programming-dca3703