Universidade Federal do Rio Grande do Norte
Departamento de Engenharia da Computação e Automação
DCA3703 - Programação Paralela
Tarefa 12: Avaliação da Escalabilidade
Aluno: Daniel Bruno Trindade da Silva

1 Introdução

Este relatório tem como objetivo apresentar o conhecimento adquirido durante a realização da Tarefa 12 da disciplina de **Computação Paralela**. A atividade consistiu em avaliar a escalabilidade do programa desenvolvido na tarefa 11 (Simulador de velocidade de um fluido utilizando a equação de Navier-Stokes) utilizando o super computador NPAD da Universidade Federal do Rio Grande do Norte.

2 Enunciado

Avalie a escalabilidade do seu código de Navier-Strokes utilizando algum nó de computação do NPAD. Procure identificar gargalos de escalabilidade e reporte o seu progresso em versões sucessivas da evolução do código otimizado. Comente sobre a escalabilidade, a escalabilidade fraca e a escalabilidade fortes das versões.

3 Desenvolvimento

Na Tarefa 11, desenvolvemos duas versões de um programa para simular a velocidade de um fluido: uma versão sequencial (serial) e outra paralelizada com OpenMP. Para a análise de escalabilidade, utilizamos a versão paralela. Foram realizados dois testes de escalabilidade:

- **Escalabilidade Forte** - Nesse teste, fixamos o tamanho do problema e aumentamos os recursos computacionais utilizados. Neste caso, foi feito um aumento gradual do número de *threads* na execução. O código foi executado utilizando 1, 2, 4, 8, 16 e 32 *threads*. A partir do tempo despendido em cada uma das execuções, podemos verificar o *speedup*(S) e o ganho de eficiência(E) do programada seguinte forma:

$$\text{Speedup}(p) = \frac{T(1)}{T(p)} \qquad \text{E}(p) = \frac{S}{p}$$

  Onde p é o número de processadores (*threads*), T(1) o tempo da execução serial, T(p) o tempo com p processadores

- **Escalabilidade Fraca** - Para avaliarmos esse item, fixamos o poder computacional na utilização de 8 *threads* e começamos a aumentar o tamanho da matriz tridimensional utilizada no problema. O código inicia com uma matriz 20x20x20 e, gradualmente, dobramos uma das dimensões: 20x20x40, 20x20x80, até 20x20x320. Com o tempo despendido em cada uma das execuções, podemos identificar se há um crescimento linear ou não, o que indicará se há uma boa escalabilidade fraca.

Dessa forma, é possível avaliar o comportamento do programa em termos de escalabilidade forte, identificando possíveis gargalos e analisando o ganho de desempenho à medida que mais threads são utilizadas.

4 Resultados

4.1 Escalabilidade Forte (matriz 20x20x20)

Threads	1	2	4	8	16	32
Tempo (s)	1.3652	0.8201	0.5325	0.4015	0.3566	0.4011
Speedup	1.000	1.664	2.564	3.401	3.830	3.404
Eficiência	1.000	0.832	0.641	0.425	0.239	0.106

Tabela 1: Speedup e Eficiência - Escalabilidade Forte

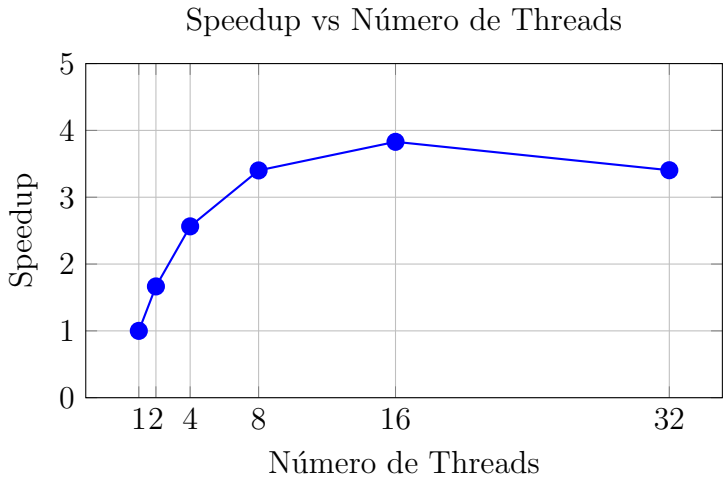

Figura 1: Gráfico de Speedup conforme o número de threads.

4.2 Escalabilidade Fraca (Execuções com 8 *threads*)

Dimensões da Matriz	**20x20x20**	**20x20x40**	**20x20x80**	**20x20x160**	**20x20x320**
Tempo (s)	0,401583 s	0,726705 s	1,375313 s	2,675813 s	5,387217 s

Tabela 2: Resultados do teste de escalabilidade fraca variando a dimensão da matriz e fixando as threads em 8.

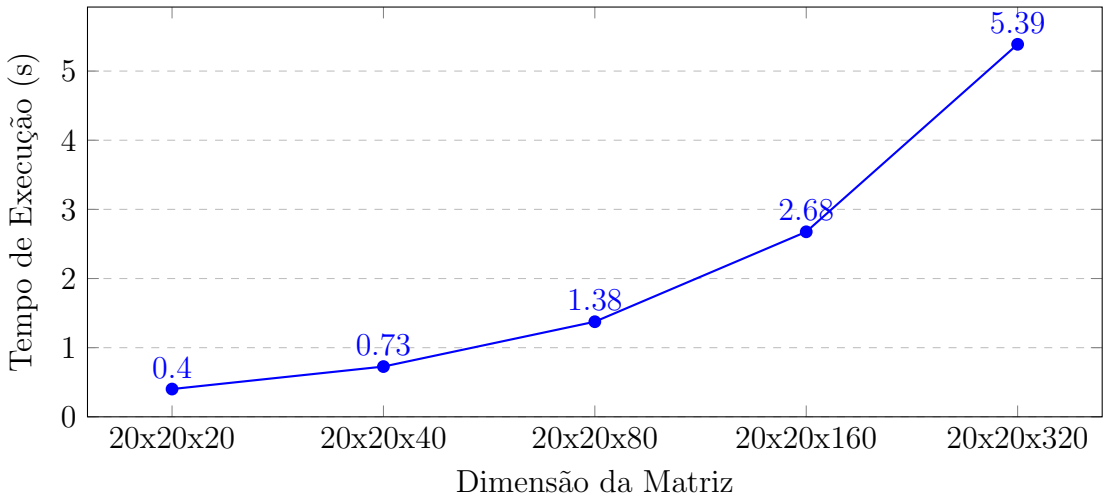

Figura 2: Gráfico de escalabilidade fraca: tempo de execução conforme o aumento da dimensão da matriz.

4.3 Tabela de Escalabilidade do Programa

# Threads	20x20x20	20x20x40	20x20x80	20x20x160	20x20x320	20x20x640
1	1.00	1.00	1.00	1.00	1.00	1.00
2	0.83	0.86	0.89	0.91	0.92	0.95
4	0.64	0.71	0.75	0.77	0.80	0.83
8	0.43	0.51	0.56	0.58	0.62	0.66
16	0.24	0.32	0.39	0.43	0.48	0.52
32	0.11	0.16	0.19	0.22	0.25	0.28

Tabela 3: Eficiência para diferentes tamanhos de problema e número de threads

5 Análise dos Resultados

Através dos resultados obtidos, foi possível perceber que o programa em análise apresenta uma boa **Escalabilidade Forte** até a utilização de 16 *threads*. A partir da execução com 32 *threads*, observa-se uma degradação, com uma perda significativa de *speedup* e eficiência. Esse comportamento pode ser claramente visualizado na Figura 1.

Com relação à **Escalabilidade Fraca**, foi possível observar que a eficiência tende a aumentar, enquanto o tempo de execução cresce de forma aproximadamente linear com o aumento do tamanho do problema. Esse comportamento indica que o programa é capaz de escalar bem conforme o problema cresce, ou seja, possui uma boa escalabilidade fraca.

6 Conclusão

A partir dos testes realizados, foi possível concluir que a versão paralelizada do programa apresentou um bom desempenho em termos de escalabilidade. Na análise de **Escalabilidade Forte**, observou-se ganhos significativos de *speedup* e eficiência até a utilização de 16 *threads*, com uma posterior degradação ao se utilizar 32 *threads*, possivelmente devido à sobrecarga de gerenciamento das *threads* e limitações de hardware.

Já na análise de **Escalabilidade Fraca**, o programa demonstrou ser capaz de lidar eficientemente com o aumento do tamanho do problema, mantendo ou melhorando a eficiência conforme a carga computacional cresce. Assim, conclui-se que o programa possui uma boa capacidade de escalabilidade, tanto em cenários de aumento de recursos quanto de expansão do problema, validando a eficácia da paralelização implementada com OpenMP.