UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
IDCA3703 - PROGRAMAÇÃO PARALELA

TAREFA 13 - AFINIDADE DE THREADS
RELATÓRIO DE EXECUÇÃO

ERNANE FERREIRA ROCHA JUNIOR

NATAL/RN, 13 DE MAIO DE 2025

SUMÁRIO

1. INTRODUÇÃO

A simulação de escoamentos utilizando as equações de Navier-Stokes é uma tarefa computacionalmente intensiva, especialmente quando realizada em três dimensões e com alta resolução espacial. Para viabilizar execuções eficientes, é comum empregar técnicas de paralelização, sendo o *OpenMP* uma das abordagens mais utilizadas em arquiteturas de memória compartilhada. No entanto, o desempenho de programas paralelos depende não apenas do número de threads utilizadas, mas também da forma como essas threads são distribuídas entre os núcleos do processador — um aspecto controlado por meio da afinidade de threads.

Neste trabalho, avaliamos o impacto das diferentes configurações de afinidade de *threads* suportadas pelo *OpenMP* e pelo sistema operacional sobre a escalabilidade de uma simulação simplificada das equações de Navier-Stokes. Os experimentos foram realizados no mesmo nó de computação utilizado na Tarefa 12 do ambiente *NPAD*, aproveitando uma máquina com 24 núcleos disponíveis. Foram testadas as opções *false*, *true*, *close*, *spread* e *master* para a variável *OMP_PROC_BIND*, variando também o número de *threads* de 1 até 24. O objetivo é identificar como essas configurações influenciam o tempo de execução, o *speedup* e a eficiência da aplicação paralela, fornecendo subsídios para melhores decisões de configuração em ambientes de alto desempenho.

2. METODOLOGIA

Para avaliar o impacto das diferentes configurações de afinidade de threads, foi implementada uma simulação tridimensional simplificada baseada nas equações de Navier-Stokes. O código, escrito em *C* com suporte a *OpenMP*, resolve uma equação de difusão utilizando um esquema explícito e opera sobre um domínio cúbico de dimensão $N = 256$ por 1000 passos de tempo. A matriz de velocidades é inicializada com um pulso no centro do domínio, e a evolução temporal é paralelizada nos três eixos espaciais utilizando a diretiva *collapse*(3).

O código foi compilado com otimizações e suporte a *OpenMP* via o compilador *gcc* com as *flags -O3* e *-fopenmp*. A execução foi feita em um ambiente *SLURM*, utilizando um único nó com *24 CPUs* da partição *intel-512*, com um *script* que automatiza a variação dos parâmetros de execução.

As seguintes configurações de afinidade de threads (OMP_PROC_BIND) foram testadas: *false*, *true*, *close*, *spread* e *master*. Para cada configuração, a simulação foi executada com 1, 2, 4, 8, 12, 16, 20 e 24 *threads*. A variável *OMP_PLACES* foi fixada como *cores*, garantindo que as *threads* fossem distribuídas apenas entre os núcleos físicos. O tempo total de execução da simulação foi registrado ao final de cada execução por meio da função *omp_get_wtime*().

Os resultados coletados incluem o tempo de execução, a aceleração (*speedup*) em relação ao tempo de execução com uma única *thread*, e a eficiência paralela, definida como o *speedup* dividido pelo número de *threads*. Esses dados foram organizados em uma tabela e visualizados por meio de gráficos gerados com a biblioteca *matplotlib* em *Python*, de modo a facilitar a interpretação comparativa entre as diferentes afinidades de *threads*.

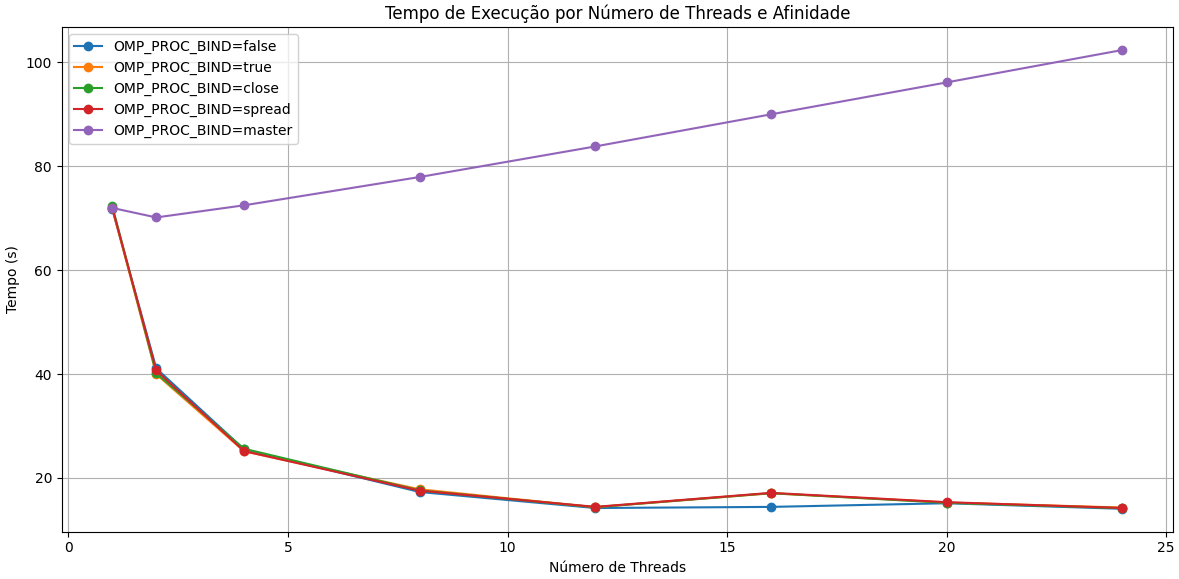
3. RESULTADOS

Os experimentos revelaram diferenças significativas no desempenho da simulação de acordo com a configuração de afinidade de *threads* utilizada. A tabela abaixo apresenta os tempos de execução medidos para cada combinação de número de *threads* e valor da variável *OMP_PROC_BIND*.

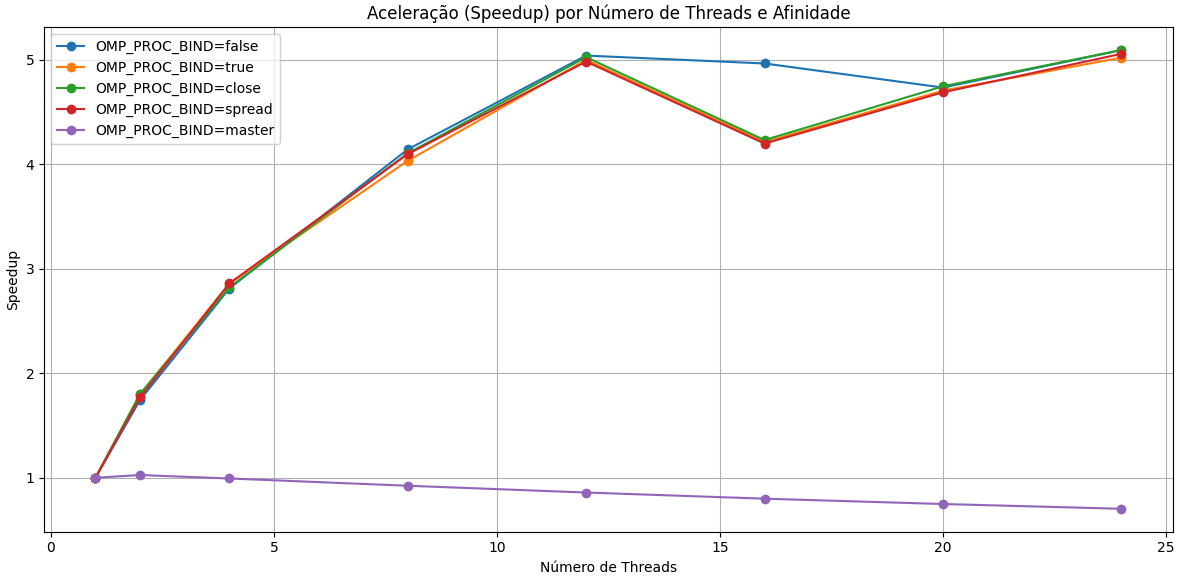
OMP_PROC_BIND	Threads	Tempo (s)
false	1	71.676
false	2	41.090
false	4	25.499
false	8	17.304
false	12	14.222
false	16	14.439
false	20	15.142
false	24	14.075
true	1	71.868
true	2	39.974
true	4	25.152
true	8	17.819
true	12	14.379
true	16	17.061
true	20	15.277
true	24	14.316
close	1	72.279
close	2	40.129
close	4	25.620
close	8	17.611
close	12	14.385
close	16	17.082
close	20	15.231
close	24	14.192
spread	1	71.945
spread	2	40.682
spread	4	25.166
spread	8	17.561
spread	12	14.440

spread	16	17.138
spread	20	15.346
spread	24	14.228
master	1	71.953
master	2	70.127
master	4	72.462
master	8	77.905
master	12	83.789
master	16	89.964
master	20	96.106
master	24	102.338

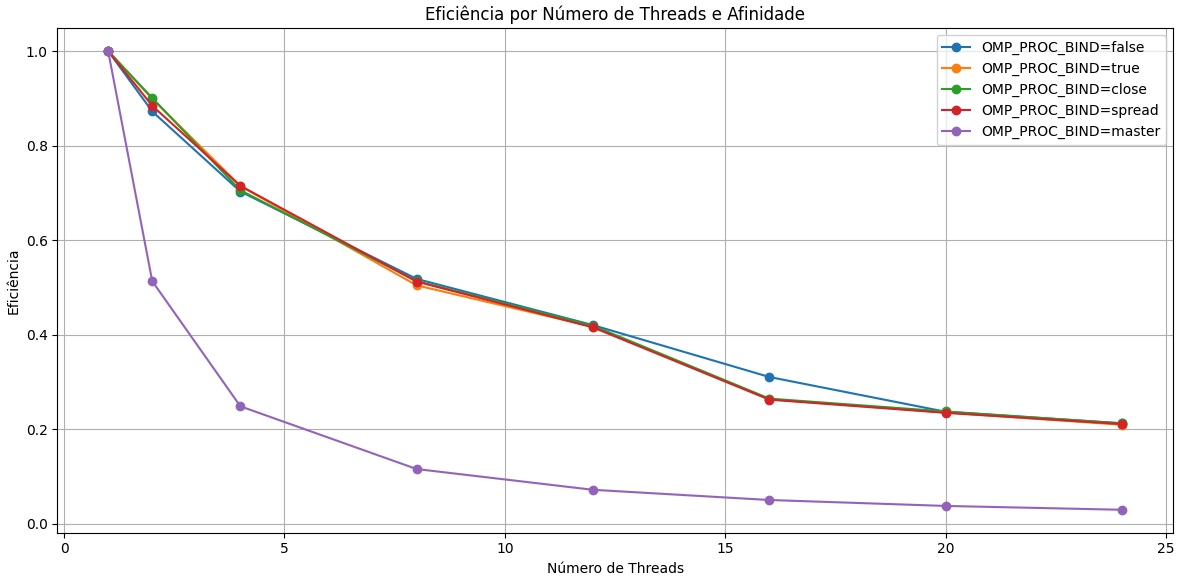
A imagem abaixo mostra o comportamento do tempo de execução conforme o número de *threads* aumenta, para cada uma das afinidades testadas.



A partir desses tempos, foi possível calcular a aceleração (*speedup*) em relação à execução com uma única *thread*. Como esperado, as afinidades *false*, *true*, *close* e *spread* apresentaram ganhos significativos de desempenho à medida que o número de *threads* aumentava, alcançando até *5x* de aceleração com 24 *threads*. Por outro lado, a configuração *master*, que força todas as threads a serem executadas no mesmo core, apresentou resultados consistentemente ruins, com aceleração inferior a 1.



A eficiência paralela, mostrada na imagem acima, reforça essas observações. Para as quatro primeiras afinidades, a eficiência cai progressivamente com o aumento no número de *threads* — como é esperado devido à sobrecarga de comunicação e à limitação de largura de banda da memória. No entanto, mesmo com 24 *threads*, elas mantêm uma eficiência entre 20% e 22%, o que pode ser considerado satisfatório para este tipo de aplicação. A afinidade *master*, novamente, destaca-se negativamente, com eficiências abaixo de 10% já com 4 *threads* e inferiores a 5% a partir de 16 *threads*.



Esses resultados mostram que a escolha da afinidade de *threads* tem impacto direto no aproveitamento dos recursos computacionais e que, quando bem configurada, pode melhorar

significativamente a escalabilidade de aplicações paralelas.

4. CONCLUSÃO

Os experimentos realizados demonstram de forma clara que a afinidade de *threads* é um fator determinante para o desempenho de aplicações paralelas em ambientes de memória compartilhada. Ao utilizar diferentes configurações de *OMP_PROC_BIND*, observou-se que as opções *false*, *true*, *close* e *spread* proporcionaram boa escalabilidade até cerca de 12 *threads*, com ganhos marginais a partir desse ponto, o que reflete os limites impostos pela arquitetura de memória e pela própria natureza do problema. Ainda assim, essas configurações mantiveram uma eficiência razoável, mesmo com 24 *threads*, evidenciando uma distribuição eficiente da carga de trabalho entre os núcleos disponíveis.

Por outro lado, a configuração *master*, que restringe todas as *threads* ao mesmo núcleo, apresentou um desempenho drasticamente inferior, com aumento do tempo de execução à medida que mais *threads* eram adicionadas. Isso evidencia o impacto negativo de uma má política de afinidade, que resulta em saturação do cache e concorrência excessiva pelos mesmos recursos físicos do processador.

Conclui-se, portanto, que para aplicações como a simulação das equações de Navier-Stokes — que envolvem grande volume de cálculos e acesso intensivo à memória — a escolha adequada da afinidade de *threads* pode representar a diferença entre uma execução eficiente e um desperdício significativo de recursos computacionais. Afinidades como *close* e *spread*, que buscam balancear o uso dos núcleos e a localidade de memória, mostraram-se as mais adequadas no contexto testado, oferecendo um bom equilíbrio entre desempenho e escalabilidade.

5. ANEXOS

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