Universidade Federal do Rio Grande do Norte
Departamento de Engenharia da Computação e Automação
DCA3703 - Programação Paralela
Tarefa 13: Afinidade de threads
Aluno: Daniel Bruno Trindade da Silva

1 Introdução

Este relatório tem como objetivo apresentar os conhecimentos adquiridos durante a realização da Tarefa 13 da disciplina de **Computação Paralela**. A atividade consistiu em avaliar a escalabilidade do programa desenvolvido na Tarefa 11 — um simulador da velocidade de um fluido utilizando a equação de Navier-Stokes — aplicando diferentes políticas de afinidade de *threads*.

2 Enunciado

Avalie como a escalabilidade do seu código de Navier-Stokes muda ao utilizar os diversos tipos de afinidades de *threads* suportados pelo sistema operacional e pelo OpenMP, no mesmo nó de computação do NPAD utilizado para a Tarefa 12.

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 requerida nesta tarefa, utilizamos a versão paralelizada do código.

Nesta tarefa, analisamos os impactos da cláusula `proc_bind()` com as seguintes políticas de afinidade:

- `spread` — O openMP distribui as *threads* de forma espalhada pelos processadores, maximizando a distância entre elas. O objetivo é utilizar o máximo de recursos do hardware possível, como diferentes *sockets* ou núcleos físicos. Aumenta a latência de comunicação entre as *threads* em troca de menos disputa por recurso. Ideal quando temos programas com *threads* independentes;
- `close` — O openMP agrupa as *threads* próximas umas das outras, preferencialmente no mesmo *socket* ou núcleos adjacentes ao da *thread* master. Utilizada quando queremos manter as *threads* próximas pra diminuir a latência, mas não queremos sobrecarregar um mesmo núcleo;
- `master` — Nessa política todas as *threads* são alocadas no mesmo local onde a *thread* principal (*master/primary*) está executando. Utilizado quando a comunicação entre *threads* é muito intensa e isso compença a perda de paralelismo. Pois ela gera disputa por recursos, mas reduz a latência de comunicação entre as *threads*;
- `true` — herda a política de afinidade da região paralela pai. Se não houver região pai, comporta-se de acordo com a política padrão definida pela implementação.
- `false` — Desativa a afinidade de *threads* o escalonador do sistema operacional é quem vai decidir onde cada *thread* ficará. Com isso pode haver migração de *threads* entre núcleos causando mais overhead e menos previsibilidade;

Reorganizamos o código de forma a possibilitar o teste de todas as políticas em uma única execução. Assim como realizado na Tarefa 12, o código foi executado com 1, 2, 4, 8, 16 e 32 *threads*, para que ao final pudéssemos analisar se houve influência dessas políticas de afinidade na eficiência do código.

O código foi executado no super computador da universidade utilizando o nó com o processador intel-128, cada teste foi executado 6 vezes para termos a certeza da constância dos resultados.

4 Resultados

Tendo em vista que a ideia é observar como as politicas de afinidade de *thread* afetam a escalabilidade do nosso programa, faremos uma tabela de escalabilidade para cada politica o metodo para criação dessa tabela já foi visto anteriormente na tarefa 12.

4.1 Spread

# cores	20x20x20	20x20x40	20x20x80	20x20x160	20x20x320	20x20x640
1	1.00	1.00	1.00	1.00	1.00	1.00
2	0.45	0.46	0.48	0.49	0.50	0.50
4	0.19	0.22	0.23	0.24	0.21	0.25
8	0.07	0.09	0.11	0.11	0.11	0.12
16	0.02	0.04	0.05	0.05	0.06	0.06
32	0.01	0.01	0.02	0.02	0.03	0.03

Table 1: Tabela de Escalabilidade para Política Spread

4.2 Close

# cores	20x20x20	20x20x40	20x20x80	20x20x160	20x20x320	20x20x640
1	1.00	1.00	1.00	1.00	1.00	1.00
2	0.44	0.47	0.47	0.49	0.51	0.50
4	0.19	0.22	0.23	0.24	0.25	0.25
8	0.07	0.09	0.10	0.11	0.12	0.12
16	0.02	0.04	0.04	0.05	0.06	0.06
32	0.01	0.01	0.02	0.02	0.03	0.03

Table 2: Tabela de Escalabilidade para Política Close

4.3 Master

# cores	20x20x20	20x20x40	20x20x80	20x20x160	20x20x320	20x20x640
1	1.00	1.00	1.00	1.00	1.00	1.00
2	0.01	0.01	0.03	0.06	0.11	0.12
4	0.00	0.00	0.01	0.01	0.02	0.04
8	0.07	0.09	0.11	0.11	0.11	0.12
16	0.02	0.04	0.05	0.05	0.06	0.06
32	0.01	0.01	0.02	0.02	0.03	0.03

Table 3: Tabela de Escalabilidade para Política Master

4.4 False

# cores	20x20x20	20x20x40	20x20x80	20x20x160	20x20x320	20x20x640
1	1.00	1.00	1.00	1.00	1.00	1.00
2	0.83	0.86	0.91	0.91	0.90	0.96
4	0.45	0.48	0.49	0.49	0.50	0.56
8	0.10	0.14	0.17	0.19	0.21	0.24
16	0.05	0.07	0.08	0.09	0.10	0.12
32	0.03	0.04	0.04	0.05	0.05	0.06

Table 4: Tabela de Escalabilidade para Política False

4.5 True

# cores	20x20x20	20x20x40	20x20x80	20x20x160	20x20x320	20x20x640
1	1.00	1.00	1.00	1.00	1.00	1.00
2	0.44	0.47	0.48	0.49	0.49	0.50
4	0.19	0.22	0.23	0.24	0.25	0.25
8	0.07	0.09	0.11	0.11	0.12	0.12
16	0.02	0.04	0.05	0.05	0.06	0.06
32	0.01	0.01	0.02	0.02	0.02	0.03

Table 5: Insira a legenda da sua tabela aqui

5 Analise dos Resultados

As politicas de afinidade de *thread* foram testadas na sequencia em que as tabelas foram montadas. Essas tabelas mostram o comportamento da escalabilidade relacionando o tamanho do problema com a quantidade de *thread* que a executaram, medindo através da sua eficiência.

5.1 Spread, True e Close

Essas três políticas apresentaram comportamentos muito similares, caracterizados por eficiências bastante baixas em praticamente todas as configurações. A eficiência caiu rapidamente com o aumento do número de threads, atingindo valores próximos a zero com 16 e 32 threads. O aumento do tamanho do problema trouxe pouca ou nenhuma melhora significativa na eficiência, indicando baixa escalabilidade fraca.

5.2 Master

A política MASTER apresentou a pior eficiência dentre todas as políticas avaliadas. Com 2 e 4 threads, as eficiências foram próximas de zero. Apenas a partir de 8 threads observou-se alguma melhora, embora ainda em níveis muito baixos (em torno de 0,11 a 0,12). A ampliação do tamanho do problema não resultou em ganhos expressivos.

Esse resultado era totalmente esperado, uma vez que essa política centraliza a execução na thread principal, criando um gargalo que impede a distribuição eficiente da carga de trabalho. Assim, a política MASTER é reconhecidamente inadequada para programas paralelos que visam alta escalabilidade, conforme foi comprovado experimentalmente

5.3 False

Esta política apresentou as melhores eficiências gerais em todas as configurações, especialmente com 2 e 4 threads, onde atingiu valores próximos a 0,9, indicando bom aproveitamento do paralelismo. Apesar de a eficiência cair progressivamente com o aumento do número de threads, ela se manteve superior às demais políticas. O aumento do tamanho do problema resultou em melhoras discretas na eficiência, evidenciando um comportamento típico de escalabilidade fraca.

Era esperado que a ausência de afinidade permitisse ao sistema operacional otimizar a distribuição das threads, minimizando contenções e promovendo melhor balanceamento de carga. De fato, isso foi confirmado pelos resultados. O desempenho atendeu às expectativas.

6 Conclusão

A análise realizada demonstrou que as políticas de afinidade de threads exercem um impacto significativo na eficiência e escalabilidade de programas paralelos. Os experimentos confirmaram que a escolha inadequada da política pode comprometer substancialmente o desempenho da aplicação.

De forma geral, os resultados obtidos foram coerentes com a literatura e com o comportamento esperado para cada política, destacando a necessidade de uma análise criteriosa do perfil computacional de cada aplicação antes de definir a política de afinidade de threads. A escalabilidade forte se mostrou limitada em todas as políticas quando o número de threads foi elevado excessivamente, enquanto a escalabilidade fraca apresentou melhoras modestas, especialmente na política FALSE.

Por fim, esta avaliação reforça a importância de realizar experimentos empíricos para orientar decisões de configuração em sistemas paralelos, visando sempre o melhor aproveitamento dos recursos computacionais disponíveis.