Relatório Tarefa 21: Programação híbrida MPI + OpenMP para simulação de dissipação de calor

Aluno: Cristovao Lacerda Cronje

1. Introdução

Nesta atividade, implementamos e analisamos um código em C para a simulação da dissipação de calor em uma placa 2D quadrada. A abordagem adotada foi a programação híbrida, combinando MPI para distribuir o trabalho entre processos, com OpenMP para paralelismo dentro de cada processo. A distribuição dos dados foi feita por tiras horizontais da placa, com cada processo responsável pela simulação em sua respectiva faixa.

Os testes foram realizados no cluster NPAD, utilizando até 4 nós, com configurações variadas de processos MPI e threads OpenMP. A avaliação de desempenho considerou três tamanhos de problemas (`small`, `medium` e `large`) e diversas combinações de nós e processos, respeitando limitações de hardware detectadas no cluster.

2. Objetivos

- Implementar a simulação da equação de transferência de calor utilizando programação híbrida MPI + OpenMP.
- Avaliar o desempenho da simulação com diferentes tamanhos de problema, números de nós, processos MPI e threads por processo.
- Minimizar o bloqueio durante a comunicação MPI, buscando otimizar o tempo total de execução.
- Analisar a influência da comunicação e do paralelismo no desempenho global da aplicação.

3. Metodologia

3.1 Estrutura do Código

O código implementado realiza a simulação da dissipação de calor distribuindo a placa 2D em tiras horizontais, uma para cada processo MPI. A cada iteração, os processos computam localmente a equação do calor para suas tiras usando paralelismo OpenMP.

A comunicação entre processos ocorre nas bordas das tiras para troca de dados das células vizinhas, fundamental para a correta simulação da transferência de calor entre faixas. Para evitar bloqueios durante a comunicação, foram utilizadas operações MPI não bloqueantes (`MPI_Isend`, `MPI_Irecv`), permitindo sobreposição de comunicação e computação.

Por exemplo, em uma configuração com 2 nós e 4 processos, seriam atribuídos 2 threads por processo. Isso aproveita os 8 núcleos por nó disponíveis no cluster, maximizando a utilização local dos recursos de CPU.

Detalhes do paralelismo:

- **MPI:** divide a placa em fatias horizontais.
- **OpenMP:** múltiplas threads aceleram o cálculo dentro de cada processo.

- **Threads por processo:** calculadas como `2 × (processos / nós)`, conforme a topologia do cluster.

4. Resultados e Análise

4.1 Resultados obtidos

problem_size	*nodes*	processes	threads_per_process	total_time	*computation_time*	communication_time	speedup	efficiency
small	*1*	1	2	8.008795	*8.007075*	0.0	1.0	1.0
small	*1*	2	4	3.264892	*3.189794*	0.017256	2.4530045710547235	1.2265022855273617
small	*1*	4	8	0.583765	*0.550607*	0.012013	13.719210641268317	3.4298026603170793
small	*2*	2	2	4.363787	*4.103122*	0.13918	1.8352854985818507	0.9176427492909254
small	*2*	4	4	1.099163	*0.852732*	0.11462	7.286266914006385	1.8215667285015962
small	*2*	8	8	0.475924	*0.315341*	0.072803	16.827886385221166	2.103485798152646
small	*4*	4	2	2.363831	*1.98621*	0.200614	3.388057352661844	0.847014338165461
small	*4*	8	4	0.995949	*0.639805*	0.175454	8.041370592269283	1.0051713240336604
medium	*1*	1	2	33.674202	*33.666787*	0.0	1.0	1.0
medium	*1*	2	4	11.753369	*11.607042*	0.019397	2.8650680498502177	1.4325340249251088
medium	*1*	4	8	1.836733	*1.73165*	0.016069	18.333749107790844	4.583437276947711
medium	*2*	2	2	18.453578	*17.283304*	0.679212	1.8248061161905837	0.9124030580952919
medium	*2*	4	4	3.587617	*3.271748*	0.12494	9.386231027448025	2.346557756862006
medium	*2*	8	8	1.21242	*0.952984*	0.113947	27.774370267729005	3.4717962834661256
medium	*4*	4	2	8.33766	*7.658435*	0.252683	4.038807291254381	1.0097018228135952
medium	*4*	8	4	2.536858	*2.190665*	0.185298	13.273979860126188	1.6592474825157735
large	*2*	2	2	64.783935	*63.846391*	0.469253	1.0	0.5
large	*2*	4	4	13.502725	*13.009292*	0.139788	4.797841546798887	1.1994603866997218
large	*2*	8	8	28.938489	*16.30038*	0.1497	2.238677181797571	0.27983464772469635
large	*4*	4	2	34.158938	*32.634492*	0.509967	1.8965441782762684	0.4741360445690671
large	*4*	8	4	9.060317	*8.542502*	0.251	7.150294520600108	0.8937868150750135

*Observa-se que, em algumas configurações, **a eficiência ultrapassou o valor 1.0 (ou 100%).** Isso pode ocorrer devido a fatores como melhor aproveitamento da hierarquia de memória (ex.: cache), variação na alocação de recursos no cluster ou redução do impacto de overhead em execuções menores. Embora não seja comum em teoria ideal, esse fenômeno pode ocorrer em ambientes reais.

**Em algumas execuções, especialmente com apenas 1 processo MPI, o tempo de comunicação foi registrado como zero, o que é esperado, já que não há troca de dados entre processos. Além disso, mesmo em configurações com múltiplos processos, a comunicação foi realizada com operações não bloqueantes (MPI_Isend e MPI_Irecv), permitindo sobreposição com a computação. Como o volume de dados trocados por iteração é pequeno (apenas linhas de borda), o tempo de comunicação ficou muitas vezes abaixo da precisão de medição (MPI_Wtime), resultando em valores nulos ou desprezíveis.

4.2 Explicação detalhada dos resultados

Escalabilidade e Paralelismo

Observou-se redução significativa no tempo total de execução com o aumento de processos e nós, especialmente para problemas `medium` e `large`. A combinação de MPI para distribuir carga e OpenMP para acelerar a computação local foi eficaz.

Tamanho do Problema

- Problemas `small` mostraram menor ganho com paralelismo extremo (overhead supera benefício).
- Problemas `medium` e `large` escalaram melhor, com speedup e eficiência visivelmente superiores nas configurações com mais nós e processos.

Limitações de Execução

A configuração large não pôde ser executada em apenas 1 nó devido à limitação de memória RAM disponível. Isso ocorre porque o código aloca múltiplas matrizes grandes para cada processo, exigindo uma quantidade significativa de memória. A utilização de múltiplos nós permite dividir essa carga de memória entre eles, viabilizando a execução do problema. Além disso, não foram consideradas configurações onde o número de processos MPI fosse menor que o número de nós, pois isso resultaria em nós ociosos e não permitiria uma comparação justa entre as execuções.

4.3 Escalabilidade forte e fraca

- **Escalabilidade forte:**
  Observada quando o número de processos aumenta com o problema fixo. Por exemplo, para `small`, o tempo caiu de 8s (1 processo) para 0.47s (8 processos em 2 nós), com speedup > 16x. No entanto, eficiência não é perfeita devido ao crescimento do tempo de comunicação.
- **Escalabilidade fraca:**
  Para problemas maiores (`medium` e `large`) usando mais nós, o tempo total aumentou pouco, indicando que o sistema lida bem com o aumento conjunto de carga e recursos — boa escalabilidade fraca.

4.4 Overhead e Comunicação

- **Overhead:**
  É o custo extra da gestão paralela (sincronizações, mensagens, espera). Observado principalmente nas execuções com muitos processos em problemas pequenos.
- **Tempo de comunicação:**
  Embora aumente com o número de processos, o tempo de comunicação representou apenas 1% a 3% do tempo total em quase todas as execuções. Isso se deve ao uso de comunicação não bloqueante (`MPI_Isend` + `MPI_Waitall`), que permite sobreposição com a computação.

5. Conclusão

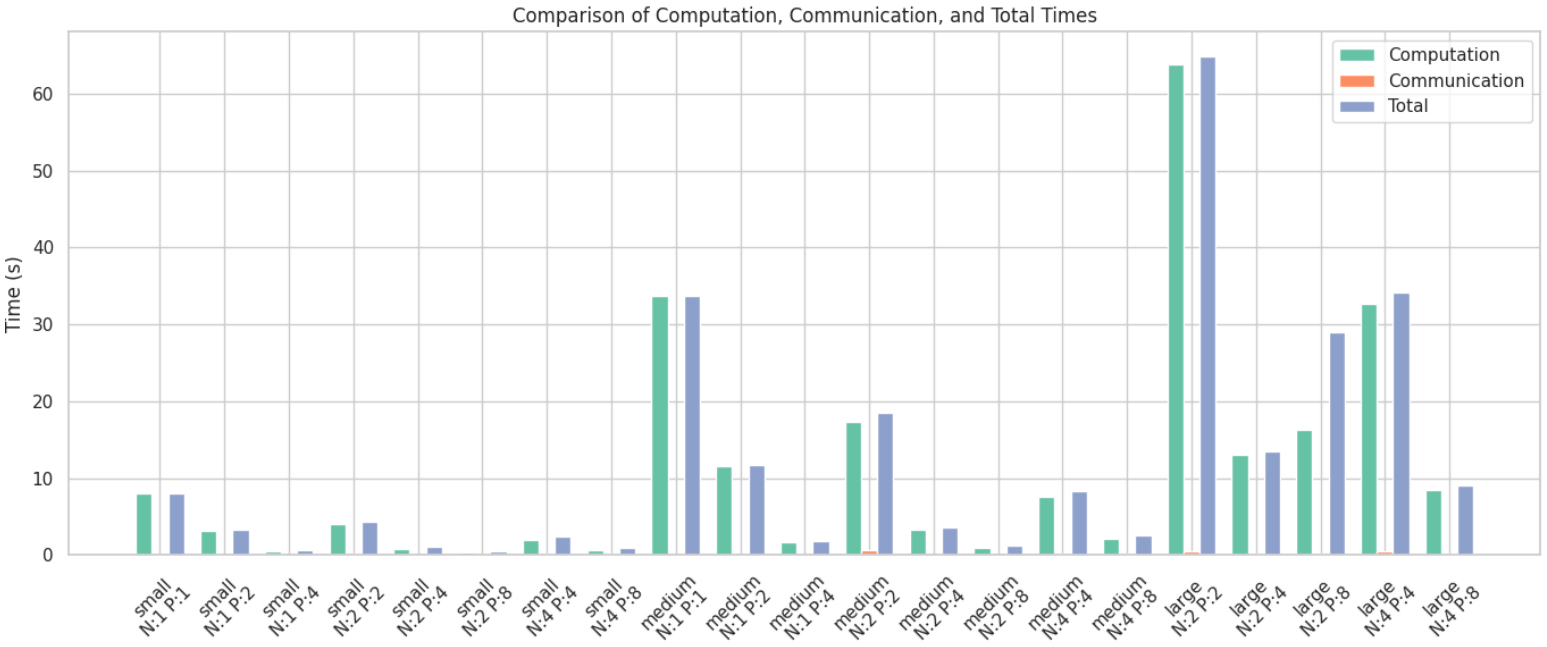
A implementação híbrida MPI + OpenMP mostrou-se eficaz para acelerar a simulação da dissipação de calor. O código escalou bem até 4 nós e 8 processos, especialmente para tamanhos `medium` e

`large`. A sobreposição entre computação e comunicação reduziu o impacto da comunicação, resultando em bons speedups e eficiências.

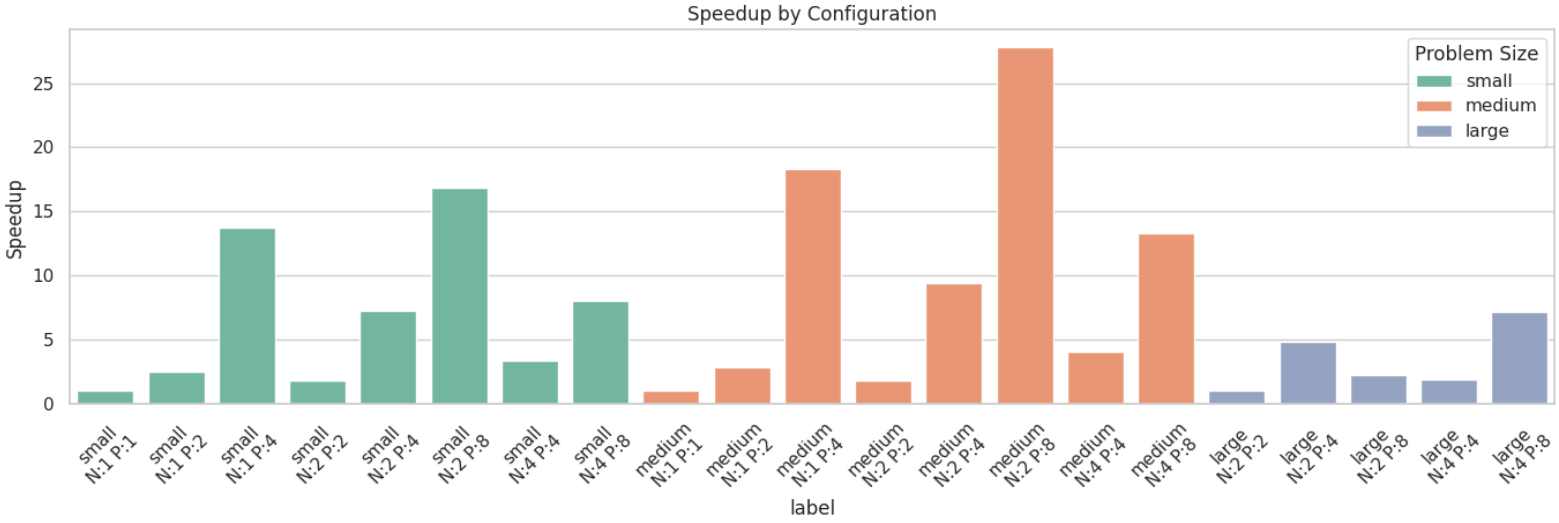
A análise reforça que o uso de programação híbrida é essencial em clusters modernos, aproveitando ao máximo os recursos disponíveis por nó (multicore) e distribuindo a carga eficientemente com MPI.

6. Espaço para Gráficos

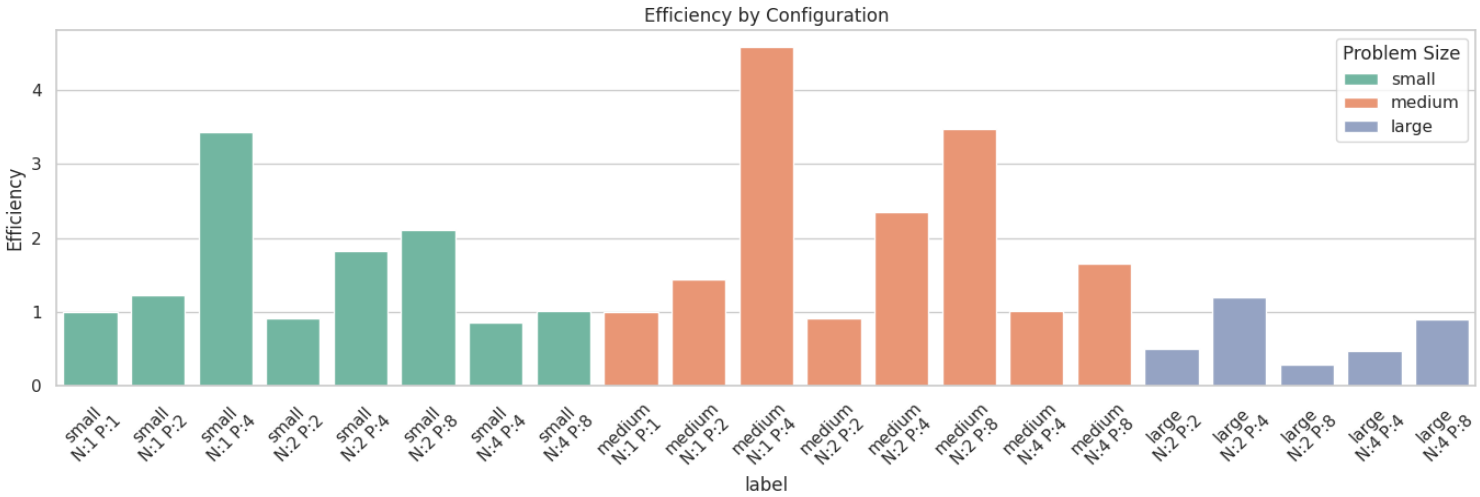
- **Gráfico 1:** Computation, Communication e Total Time (por configuração)



- **Gráfico 2:** Speedup (por tamanho de problema)



- **Gráfico 3:** Eficiência (por tamanho de problema)


Efficiency by Configuration
Problem Size
small
medium
large
Efficiency
0
1
2
3
4
label
small N:1 P:1
small N:1 P:2
small N:1 P:4
small N:2 P:2
small N:2 P:4
small N:2 P:8
small N:4 P:4
small N:4 P:8
medium N:1 P:1
medium N:1 P:2
medium N:1 P:4
medium N:2 P:2
medium N:2 P:4
medium N:2 P:8
medium N:4 P:4
medium N:4 P:8
large N:2 P:2
large N:2 P:4
large N:2 P:8
large N:4 P:4
large N:4 P:8