UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
IDCA3703 - PROGRAMAÇÃO PARALELA

TAREFA 15 - DIFUSÃO DE CALOR USANDO MPI
RELATÓRIO DE EXECUÇÃO

ERNANE FERREIRA ROCHA JUNIOR

NATAL/RN, 15 DE MAIO DE 2025

SUMÁRIO

1. INTRODUÇÃO

A difusão de calor é um fenômeno físico amplamente estudado na engenharia e nas ciências computacionais por sua relevância em processos térmicos, modelagem matemática e simulações numéricas. No contexto da computação de alto desempenho (*HPC*), esse tipo de problema é comumente utilizado como *benchmark* para avaliar desempenho de sistemas paralelos e estratégias de comunicação entre processos.

Este trabalho tem como objetivo implementar uma simulação da difusão de calor em uma barra unidimensional (1D), utilizando programação paralela com *MPI* (*Message Passing Interface*). A barra é dividida em segmentos, cada um atribuído a um processo *MPI*, sendo que cada processo é responsável por calcular a evolução temporal do calor em sua parte da barra. Para garantir a coerência nos cálculos, é necessária a troca de informações nas bordas entre processos vizinhos — o que é realizado por meio de comunicação ponto a ponto.

Para estudar o impacto da forma de comunicação no desempenho do sistema, foram desenvolvidas três versões da simulação, cada uma adotando uma abordagem distinta de comunicação:

- Comunicação bloqueante, usando *MPI_Send* e *MPI_Recv*;
- Comunicação não bloqueante com espera, utilizando *MPI_Isend*, *MPI_Irecv* e *MPI_Waitall*;
- Comunicação não bloqueante com sobreposição, fazendo uso de *MPI_Isend*, *MPI_Irecv* e *MPI_Test* para computar pontos internos enquanto aguarda o recebimento das bordas.

A partir dessas três abordagens, foram realizados experimentos variando o número de processos *MPI* e medindo o tempo de execução de cada versão. Os resultados obtidos permitem analisar o comportamento de cada técnica, evidenciar os ganhos (ou não) com a sobreposição de comunicação e computação e identificar possíveis limitações na escalabilidade do problema proposto.

2. METODOLOGIA

A simulação da difusão de calor foi baseada na discretização de uma barra unidimensional com condição inicial concentrada no centro e bordas inicialmente frias. A equação utilizada é a forma explícita da equação de difusão de calor:

$$u[i] = u[i] + \alpha \cdot (u[i-1] - 2u[i] + u[i+1])$$

onde $u[i]$ representa a temperatura no ponto α é o coeficiente de difusão térmica. Essa equação foi aplicada iterativamente por um número fixo de passos de tempo.

A barra foi dividida igualmente entre os processos *MPI*, sendo que cada processo é responsável por um segmento da barra. Para possibilitar o cálculo local, mas levando em conta a vizinhança global, cada processo mantém duas células fantasmas (*ghost cells*): uma à esquerda e outra à direita, utilizadas exclusivamente para armazenar os valores recebidos dos processos vizinhos. Essa estratégia permite que a atualização das células reais ocorra localmente após a sincronização com os vizinhos imediatos.

Foram implementadas três versões distintas da simulação, diferenciadas pela estratégia de comunicação entre os processos:

- **Versão 1** – Comunicação bloqueante: Utiliza *MPI_Send* e *MPI_Recv* para realizar a troca de bordas entre processos. Os processos ficam bloqueados durante as operações de envio e recebimento, aguardando que a comunicação seja concluída para prosseguir com o cálculo.
- **Versão 2** – Comunicação não bloqueante com espera: Utiliza *MPI_Isend* e *MPI_Irecv* para iniciar a comunicação de forma assíncrona, mas utiliza *MPI_Waitall* para aguardar a conclusão das operações antes de iniciar os cálculos.
- **Versão 3** – Comunicação não bloqueante com sobreposição: Também utiliza *MPI_Isend* e *MPI_Irecv*, porém aplica *MPI_Test* para verificar a chegada das bordas. Enquanto os dados das bordas ainda não chegaram, os processos avançam com o cálculo dos pontos internos da barra, o que permite sobrepor comunicação e computação e potencialmente esconder latência.

O código foi desenvolvido em linguagem *C* com *MPI*, compilado com *mpicc* e executado no ambiente de supercomputação do *NPAD* (*UFRN*), utilizando a partição *intel-512*. O experimento consistiu em uma simulação com barra de $N = 10^7$ pontos e 10^5 passos de tempo. O código foi executado com diferentes quantidades de processos *MPI* (2, 4, 8, 16 e 32), em um script *SLURM* automatizado, e os tempos de execução de cada versão foram registrados utilizando a função *MPI_Wtime*.

Essa configuração foi projetada para simular uma carga computacional significativa e permitir a avaliação de desempenho em diferentes níveis de paralelismo, observando os impactos da comunicação sobre o tempo total de execução.

3. RESULTADOS

Com o objetivo de avaliar o impacto das diferentes estratégias de comunicação na eficiência da simulação paralela da difusão de calor, foi conduzido um experimento variando a quantidade de processos *MPI* de 2 a 32. Para cada valor, foram executadas as três versões do programa: com comunicação bloqueante, com comunicação não bloqueante e espera (*Wait*), e com comunicação não bloqueante e sobreposição com *Test*.

A simulação foi configurada com um total de 10 milhões de células e 100 mil passos de tempo, o que representa uma carga computacional suficientemente grande para destacar os efeitos da paralelização e da latência de comunicação.

A tabela a seguir apresenta os tempos totais de execução (em segundos) para cada versão da simulação, de acordo com a quantidade de processos utilizados:

Nº de Processos	**Bloqueante (S*end*/*Recv*)**	**Não Bloqueante + *Wait***	**Não Bloqueante + *Test***
2	1138.36 s	1140.14 s	1144.12 s
4	0746.66 s	0746.57 s	0745.03 s
8	0685.91 s	0685.95 s	0693.82 s
16	0686.11 s	0684.51 s	0689.70 s
32	0687.53 s	0687.63 s	0680.97 s

A partir dos dados, observa-se que a maior redução no tempo de execução ocorre entre 2 e 4 processos, o que demonstra um bom aproveitamento da paralelização inicial. A partir de 8 processos, os ganhos adicionais tornam-se marginais, e para 16 e 32 processos os tempos praticamente estabilizam, indicando um ponto de saturação da escalabilidade para o problema proposto.

Em relação às versões, nota-se que a estratégia com *MPI_Test* não apresentou vantagem significativa em todos os cenários. Para 4 e 32 processos, ela foi ligeiramente mais rápida que as demais, o que evidencia algum ganho com a sobreposição de comunicação e computação. No entanto, para 8 e 16 processos, o desempenho foi comparável ou inferior. Isso pode ser atribuído ao equilíbrio entre o tempo de comunicação e a quantidade de trabalho computacional restante, bem como à granularidade da carga por processo.

De forma geral, a versão com *MPI_Wait* apresentou desempenho muito semelhante à versão bloqueante, o que confirma que, embora a comunicação seja iniciada de forma assíncrona, não há real sobreposição se o processo espera todas as comunicações antes de calcular.

4. CONCLUSÃO

A simulação da difusão de calor em uma barra unidimensional, paralelizada com *MPI*, permitiu avaliar de forma prática o impacto de diferentes estratégias de comunicação entre processos. As três versões implementadas — comunicação bloqueante, comunicação não bloqueante com espera, e comunicação não bloqueante com sobreposição — apresentaram comportamentos distintos, embora com tempos de execução semelhantes em cenários de maior paralelismo.

A versão com comunicação bloqueante mostrou-se simples de implementar e eficiente até certo ponto, com bom desempenho nas primeiras duplicações de processos. A versão com *MPI_Wait* demonstrou desempenho praticamente idêntico, confirmando que apenas iniciar a comunicação de forma não bloqueante não é suficiente para obter ganhos reais se o processo ainda aguarda explicitamente por todas as mensagens antes de continuar.

A versão com *MPI_Test*, que introduz a sobreposição de comunicação e computação, obteve desempenho ligeiramente superior em alguns cenários. Esse ganho, no entanto, não foi expressivo, e em certos casos foi superado pelas outras versões. Isso sugere que a sobreposição pode ser vantajosa, mas depende da proporção entre tempo de comunicação e quantidade de trabalho computacional disponível para ser executado enquanto as mensagens estão em trânsito.

Em termos de escalabilidade, a simulação apresentou boa redução de tempo de execução ao passar de 2 para 4 e 8 processos. Contudo, a partir de 16 processos os ganhos se estabilizam, indicando uma saturação do paralelismo para a carga configurada. Esse comportamento é típico em aplicações com comunicação ponto a ponto intensiva e pode ser mitigado com cargas maiores ou algoritmos que minimizem a dependência entre processos.

Em resumo, a atividade atendeu aos objetivos propostos, demonstrando a viabilidade da simulação paralela da difusão de calor, o funcionamento de diferentes modelos de comunicação com *MPI* e os efeitos da sobreposição de comunicação e computação sobre o desempenho da aplicação.

5. ANEXOS

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