Universidade Federal do Rio Grande do Norte
Departamento de Engenharia da Computação e Automação
DCA3703 - Programação Paralela
**Tarefa 14: Latência de comunicação usando MPI**
**Aluno:** Daniel Bruno Trindade da Silva

# 1 Introdução

Este relatório tem como objetivo apresentar os conhecimentos adquiridos durante a realização da Tarefa 15 da disciplina de **Programação Paralela**. A atividade teve como objetivo mensurar o tempo de comunicação entre os processos usando ferramentas do MPI. Para isso será realizada a implementação e análise de um programa que simulasse a difusão de calor em uma barra 1D.

# 2 Enunciado

Implemente uma simulação da difusão de calor em uma barra 1D, dividida entre dois ou mais processos MPI. Cada processo deve simular um trecho da barra com células extras para troca de bordas com vizinhos. Implemente três versões: uma com `MPI_Send/MPI_Recv`, outra com `MPI_Isend/MPI_Irecv` e `MPI_Wait`, e uma terceira usando `MPI_Test` para atualizar os pontos internos enquanto aguarda a comunicação. Compare os tempos de execução e discuta os ganhos com sobreposição de comunicação e computação.

# 3 Desenvolvimento

Para atender aos requisitos da atividade, foram desenvolvidas três versões de um programa em C com MPI para simular a difusão de calor unidimensional. A equação de difusão do calor discretizada utilizada é:

$$u_i' = u_i + \alpha(u_{i-1} - 2u_i + u_{i+1})$$

onde $u_i$ é a temperatura no ponto $i$, $u_i'$ é a nova temperatura após um intervalo de tempo, e $\alpha$ é a constante de difusividade térmica.

Em todas as implementações, a barra 1D de tamanho $N$ é dividida entre os $P$ processos MPI, onde cada processo é responsável por calcular a temperatura de um segmento de $N/P$ células. Para permitir a troca de informações de contorno com os processos vizinhos, cada processo aloca duas células extras (ghost cells), uma no início e outra no fim de seu subdomínio local. A inicialização da barra atribui um valor de temperatura elevado (e.g., 100.0) a uma célula central da barra global e zero às demais. As condições de contorno globais da barra são fixas em 0.0. O tempo de execução de cada simulação é medido utilizando `MPI_Wtime()` e o tempo máximo entre todos os processos é obtido com `MPI_Reduce`.

As três versões implementadas diferem na forma como a comunicação das células de borda é realizada

## Versão 1: `MPI_Send` / `MPI_Recv`

Nesta implementação, a troca de dados das células de borda é realizada utilizando as funções bloqueantes `MPI_Send` e `MPI_Recv`. Cada processo:

- Envia o valor da sua primeira célula útil (fronteira interna esquerda) para o vizinho da esquerda (se existir).
- Recebe o valor da célula de borda do vizinho da esquerda em sua ghost cell esquerda (se existir).

- Envia o valor da sua última célula útil (fronteira interna direita) para o vizinho da direita (se existir).
- Recebe o valor da célula de borda do vizinho da direita em sua ghost cell direita (se existir).

Após a conclusão de todas as comunicações, o cálculo da nova temperatura para todas as células locais é realizado. A natureza bloqueante dessas chamadas implica que um processo pode ficar ocioso enquanto espera a conclusão de uma operação de envio ou recebimento.

### Versão 2: `MPI_Isend` / `MPI_Irecv` e `MPI_Waitall`

A segunda implementação utiliza as versões não bloqueantes das chamadas de comunicação, `MPI_Isend` e `MPI_Irecv`, para a troca das células de borda.

- Para cada vizinho (esquerdo e direito, se existirem), o processo inicia um recebimento não bloqueante (`MPI_Irecv`) para a ghost cell correspondente.
- Em seguida, inicia um envio não bloqueante (`MPI_Isend`) da sua célula de fronteira interna para o respectivo vizinho.

Após iniciar todas as operações de comunicação necessárias, a função `MPI_Waitall` é chamada. Esta função bloqueia o processo até que todas as comunicações não bloqueantes (tanto envios quanto recebimentos) especificadas sejam concluidas. Somente após o retorno de `MPI_Waitall`, o cálculo da nova temperatura para todas as células locais é efetuado. Nesta abordagem, embora as operações sejam não bloqueantes, a computação dos pontos da malha só ocorre após a finalização de toda acomunicação.

### Versão 3: `MPI_Isend` / `MPI_Irecv` e `MPI_Test`

A terceira implementação (arquivo `mpi_test.c`) visa sobrepor a computação com a comunicação, utilizando `MPI_Isend`, `MPI_Irecv` e `MPI_Test`[cite: 6]. O fluxo de operações em cada iteração é:

- São iniciadas as operações não bloqueantes de recebimento (`MPI_Irecv`) das ghost cells dos vizinhos esquerdo e direito, se existirem.
- São iniciadas as operações não bloqueantes de envio (`MPI_Isend`) das células de fronteira internas para os respectivos vizinhos, se existirem.
- Imediatamente após iniciar as comunicações, o cálculo dos **pontos internos** do subdomínio local (aqueles que não dependem dos valores das ghost cells recém-recebidas) é realizado. Isso permite que a computação desses pontos ocorra em paralelo com a transferência de dados das bordas.
- Em seguida, o programa entra em laços que utilizam `MPI_Test` para verificar repetidamente se os dados das ghost cells já foram recebidos. `MPI_Test` é uma chamada não bloqueante que verifica o status de uma comunicação pendente.
- Assim que `MPI_Test` indica que um recebimento foi concluído para uma determinada borda (e.g., esquerda), o cálculo do ponto adjacente a essa borda (e.g., $u'_1$) é realizado. O mesmo processo é feito para a outra borda.
- Finalmente, `MPI_Wait` é chamado para cada operação de envio pendente para garantir sua conclusão antes de prosseguir para a próxima iteração, embora o impacto principal da sobreposição seja obtido ao computar os pontos internos enquanto se espera pelos recebimentos.

Esta abordagem tem o potencial de reduzir o tempo ocioso do processador, melhorando a eficiência da paralelização.