

Análise de Latência e Largura de Banda em MPI

Caracterização do Desempenho de Comunicação Ponto-a-Ponto

Werbert Arles de Souza Barradas

Universidade Federal do Rio Grande do Norte (UFRN)
DCA3703 - Programação Paralela

25 de Setembro de 2025

Introdução: O Desafio da Comunicação

Objetivo Principal

Analisar o tempo de comunicação em função do tamanho da mensagem para caracterizar a plataforma nos regimes de **Latência** (τ) e **Largura de Banda** (β).

O Modelo Latência-Largura de Banda

O tempo de comunicação é definido por:

$$\mathbf{T} = \tau + \frac{\mathbf{L}}{\beta}$$

- τ : Latência (tempo fixo, *overhead*).
- \mathbf{L} : Tamanho da mensagem.
- β : Largura de Banda (taxa de transferência).

Metodologia: O Benchmark Ping-Pong

- **Implementação:** Código C com **MPI** (Message Passing Interface).
- **Configuração:** Exatamente **2 Processos** (`mpirun -np 2`).
- **Padrão:** **Ping-Pong** (P0 envia, P1 recebe/responde, P0 recebe).

Parâmetros de Teste

- **Medição:** Utilização de `MPI_Wtime` (e.g., 1000 iterações) para precisão estatística.
- **Tamanhos de Mensagem:** Sequência geométrica (dobrando) de **8 bytes** a **64 MB**.

Resultados Gráficos: A Curva de Comunicação

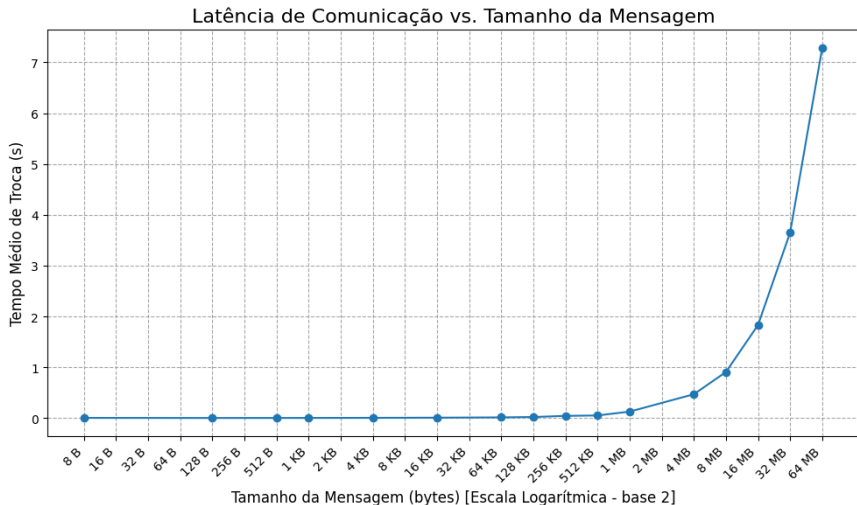


Figura: Tempo Médio por Troca vs. Tamanho da Mensagem (Eixo X Logarítmico)

Análise: Latência Domínio vs. Largura de Banda Domínio

Regime de Latência (τ)

Onde: Mensagens pequenas (**16 B a ~8 KB**).

- A curva é **plana**.
- O tempo é dominado pelo **custo fixo** de inicializar o protocolo MPI.
- O custo de mover os dados é irrelevante.

Regime de Largura de Banda (β)

Onde: Mensagens grandes (**~64 KB a 4 MB**).

- A curva **cresce acentuadamente**.
- O tempo é dominado pela **transferência real de bytes**.
- O sistema atingiu a capacidade máxima de taxa de dados da interconexão.

O Ponto de Transição

A transição ocorre quando $\tau \approx L/\beta$, que no nosso experimento se dá aproximadamente entre 10 KB e 60 KB.

Conclusão: Otimização da Comunicação

- **Sucesso do Experimento:** O *benchmark* validou o Modelo Latência-Largura de Banda e caracterizou a interconexão da plataforma.
- **Impacto na Otimização (Lei de Amdahl da Comunicação):**
 - ① Algoritmos com **muitas mensagens pequenas** são limitados pela **Latência** (τ). A otimização deve ser o *message aggregation* (agrupar mensagens).
 - ② Algoritmos com **mensagens grandes** são limitados pela **Largura de Banda** (β).
- **Relevância:** A identificação dos regimes permite ao programador MPI escolher a granularidade de comunicação correta para maximizar o *speedup*.

ping_pong_mpi.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <mpi.h>
4
5  int main(int argc, char** argv) {
6      int rank, size;
7      double start_time, end_time;
8
9      // Inicializa o ambiente MPI
10     MPI_Init(&argc, &argv);
11
12     // Obtém o rank (ID do processo) e o tamanho do comunicador
13     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
14     MPI_Comm_size(MPI_COMM_WORLD, &size);
15
16     // O programa deve ser executado com exatamente 2 processos
17     if (size != 2) {
18         if (rank == 0) {
19             fprintf(stderr, "Este programa requer exatamente 2 processos para
ser executado.\n");
20         }
21         MPI_Finalize();
22         return 1;
23     }
24
25     // Vetor de tamanhos de mensagem a serem testados, incluindo 4 MB
26     int message_sizes[] = {8, 128, 512, 1024, 4096, 16384, 65536, 131072,
262144, 524288, 1048576, 4194304};
27     int num_sizes = sizeof(message_sizes) / sizeof(int);
28
29     // Número de trocas para cada tamanho de mensagem
30     int num_iterations = 1000;
31
32     for (int i = 0; i < num_sizes; i++) {
33         int current_size = message_sizes[i];
34
35         // Aloca o buffer para a mensagem
36         char *send_buffer = (char*) malloc(current_size);
37         char *recv_buffer = (char*) malloc(current_size);
38
39         // O processo 0 inicia a comunicação
40         if (rank == 0) {
41             start_time = MPI_Wtime();
42
43             for (int j = 0; j < num_iterations; j++) {
44                 // Envia a mensagem para o processo 1
45                 MPI_Send(send_buffer, current_size, MPI_CHAR, 1, 0,
MPI_COMM_WORLD);
46                 // Recebe a mensagem de volta do processo 1
47                 MPI_Recv(recv_buffer, current_size, MPI_CHAR, 1, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

```
48     }
49
50     end_time = MPI_Wtime();
51     double total_time = end_time - start_time;
52     double average_time_ms = (total_time / (2 * num_iterations)) * 1000;
53
54     // Exibe os resultados
55     printf("Tamanho da Mensagem: %d bytes, Tempo Médio por troca: %f
ms\n", current_size, average_time_ms);
56 }
57
58 // 0 processo 1 responde à comunicação
59 else if (rank == 1) {
60     for (int j = 0; j < num_iterations; j++) {
61         // Recebe a mensagem do processo 0
62         MPI_Recv(recv_buffer, current_size, MPI_CHAR, 0, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
63         // Envia a mesma mensagem de volta
64         MPI_Send(send_buffer, current_size, MPI_CHAR, 0, 0,
MPI_COMM_WORLD);
65     }
66 }
67
68 free(send_buffer);
69 free(recv_buffer);
70
71 }
72
73 // Finaliza o ambiente MPI
74 MPI_Finalize();
75
76 return 0;
77 }
```