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 14 da disciplina de **Programação Paralela**. A atividade consistiu na implementação e análise de um sistema de troca de mensagens utilizando a biblioteca MPI (Message Passing Interface). O foco principal foi observar como o tempo de comunicação entre dois processos se comporta à medida que o tamanho das mensagens aumenta, destacando aspectos como latência e largura de banda.

2 Enunciado

Implemente um programa MPI com exatamente dois processos. O processo 0 deve enviar uma mensagem ao processo 1, que imediatamente responde com a mesma mensagem. Meça o tempo total de execução de múltiplas trocas consecutivas dessa mensagem,utilizando `MPI_Wtime`. Registre os tempos para diferentes tamanhos, desde mensagens pequenas (como 8 bytes) até mensagens maiores (como 1 MB ou mais). Analise o graficamente o tempo em função do tamanho da mensagem e identifique os regimes onde a latência domina e onde a largura de banda se torna o fator principal.

3 Desenvolvimento

Conforme solicitado no enunciado, desenvolvemos um programa do tipo "*ping-pong*", utilizando a biblioteca MPI. O programa cria exatamente dois processos que se comunicam entre si, realizando sucessivas trocas de mensagens.

A lógica da implementação foi estruturada da seguinte forma: o processo de rank 0 envia uma mensagem ao processo de rank 1, que imediatamente a recebe e devolve a mesma mensagem para o processo 0. Esse ciclo de envio e recebimento foi repetido diversas vezes para cada tamanho de mensagem, com o objetivo de obter uma medição mais precisa do tempo de execução.

Para mensurar o tempo total das trocas de mensagens, utilizamos a função `MPI_Wtime`, que fornece uma medida de tempo de alta resolução adequada para benchmarking em aplicações paralelas. Antes de iniciar as medições, foi utilizada a função `MPI_Barrier` para sincronizar os dois processos, garantindo que ambos estivessem prontos para iniciar a troca de mensagens ao mesmo tempo.

3.1 Código

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NUM_REPETICOES 1000  // Número de trocas para cada tamanho

int main(int argc, char** argv) {
 int rank, size;
 MPI_Init(&argc, &argv);
```

```
 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
 MPI_Comm_size(MPI_COMM_WORLD, &size);

 if (size != 2) {
  if (rank == 0) {
   printf("Este programa requer exatamente 2 processos.\n");
  }
  MPI_Finalize();
  return 0;
 }

 // Tamanhos de mensagens a serem testadas (em bytes)
 int tamanhos[] = {8, 1024, 10*1024, 100*1024, 1024*1024};
 int num_tamanhos = sizeof(tamanhos) / sizeof(tamanhos[0]);

 for (int t = 0; t < num_tamanhos; t++) {
  int tamanho_msg = tamanhos[t];
  char* buffer = (char*)malloc(tamanho_msg);
  memset(buffer, 'A', tamanho_msg);

  MPI_Barrier(MPI_COMM_WORLD);  // Sincroniza antes de medir o tempo
  double start_time = MPI_Wtime();

  for (int i = 0; i < NUM_REPETICOES; i++) {
   if (rank == 0) {
    MPI_Send(buffer, tamanho_msg, MPI_CHAR, 1, 0, MPI_COMM_WORLD);
    MPI_Recv(buffer, tamanho_msg, MPI_CHAR, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
   } else if (rank == 1) {
    MPI_Recv(buffer, tamanho_msg, MPI_CHAR, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    MPI_Send(buffer, tamanho_msg, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
   }
  }

  double end_time = MPI_Wtime();

  if (rank == 0) {
   double total_time = end_time - start_time;
   printf("Tamanho da mensagem: %d bytes, Tempo total para %d trocas: %f segundos\n",
   tamanho_msg, NUM_REPETICOES, total_time);
  }

  free(buffer);
 }

 MPI_Finalize();
 return 0;
}
```

4 Resultados

Tamanho da Mensagem	Tempo total para 1000 trocas (s)
8 bytes	0.003100
1 KB (1024 bytes)	0.006715
10 KB (10240 bytes)	0.014761
100 KB (102400 bytes)	0.079849
1 MB (1048576 bytes)	0.680651

Tabela 1: Resultados do tempo total de comunicação para diferentes tamanhos de mensagem

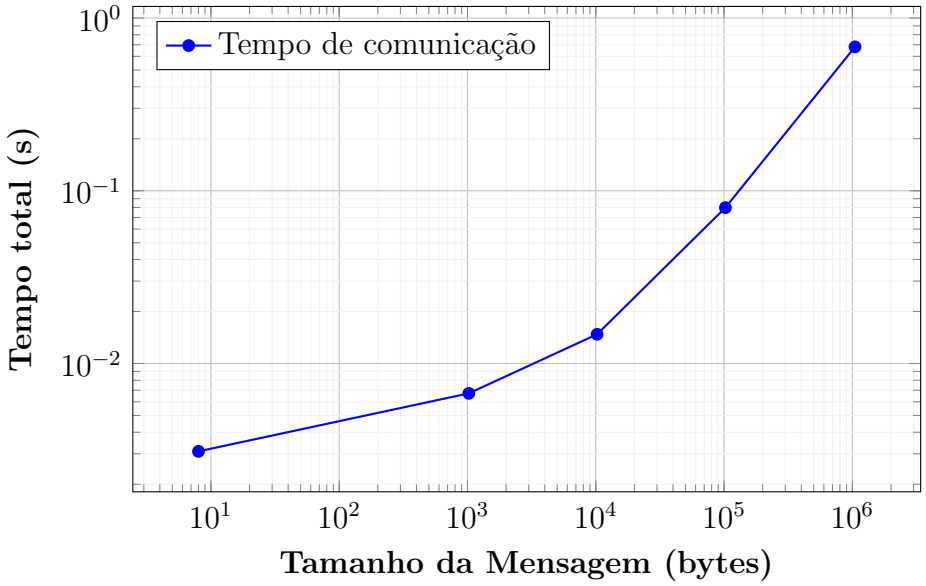

Figura 1: Tempo total de 1000 trocas em função do tamanho da mensagem

5 Análise dos Resultados

Foram testados diferentes tamanhos de mensagens: 8 bytes, 1 kilobyte (KB), 10 KB, 100 KB e 1 megabyte (MB). Cada configuração de tamanho foi submetida a um número fixo de repetições (1000 trocas consecutivas) para minimizar os efeitos de variações momentâneas de desempenho do sistema e tornar os resultados mais confiáveis.

Os resultados obtidos demonstram claramente a existência de dois regimes distintos na comunicação entre processos MPI: o regime dominado pela latência e o regime dominado pela largura de banda.

Para tamanhos de mensagens muito pequenos, como 8 bytes e 1 KB, observou-se que o tempo total de execução das trocas variou muito pouco, evidenciando que o fator preponderante foi a **latência** da comunicação — ou seja, o tempo fixo necessário para iniciar cada envio e recebimento de mensagem, independentemente do volume de dados transmitido.

Por outro lado, à medida que o tamanho da mensagem aumentou para valores como 100 KB e 1 MB, o tempo total cresceu de forma acentuada. Isso indica que, para mensagens maiores, o fator predominante passa a ser a **largura de banda**, que corresponde à capacidade do sistema de transmitir grandes volumes de dados em determinado intervalo de tempo.

O comportamento observado é característico de sistemas de comunicação paralela e confirma as expectativas teóricas. No gráfico apresentado, é possível identificar uma região inicial quase constante (regime de latência dominante) e uma região de crescimento quase linear (regime de largura de banda dominante).

6 Conclusão

A realização desta atividade permitiu compreender de forma prática como o tempo de comunicação entre processos em sistemas paralelos é influenciado tanto pela latência quanto pela largura de banda. A implementação do programa "ping-pong"utilizando a biblioteca MPI proporcionou a observação direta desses dois regimes, evidenciando como, para mensagens pequenas, a latência é o fator determinante, enquanto para mensagens maiores, a largura de banda se torna o elemento preponderante.

Além disso, a análise dos resultados demonstrou a importância de considerar o tamanho das mensagens na otimização do desempenho de aplicações paralelas, uma vez que diferentes tamanhos podem impactar significativamente o tempo total de comunicação. A utilização de técnicas de medição precisas, como o uso da função `MPI_Wtime` e a sincronização com `MPI_Barrier`, também se mostrou essencial para garantir a confiabilidade dos resultados.

Por fim, a atividade contribuiu para consolidar os conhecimentos sobre comunicação paralela com MPI, bem como sobre a importância da análise empírica no entendimento do comportamento de sistemas distribuídos.