Comunicação Avançada em MPI

Tópicos:
- Exemplo: Difusão de Dados em Anel
- Armazenamento de Mensagens em Trânsito
- Comunicação Sem Bloqueio em MPI
- Outros Modos de Comunicação em MPI

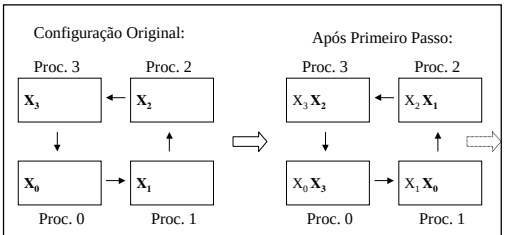
Referência: Pacheco,P.S. *Parallel Programming with MPI* Morgan Kaufmann, San Francisco, 1997.

Difusão de Dados em Anel

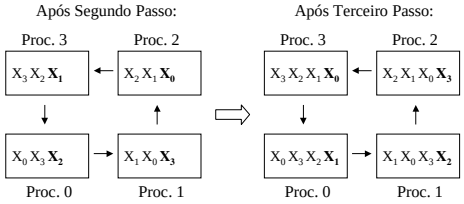
Problema de Difusão:
- Dado um array **X**, com um segmento armazenado em cada processador, obter em todos os processadores, o array total **Y** contendo todos os segmentos de **X**

Uma Possível Solução:
- Dispor os processadores num "anel"
- Enviar dados para o processador seguinte no anel; cada processador copia os dados recebidos
- Repetir o passo anterior P-2 vezes, sempre enviando adiante os novos dados recebidos (P-1 passos no total)

Difusão de Dados em Anel (cont.)



Difusão de Dados em Anel (cont.)



Difusão de Dados em Anel (cont.)

Algoritmo:
- Inicialização:
  - Copiar bloco local para a posição definitiva
  - Achar proc. vizinhos no anel (sucessor e predecessor)
- Procedimeto no Passo i:
  - Enviar bloco (my_rank - i + P) % P para o sucessor
  - Receber bloco (my_rank - i + P - 1) % P do predec.
  - Copiar bloco recebido: tamanho = *blocksize* = n / P
- Número Total de Passos: P - 1 ($0 \leq i < P-1$)

Difusão de Dados em Anel (cont.)

Implementação:

```
void Difusao_anel (
      float    x[]              /* in  */,
      int      blocksize        /* in  */,
      float    y[]              /* out */,
      MPI_Comm ring_comm  /* in  */) {
  int       i, p, my_rank, sucessor, predecessor, send_offset, recv_offset;
  MPI_Status  status;

  MPI_Comm_size(ring_comm, &p);
  MPI_Comm_rank(ring_comm,&my_rank);
  /* Copia x para o local correto em y */
  for (i = 0; i < blocksize; i++)   y[i + my_rank*blocksize] = x[i];
```

Difusão de Dados em Anel (cont.)

```
  sucessor = (my_rank + 1) % p;
  predecessor = (my_rank - 1 + p) % p;

  for (i = 0; i < p - 1; i++) {
     send_offset = ((my_rank - i + p) % p)*blocksize;
     recv_offset = ((my_rank - i - 1 + p) % p)*blocksize;
     MPI_Send(y + send_offset, blocksize, MPI_FLOAT,
        sucessor, 0, ring_comm);
     MPI_Recv(y + recv_offset, blocksize, MPI_FLOAT,
        predecessor, 0, ring_comm, &status);
  }
}
```


Armazenamento de Mensagens em Trânsito

No Exemplo Anterior (supondo P=2):

Proc.0	Proc.1
MPI_Send p/ 1	MPI_Send p/ 0
MPI_Recv de 1	MPI_Recv de 0

- Hipótese: Sist. Oper. armazena msgs em trânsito

→ O que irá ocorrer se não há buffers no sistema?
(padrão MPI não especifica que existam buffers)

- Se os dois processadores iniciam o prog. juntos:
  - Proc.0 não retorna de MPI_Send antes de 1 fazer Recv
  - Proc.1 não retorna de MPI_Send antes de 0 fazer Recv
  - *Deadlock* !

Armazenamento de Mensagens em Trânsito (cont.)

- Embora o programa do Exemplo esteja correto, pode haver deadlock se não houver buffers
  - Programas assim são ditos *inseguros*
  - Programas *seguros* não causam deadlock, mesmo que não haja buffers de sistema para mensagens em trânsito
- Como tornar um programa *seguro*?
  - Solução 1: Reorganizar as chamadas a *Send* / *Recv*

Proc.0 (par)	Proc.1 (ímpar)
MPI_Send p/ 1	MPI_Recv de 0
MPI_Recv de 1	MPI_Send p/ 0

Armazenamento de Mensagens em Trânsito (cont.)

- Solução 2:Utilizar MPI_Sendrecv( ) → não há deadlock
  - Equivalente a MPI_Send( ) seguido de MPI_Recv( )

```
int MPI_Sendrecv(
    void*          send_buf     /*  in  */,
    int            send_count   /*  in  */,
    MPI_Datatype   send_type    /*  in  */,
    int            destination  /*  in  */,
    int            send_tag     /*  in  */,
    void*          recv_buf     /*  out */,
    int            recv_count   /*  in  */,
    MPI_Datatype   recv_type    /*  in  */,
    int            source       /*  in  */,
    int            recv_tag     /*  in  */,
    MPI_Comm       recv_tag     /*  in  */,
    MPI_Status*    status       /*  out */ )
```


Comunicação Sem Bloqueio em MPI

Já vistas: Funções com bloqueio
- MPI_Send (buffer , ...):
  Só retorna quando buffer já pode ser reutilizado
- MPI_Recv (buffer , ...):
  Só retorna quando buffer já contém dados recebidos

Otimização:
- Funções sem bloqueio (retorno imediato)
- Idéia: Apenas comandar o início de *send* / *recv*
- Objetivo: Sobrepor comunicação e computação (pode ser útil se há um proc. de comunicação dedicado)

Comunicação Sem Bloqueio em MPI (cont.)

Implementação de comunicação sem bloqueio:
- MPI_Send (...) → MPI_Isend (...): Inicia *send*
- MPI_Recv (...) → MPI_Irecv (...): Inicia *recv*
- Espera pelo término da transmissão:
  - MPI_Wait( ): Bloqueia a execução esperando msg
- Verificação do término da transmissão:
  - MPI_Test( ): Não bloqueia a execução; retorna flag indicando se a transmissão já terminou
- Uso típico: MPI_Isend+MPI_Wait , MPI_Irecv+MPI_Wait

Obs: É possível misturar funções com/sem bloqueio, à vontade, num mesmo programa

Comunicação Sem Bloqueio em MPI (cont.)

Para o Exemplo Anterior:

```
MPI_Request send_request, recv_request;
. . .
send_offset = my_rank*blocksize;
recv_offset = ((my_rank - 1 + p) % p)*blocksize;
for (i = 0; i < p - 1; i++)                          {
    MPI_Isend(y + send_offset, blocksize, MPI_FLOAT,
        successor, 0, ring_comm, &send_request);      } Início da
    MPI_Irecv(y + recv_offset, blocksize, MPI_FLOAT,  } Comunic.
        predecessor, 0, ring_comm, &recv_request );
    send_offset = ((my_rank - i - 1 + p) % p)*blocksize;  } Comput.
    recv_offset = ((my_rank - i - 2 + p) % p)*blocksize;
    MPI_Wait(&send_request, &status);                 } Final da
    MPI_Wait(&recv_request, &status);       }         } Comunic.
```


Comunicação Sem Bloqueio em MPI (cont.)

Comparação de Desempenho: (tempos em ms)

	Intel-Paragon		IBM-SP2	
P	Com Bloq.	Sem Bloq.	Com Bloq.	Sem Bloq.
2	0.14	0.10	0.11	0.08
8	1.00	0.94	0.85	0.54
32	4.90	4.20	3.90	2.50

Obs: Ganho de desempenho seria maior se houvesse mais computação a ser feita

Outros Modos de Comunicação em MPI

Modos de Comunicação em MPI:

- Standard: MPI_Send( ), MPI_Isend( ), ...
- Synchronous: *Send* só termina depois que *Recv* inicia
  - MPI_Ssend( ) , MPI_Issend( ) , ...
- Ready: *Send* assume que *Recv* já ocorreu (sem buffer)
  - MPI_Rsend( ) , MPI_Irsend( ) , ...
- Buffered: Buffer auxiliar declarado pelo usuário
  - MPI_Bsend( ) , MPI_Ibsend( ) , ...
  - Tamanho do buffer auxiliar deve ser suficiente

Outros Modos de Comunicação em MPI (cont.)

Exemplo Anterior, com Modo *Buffered*:

```
char      buffer[MAX_BUF];     int    buffer_size = MAX_BUF;
. . .
MPI_Buffer_attach(buffer, buffer_size);
for (i = 0; i < p - 1; i++) {
    send_offset = ((my_rank - i + p) % p)*blocksize;
    recv_offset = ((my_rank - i - 1 + p) % p)*blocksize;
    MPI_Bsend(y + send_offset, blocksize, MPI_FLOAT,
       successor, 0, ring_comm);
    MPI_Recv(y + recv_offset, blocksize, MPI_FLOAT,
       predecessor, 0, ring_comm, &status);
}
MPI_Buffer_detach(&buffer, &buffer_size);
```