CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 29: Comunicação Unilateral em MPI

Celso L. Mendes, Stephan Stephany

LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Modos de Comunicação em MPI

- Modo de comunicação cooperativa:
  - Realizada entre um par de processos
  - Um processo envia dado, outro processo recebe dado
- Vantagens do modo cooperativo
  - Bem claro onde (que variáveis) o dado vai ser recebido
  - Bem claro quando o dado vai ser recebido: término do *recv*
- Desvantagens do modo cooperativo

a) Maior complexidade para escrever programas
   - A cada *recv* deve corresponder um *send*

b) Desempenho: envio/recebimento não são simultâneos na prática, podendo forçar esperas em algum dos lados, ou em ambos

Exemplo: como implemetar um contador compartilhado, que possa ser incrementado por qualquer um dos processadores executando o programa?

Comunicação Unilateral

- Modo de comunicação unilateral
  - *Remote Memory Access* (RMA)
  - Introduzido em MPI-2, expandido/aperfeiçoado em MPI-3
  - Uma única chamada indicando origem e destino do dado
    - Pode ser visto com um *send* combinado com um *recv*, mas com participação de um único processador (unilateral)
    - Desacopla movimentação de dados de sincronização
- Três passos necessários:
  1. Definição da *área de memória* a ser usada para RMA
     - Criação de uma "janela de memória" e objeto *MPI_Win*
  2. Especificação do dado a ser movido, e para onde mover
     - Funções *MPI_Put( )*, *MPI_Get( )*, *MPI_Accumulate( )* (MPI-2)
     - Outras funções expandidas a partir das acima (MPI-3)
  3. Especificação de como se pode determinar que a operação completou
     - Equivalente à conclusão de um *recv* no modo cooperativo

Funções do Modo Unilateral

1. Definição da janela de memória

*MPI_Win_create(void *base, MPI_Aint size, int disp_unit, MPI_Info info, MPI_Comm comm, MPI_Win *win)*

- Operação coletiva no comunicador *comm*
- Área exposta tem endereço *base*, contígua, com tamanho *size* em bytes
  - (se *size*=0, nenhuma área é exposta)
- Área exposta pode ser escrita remotamente (por *MPI_Put*) ou lida (por *MPI_Get*) pelos membros do comunicador
- Tamanho de cada ítem na janela é indicado por *disp_unit* bytes
- Potenciais otimizações de desempenho podem ser sugeridas por *info*

3. Verificação de término

MPI_Win_fence(int assert, MPI_Win win)

- Sincronização da janela *win*; parâmetro *assert* define tipo de sincronização
- Operação coletiva entre membros com acesso à janela

Funções do Modo Unilateral

2. Movimentação do(s) dado(s)

- *MPI_Put(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank, MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win)*
  - Dados: *origin_addr* (buffer local no proc. origem da oper.)
  - Alvo: Proc. *target_rank*, na área exposta pela janela
  - Transfere dados de *origin_addr* para *target_rank*
- *MPI_Get(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank, MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win)*
  - Transfere para *origin_addr* dados que estão em *target_rank*

Funções do Modo Unilateral

2. Movimentação do(s) dado(s) (cont.)

*MPI_Accumulate(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank, MPI_Aint target_disp, int target_count, MPI_Datatype target_type,* ***MPI_Op op****,MPI_Win win)*

- Dados: *origin_addr* (buffer local no proc. origem da oper.)
- Alvo: Proc. *target_rank*, na área exposta pela janela
- Transfere dados de *origin_addr* para *target_rank* aplicando o operador ***op*** antes de sobrescrever os dados
- Operação atômica
- ***op***: apenas os operadores pré-definidos são válidos

Comparação entre Modos

- Exemplo:

  Enviar *array[10]* do proc.0 para o proc.1
  - Modo cooperativo: *send/recv* (ou *Isend/Irecv*)
  - Modo unilateral: *put*
  - Apenas dois processadores estarão ativos neste exemplo
    - Novo comunicador criado, com procs {0,1}

Exemplo – Modo Cooperativo

```
MPI_Comm comm; MPI_Request request;
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split(MPI_COMM_WORLD,rank<=1,rank,&comm);
If (rank > 1) return;

If (rank==0) MPI_Isend(outbuf,n,MPI_INT,1,0,comm,&request);
else if (rank==1) MPI_Irecv(inbuf,n,MPI_INT,0,0,comm,&request);

MPI_Wait(&request,MPI_STATUS_IGNORE);
MPI_Comm_free(&comm);
```

Exemplo – Modo Unilateral

```
MPI_Comm comm; MPI_Request request; MPI_Win win;
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split(MPI_COMM_WORLD,rank<=1,rank,&comm);
If (rank > 1) return;

If (rank==0) MPI_Win_create(NULL,0,sizeof(int),MPI_INFO_NULL,comm,&win)
else if (rank==1) MPI_Win_create(inbuf,n*sizeof(int),sizeof(int),MPI_INFO_NULL,comm,&win);

MPI_Win_fence(0,win);
if (rank==0) MPI_Put(outbuf,n,MPI_INT,1,0,n,MPI_INT,win);

MPI_Win_fence(0,win);
MPI_Win_free(&win);
```

Exemplo – Modo Unilateral

```
MPI_Comm comm; MPI_Request request; MPI_Win win;
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split(MPI_COMM_WORLD,rank<=1,rank,&comm);
If (rank > 1) return;

If (rank==0) MPI_Win_create(NULL,0,sizeof(int),MPI_INFO_NULL,comm,&win)
else if (rank==1) MPI_Win_create(inbuf,n*sizeof(int),sizeof(int),MPI_INFO_NULL, comm, &win);

MPI_Win_fence(0,win);
if (rank==0) MPI_Put(outbuf,n,MPI_INT,1,0,n,MPI_INT,win);

MPI_Win_fence(0,win);
MPI_Win_free(&win);
```

Proc.1 expõe janela de memória

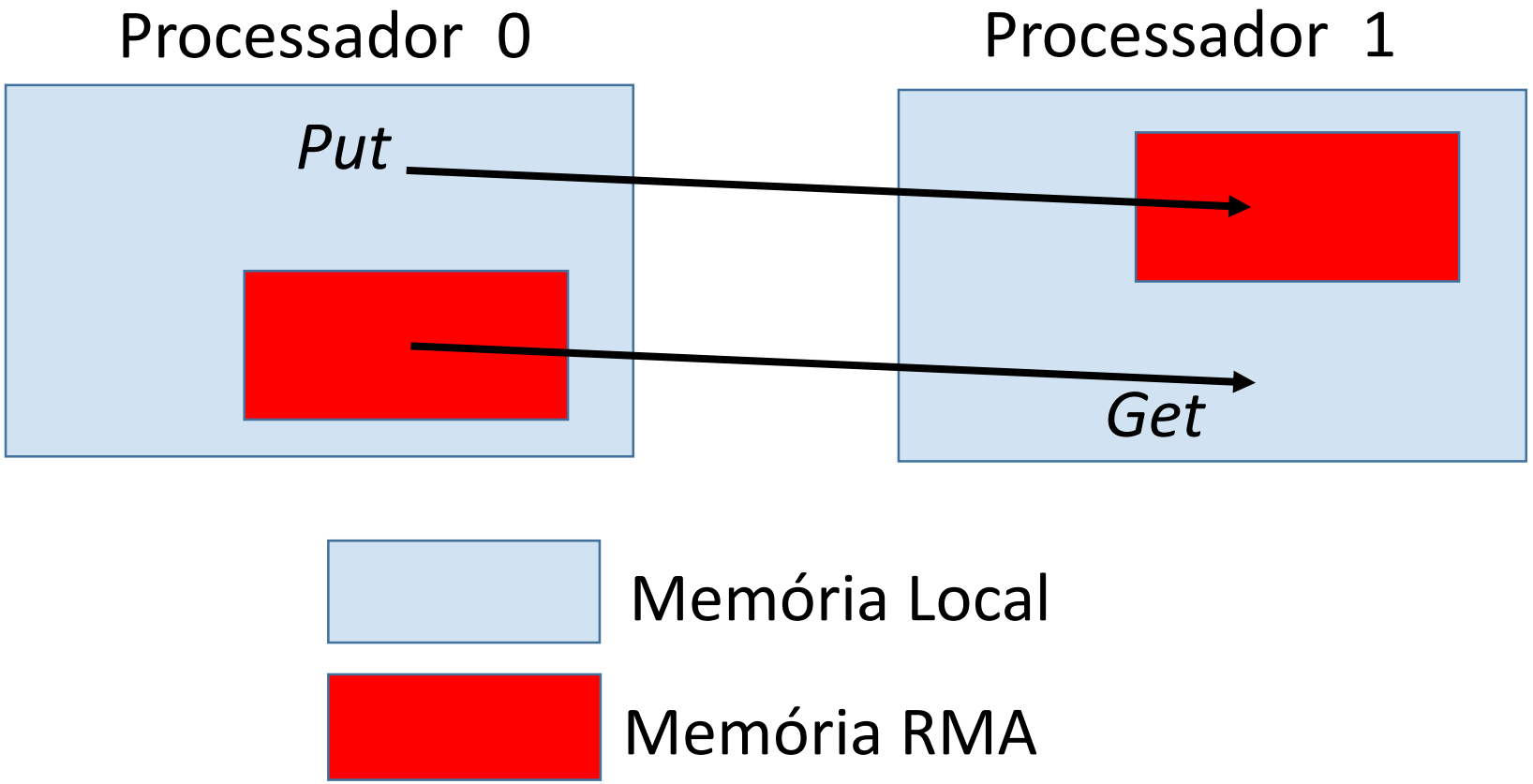
Proc.0 copia buffer local para a memória remota

Testa término

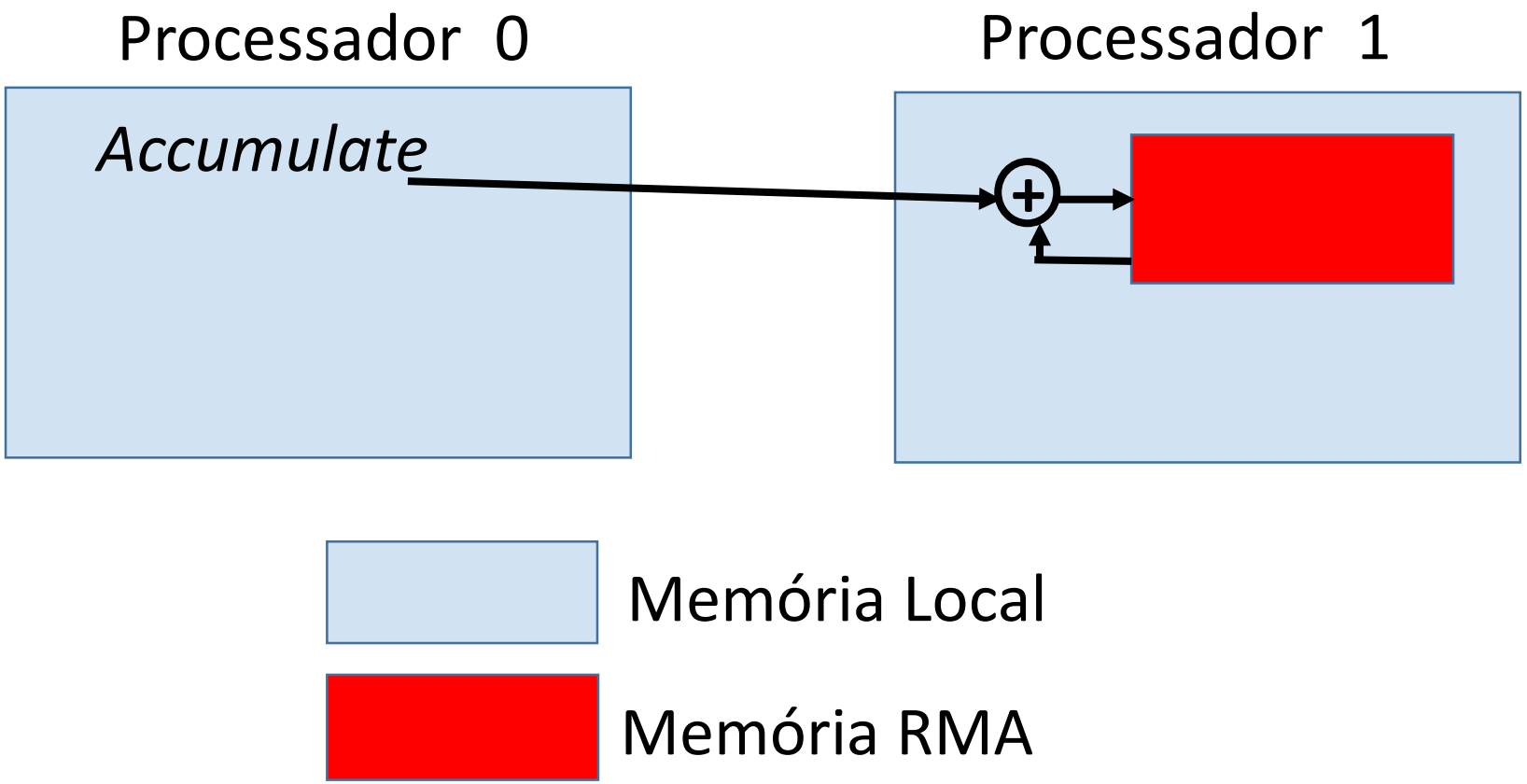
Exemplo - Comparações

- Envio de dados:
  - Modo cooperativo: Proc.0: *Isend* , Proc.1: *Irecv*
    - Quantidade de dados movidos é definida em *MPI_Isend*
    - Quantidade máxima é definida em *MPI_Irecv*
    - Área de origem dos dados no buffer local dada em *MPI_Isend*
      - Área de destino é definida em *MPI_Irecv*
  - Modo unilateral: Proc.0: *Put*
    - Quantidade de dados movidos é definida em *MPI_Put*
    - Quantidade máxima é dada pelo tamanho da janela
    - Área de origem dos dados no buffer local dada em *MPI_Put*
      - Área de destino é definida pela janela de memória
    - Programa poderia ser modificado com Proc.1: *Get*

MPI_Put e MPI_Get



MPI_Accumulate



Sincronização

- MPI_Win_fence(…):
  - Primitiva mais simple de sincronização
    → *Active Target Synchronization*
  - Operação coletiva, chamada por todos os membros com acesso à janela, inclusive o proc. *Target*
  - Excessão ao modo unilateral – participação de todos
  - Similar a MPI_Barrier( )
    - Porém barreiras não podem ser usadas em janelas
  - Cada chamada a *MPI_Win_fence* delimita uma *epoch*
    - Fronteira entre operações correntes numa janela

Sincronização (cont.)