Message Passing Interface (MPI): Novos Grupos e Comunicação Coletiva

Paulo Sérgio Lopes de Souza
pssouza@icmc.usp.br

Universidade de São Paulo / ICMC / SSC – São Carlos
Laboratório de Sistemas Distribuídos e Programação Concorrente

Novos Intracomunicadores

- Até agora trabalhamos com ***MPI_COMM_WORLD***
  - **Intracomunicador padrão** definido pelo MPI
    - Permite comunicação entre todos os processos iniciados com *mpirun*
    - Comunicações ponto-a-ponto ou coletivas (entre todos os processos)
  - Comunicações entre processos de um subgrupo pode ser desejável
    - Ex: processos responsáveis por linhas, colunas ou sub-blocos de uma matriz
  - **Novos intracomunicadores** devem especificar subgrupos de ***MPI_COMM_WORLD***
- Lembre que:
  - Intracomunicadores permitem comunicações entre processos dentro de um grupo
  - Intercomunicadores permitem comunicações entre processos com intracomunicadores diferentes
- Há diferentes maneiras de se criar novos intracomunicadores
- Funções explicadas nesta aula:

MPI_Comm_group	***MPI_Group_rank***	***MPI_Comm_split***
MPI_Group_incl	***MPI_Group_free***	
MPI_Group_excl	***MPI_Comm_create***	

Novos Intracomunicadores

- ***MPI_Comm_group()***
  - Determina o *handle* do grupo a partir de um comunicador

int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)

  - Com o *handle* de um grupo de processos é possível:
    - Selecionar processos para incluir em um subgrupo (*MPI_Group_incl*)
    - Criar um novo comunicador com um subgrupo (*MPI_Comm_create*)
    - Encontrar o *rank* do processo atual no subgrupo (*MPI_Group_rank*)

Novos Intracomunicadores

- ***MPI_Group_incl()***
  - Cria um novo grupo a partir de um já existente
  - Determina novos membros deste grupo **por inclusão**

int MPI_Group_incl(MPI_Group old_group, int count n, int members[ ], MPI_Group *new_group)

Novos Intracomunicadores

int MPI_Group_incl(MPI_Group old_group, int count n, int members[ ], MPI_Group *new_group)

```
#include "mpi.h"
MPI_Group group_world, odd_group, even_group;
int i, p, Neven, Nodd, members[8], ierr;

MPI_Comm_size(MPI_COMM_WORLD, &p);
MPI_Comm_group(MPI_COMM_WORLD, &group_world);

Neven = (p+1)/2;     /* processes of MPI_COMM_WORLD are divided */
Nodd = p - Neven;    /* into odd- and even-numbered groups */
for (i=0; i<Neven; i++) {    /* "members" determines members of
even_group */
  members[i] = 2*i;
};

MPI_Group_incl(group_world, Neven, members, &even_group);
```

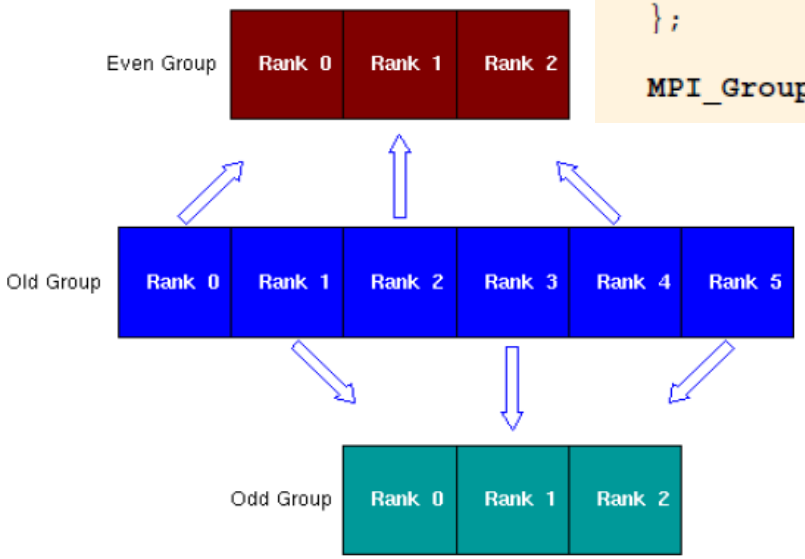


Figure 7.1. Graphical representation of MPI_GROUP_INCL example

Se *count* == 0, *new_group* = *MPI_GROUP_EMPTY*

Processos recebem *ranks* por suas posições em *members[ ]*

Grupos podem ter mesmos processos com diferentes *ranks*

PACS Training Group (2001)

Novos Intracomunicadores

- ***MPI_Group_excl()***
  - Cria um novo grupo a partir de um já existente
  - Determina novos membros deste grupo **por exclusão**

int MPI_Group_excl(MPI_Group old_group, int count, int nonmembers[ ], MPI_Group *new_group)

```
#include "mpi.h"
MPI_Group group_world, odd_group, even_group;
int i, p, Neven, Nodd, nonmembers[8], ierr;

MPI_Comm_size(MPI_COMM_WORLD, &p);
MPI_Comm_group(MPI_COMM_WORLD, &group_world);

Neven = (p+1)/2;     /* processes of MPI_COMM_WORLD are divided */
Nodd = p - Neven;    /* into odd- and even-numbered groups */
for (i=0; i<Neven; i++) {    /* "nonmembers" are even-numbered procs */
   nonmembers[i] = 2*i;
};

MPI_Group_excl(group_world, Neven, nonmembers, &odd_group);
```

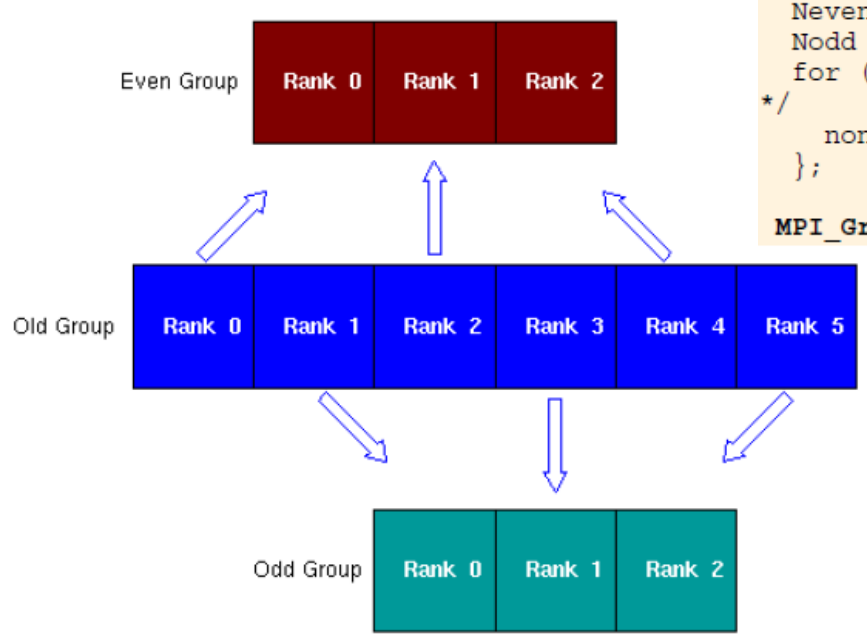


Figure 7.2. Graphical representation of MPI_Group_excl example

PACS Training Group (2001)

Novos Intracomunicadores

- ***MPI_Group_rank( )***
  - Retorna o número do *rank* do processo no grupo
  - *MPI_UNDEFINED* se não pertencer ao grupo

int MPI_Group_rank(MPI_Group group, int *rank)

- ***MPI_Group_free()***
  - Libera um grupo

int MPI_Group_free(MPI_Group *group)

- ***MPI_Comm_Create( )***
  - Cria um novo intracomunicador para um grupo

int MPI_Comm_create(MPI_Comm old_comm, MPI_Group group, MPI_Comm *new_comm)

Novos Intracomunicadores

- Exemplo de código para criação de novos intracomunicadores por grupos
  - São criados ***p*** processos com *mpirun*
  - O ***MPI_COMM_WORLD*** é dividido em dois usando ***MPI_Group_incl( )***
    - Dois grupos e comunicadores para processos pares e ímpares
    - Processo 0 vai pertencer aos dois subgrupos

Novos Intracomunicadores

- ***MPI_Comm_split ( )***
  - **Cria novos intracomunicadores** a partir de um já existente

int MPI_Comm_split(MPI_Comm old_comm, int color, int key, MPI_Comm *new_comm)

color agrupa processos em novos intracomunicadores (processos com o mesmo valor de ***color*** ficam juntos)
key controla o *rank* do processo chamador no novo intracomunicador
new_comm retorna o novo intracomunicador para o processo chamador

- Exemplo 06 processos dispostos logicamente como uma grade 03x02

```
/* logical 2D topology with nrow rows and mcol columns */
      irow = Iam/mcol;         /* logical row number */
      jcol = mod(Iam, mcol);   /* logical column number */
      comm2D = MPI_COMM_WORLD;

      MPI_Comm_split(comm2D, irow, jcol, row_comm);
      MPI_Comm_split(comm2D, jcol, irow, col_comm);
```

Iam	0	1	2	3	4	5
irow	0	0	1	1	2	2
jcol	0	1	0	1	0	1

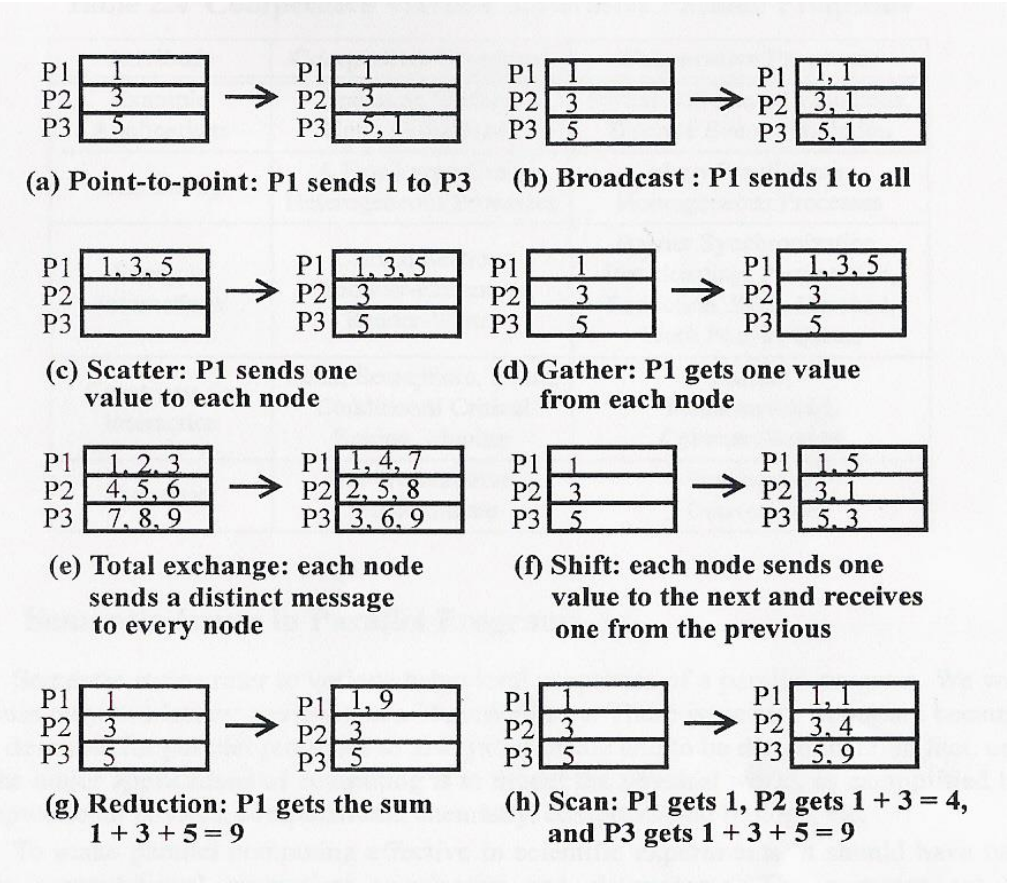
PACS Training Group (2001)

Novos Intracomunicadores

- Exemplo de código para criação de novos comunicadores por ***split***
  - São criados ***p*** processos com *mpirun*
  - O ***MPI_COMM_WORLD*** é dividido em dois usando ***MPI_Comm_split( )***
    - Dois grupos e comunicadores para processos pares e ímpares

Comunicação Coletiva entre Processos

- Comunicações coletivas
  - Otimizam a programação
  - Melhoram a portabilidade aos códigos
  - Tem potencial para melhores desempenhos
- Podem ser:
  - 1-to-1, 1-to-N, N-to-1, N-to-N



Hwang & Xu (1998)

Comunicação Coletiva no MPI

- Observação geral e importante para comunicação coletiva no MPI
  - **Primitivas coletivas fazem o envio e recebimento dos dados**
    - **Não se misturam com comunicação ponto-a-ponto**

- ***Barrier***
  - Usada para a sincronização dos processos
    - **Todos** os processos de um contexto de comunicação aguardam a chegada dos demais processos para poderem continuar
    - Comunicação n-to-n

int MPI_Barrier(MPI_Comm comm)

- ***Broadcast***
  - Envia a mesma mensagem para **todos** os demais processos do contexto de comunicação
  - Comunicação 1-to-n
  - **Responsável pelo envio e recebimento dos dados!**
    - **Não se mistura com comunicação ponto-a-ponto!**

int MPI_Bcast(void *buf, int count, MPI_Datatype datatype, int source, MPI_Comm comm)

Comunicação Coletiva no MPI

- ***Gather***
  - Coleta dados dispersos entre todos os processos em um único processo
  - Comunicação n-to-1

int MPI_Gather(void *sendbuf, int sendcount, MPI_Datatype senddatatype, void *recvbuf, int recvcount, MPI_Datatype recvdatatype, int target, MPI_Comm comm)

- Juntando *Gather* com *Broadcast*

int MPI_Allgather(void *sendbuf, int sendcount, MPI_Datatype senddatatype, void *recvbuf, int recvcount, MPI_Datatype recvdatatype, MPI_Comm comm)

Comunicação Coletiva no MPI

- ***Scatter***
  - Espalha informações de um processo sobre todos os processos de um domínio de comunicação
  - Comunicação 1-to-n

int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype senddatatype, void *recvbuf, int recvcount, MPI_Datatype recvdatatype, int source, MPI_Comm comm)

- A função abaixo envia quantidades diferentes de dados para cada processo destino

int MPI_Scatterv(void *sendbuf, int *sendcounts, int *displs, MPI_Datatype senddatatype, void *recvbuf, int recvcount, MPI_Datatype recvdatatype, int source, MPI_Comm comm)

Comunicação Coletiva no MPI

- ***All-to-All***
  - União das semânticas do ***gather*** e ***scatter***

int MPI_Alltoall(void *sendbuf, int sendcount, MPI_Datatype senddatatype, void *recvbuf, int recvcount, MPI_Datatype recvdatatype, MPI_Comm comm)

- ***Redução***
  - Associa uma computação à comunicação, por exemplo:
    - max, min, soma, prod, E/OR/XOR lógico e E/OR/XOR bit-a-bit
  - A redução considera todos os processos de um domínio de comunicação

int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int target, MPI_Comm comm)

Comunicação Coletiva no MPI

- **Prefix**
  - Faz uma redução com todos os processos de um domínio de comunicação
  - Distribui resultados parciais entre todos os processos de acordo com o *rank*
  - Operações possíveis são as mesmas das operações de redução

int MPI_Scan(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

Comunicação Coletiva no MPI

- Primitivas coletivas não bloqueantes
  - Análogas às bloqueantes explicadas
  - Há, por exemplo, uma barreira não bloqueante!
    - Processos retornam mesmo que os demais não tenham chegado na barreira
    - Programador responsável por tratar essa situação adequadamente
      - Determina o que pode ser feito enquanto há processos que ainda não chegaram na barreira

int MPI_Ibarrier(MPI_Comm comm, MPI_Request *request)

  - ***MPI_Test( )*** e ***MPI_Wait( )*** devem verificar o fim das operações coletivas não bloqueantes

- Exemplos de códigos C/MPI com primitivas coletivas

Referências

Rauber, T., & Rünger, G. (2013). *Parallel Programming*. Springer. Second edition. Capítulo 5.

Pacheco, P. (2011). *An introduction to parallel programming*. Elsevier. Capítulo 3.

Barlas, G. (2014). *Multicore and GPU Programming: An integrated approach*. Elsevier. Capítulo 5.

Grama, A., Kumar, V., Gupta, A., & Karypis, G. (2003). Introduction to parallel computing. Pearson Education. Capítulo 6.

PACS Training Group; Introduction to MPI. NCSA. University of Illinois. 2001.

Hwang, K.; Xu, Zhiwei; Scalable Paralle Computing: technology, architecture, programming. McGraw-Hill, 1998.

Apostila de Treinamento: Introdução ao MPI (Unicamp). https://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_MPI.pdf

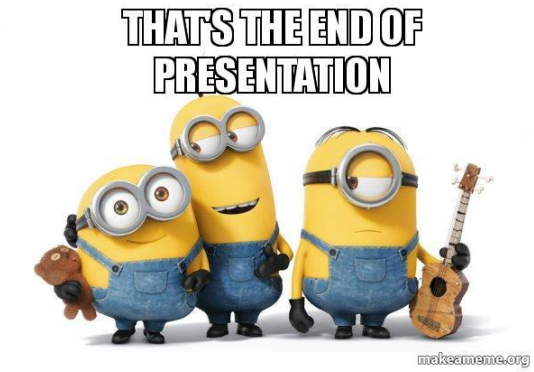
MacDonald, N; Minty, E.; Malard, J.; Harding, T.; Brown, S.; Antonioletti, M. Writing Message Passing Parallel Programs with MPI. 2020. Disponível em: https://www.researchgate.net/publication/239179288_Writing_Message_Passing_Parallel_Programs_with_MPI (último acesso 27/10/2020).

Fagg, Graham; Dongarra, Jack; Geist, Al. Heterogeneous MPI Application Interoperation and Process Management under PVMPI. 91-98. 1997. Disponível em: https://www.researchgate.net/figure/Inter-communicator-formed-inside-a-single-MPI-COMM-WORLD_fig1_221597084 (último acesso 27/10/2020).

Message Passing Interface (MPI): Novos Grupos e Comunicação Coletiva

Paulo Sérgio Lopes de Souza
pssouza@icmc.usp.br

Universidade de São Paulo / ICMC / SSC – São Carlos
Laboratório de Sistemas Distribuídos e Programação Concorrente



CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico
CAPES
FAPESP