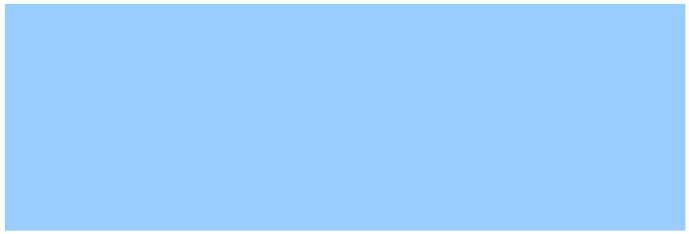
Processamento Paralelo
Aula 3 – Comunicação Coletiva

Adriano Wagner

19/01/2011

Processamento Paralelo

print*,t0
if (rank == 0) then



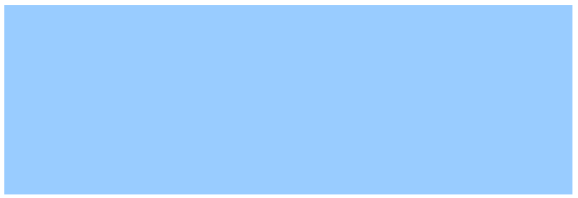
else

endif
print*,t1

~

➔ rank 0

print*,t0



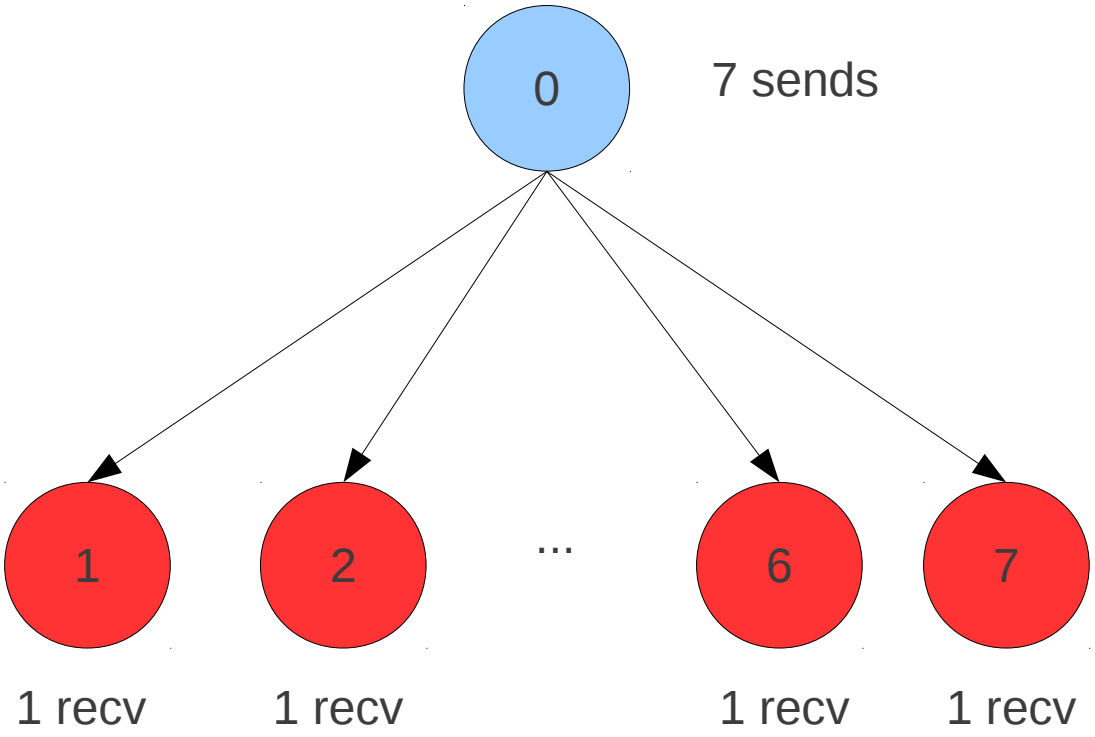
print*,t1

➔ demais ranks

print*,t0

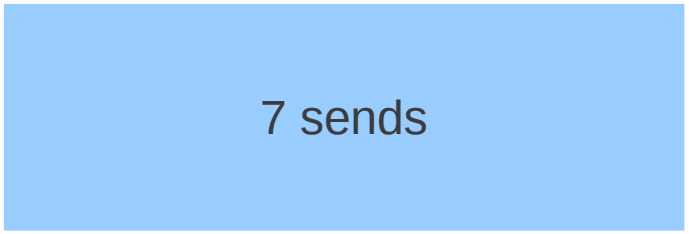
print*,t1

Processamento Paralelo



Processamento Paralelo

print*,t0
if (rank == 0) then



else

endif
print*,t1

Exemplo

- exemplo4
- Escravo envia o próprio rank para o mestre, que responde imediatamente
- Nó mestre deve responder a todos os demais nós antes de sair

Exemplo

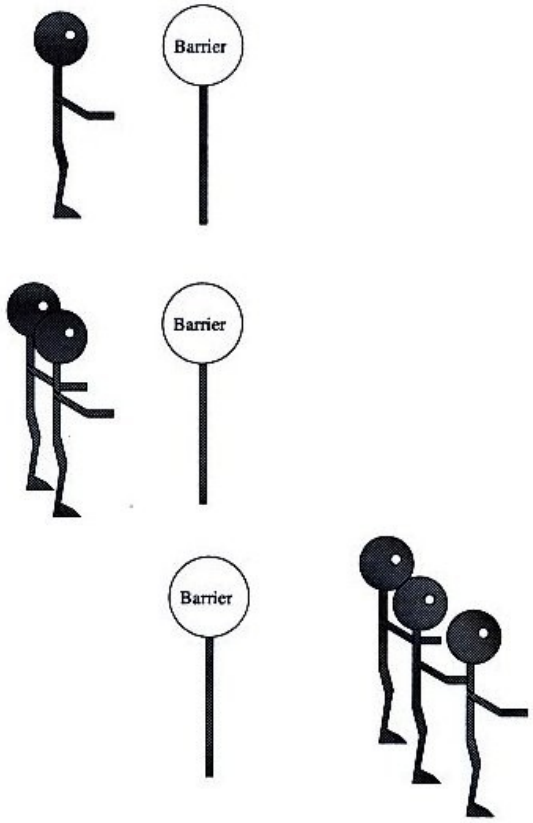
- exemplo5
- Cálculo do valor médio de um vetor
- Escravo envia o próprio rank para o mestre, que responde imediatamente, enviando uma tarefa
- Processo se repete até que não haja mais tarefas
- Resultados são reunidos de volta

MPI – Comunicação Coletiva

- Envolve todos os participantes de um grupo.
- Pode ter muitos receptores e/ou muitos remetentes.
- Exemplos:
  - Barreira
  - *Broadcast*
  - Operações de redução
- Chamada da função deve estar presente em todos os membros do grupo em questão.

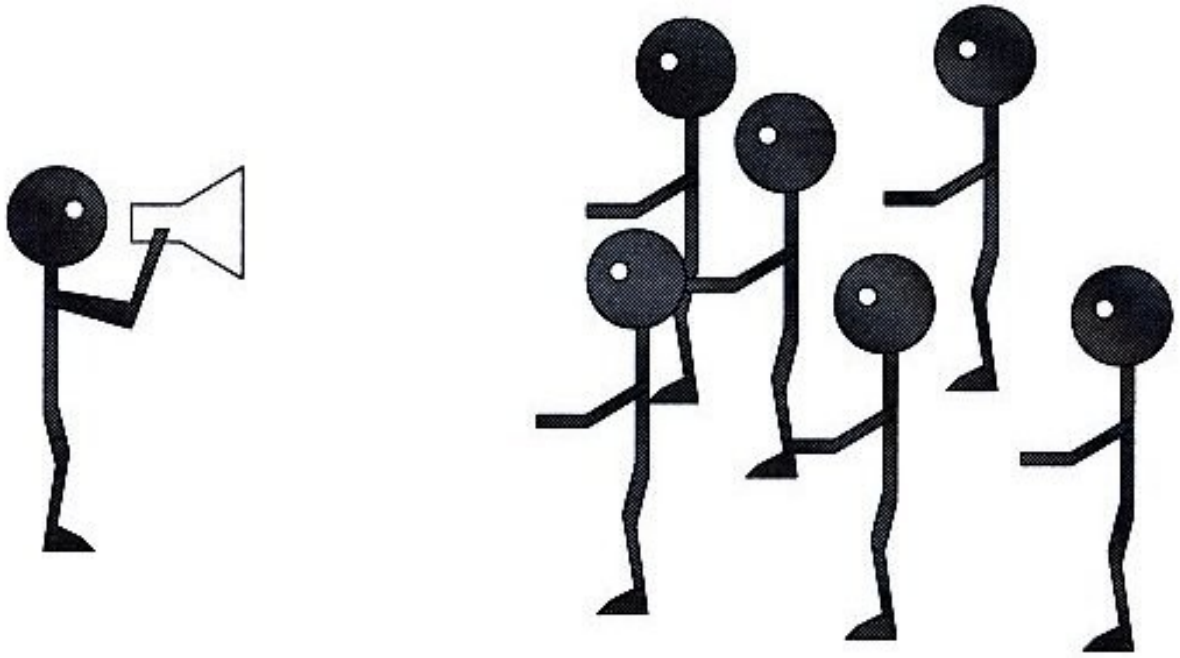
Barreira

- Sincroniza os participantes.
- Só permite a passagem quando todos os participantes alcançam a barreira.



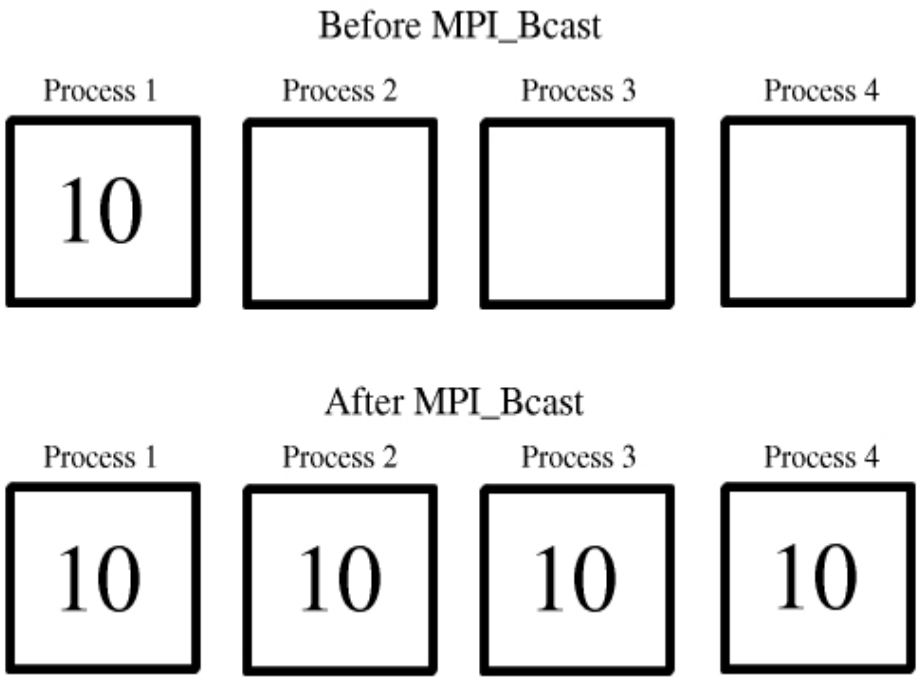
Broadcast

- Envia uma mensagem de um remetente para muitos destinos.
- Ex: E-mail marketing.



Comunicação coletiva - Funções

- MPI_Barrier(comm)
  - Cria uma barreira, sincronizando os processos.
- MPI_Bcast(buf, count, datatype, root, comm)
  - Envia uma mensagem de um destino (root) para todos os membros do grupo.



Exemplo

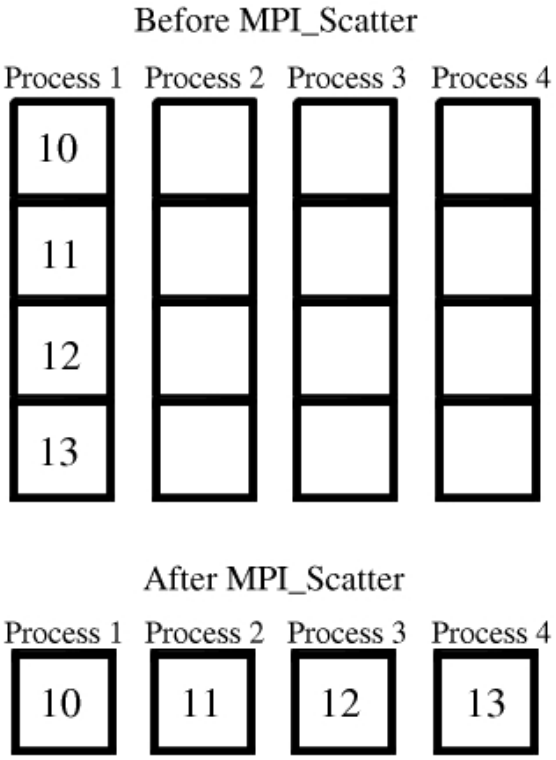
- exemplo6
- Nó mestre envia um valor para todos os membros através da função MPI_Bcast
- Ao fim do processo nós são sincronizados através da função MPI_Barrier

Exercício

- Alterar exemplo6
- Trabalhar com um vetor de tamanho 5
- Após a sincronização (MPI_Barrier), alterar o vetor apenas no rank 1
- Enviar novo vetor para os demais nós por broadcast

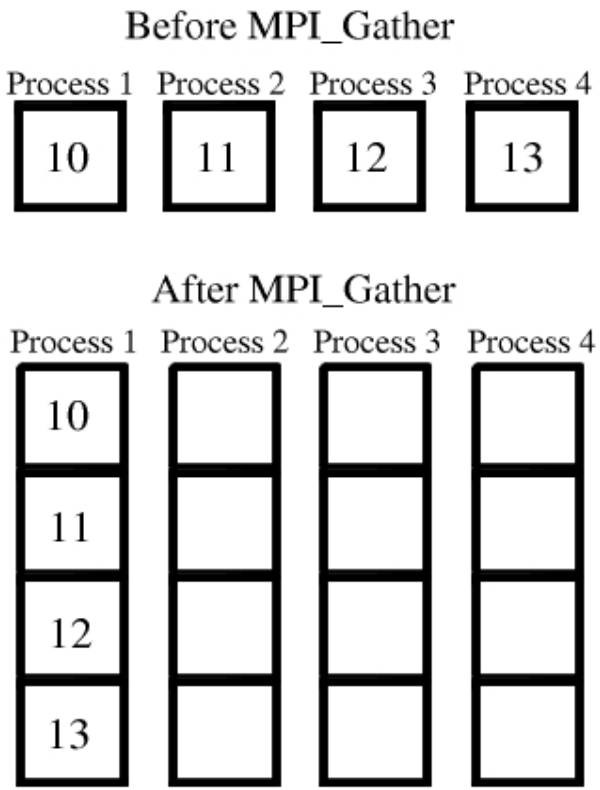
MPI – Comunicação Coletiva

- MPI_Scatter(send_buf, count, datatype, recv_buf, count, datatype, root, comm)
  - Divide um dado presente em um node (root) em partes iguais entre os demais processos.



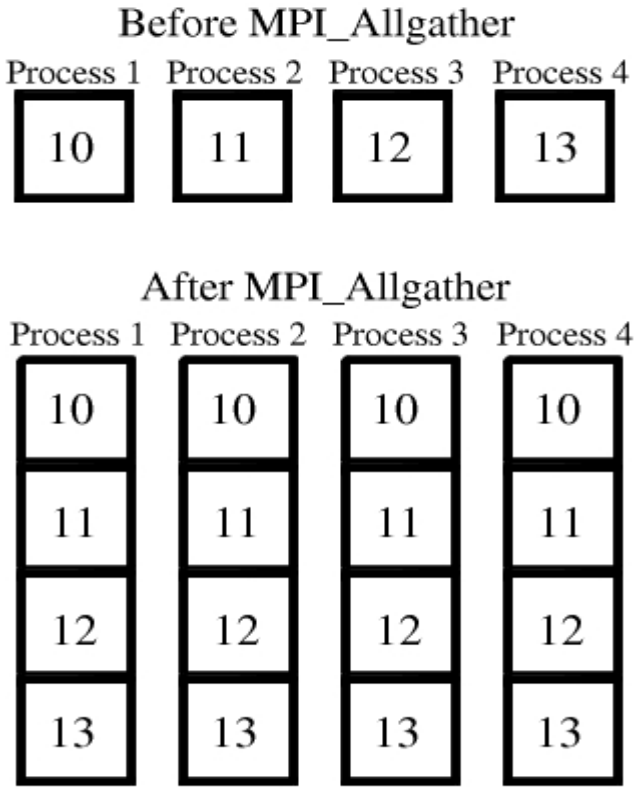
MPI – Comunicação Coletiva

- MPI_Gather(send_buf, count, datatype, recv_buf, count, datatype, root, comm)
  - Reúne os dados presentes nos processadores em um único node (root)



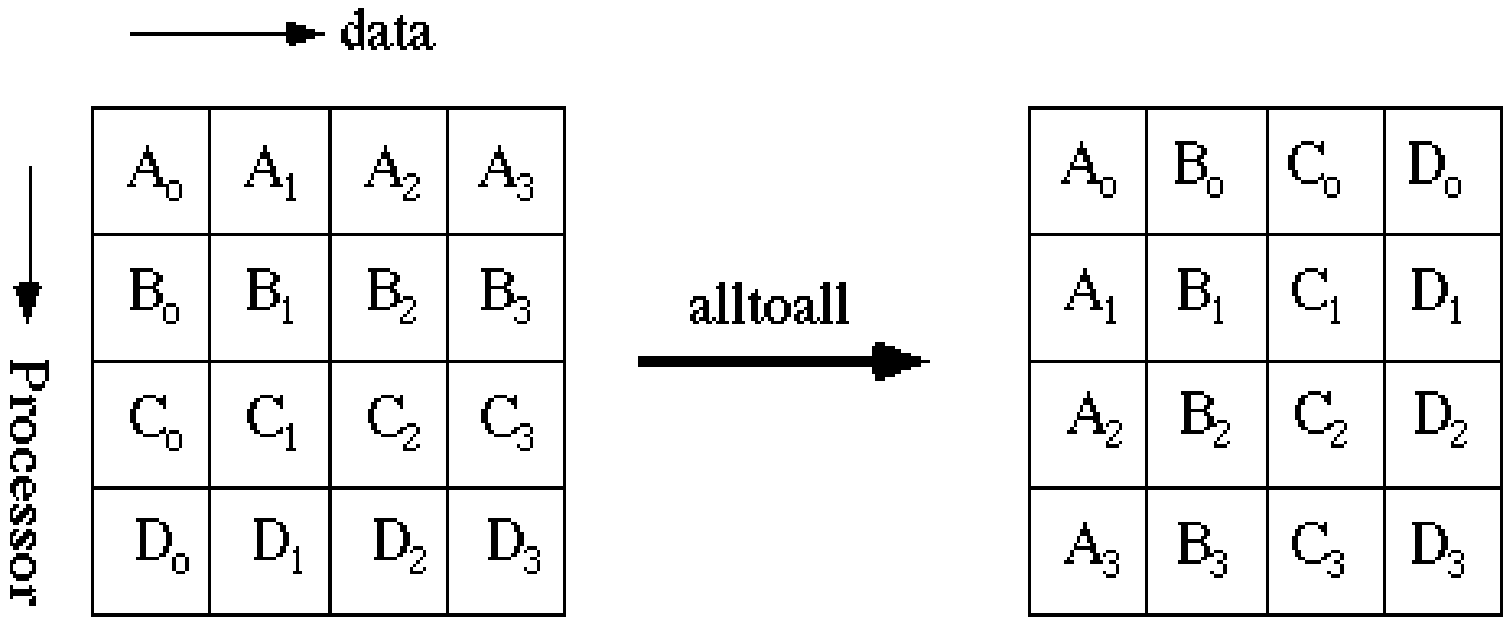
MPI – Comunicação Coletiva

- MPI_Allgather(send_buf, count, datatype, recv_buf, count, datatype, comm)
  - Reúne os dados presentes nos processadores em um único dado e distribui entre os membros do grupo.



MPI – Comunicação Coletiva

- MPI_Alltoall(send_buf, count, datatype, recv_buf, count, datatype, comm)
  - Envia uma parte distinta da mensagem de cada processo para todos os membros.



Exemplo

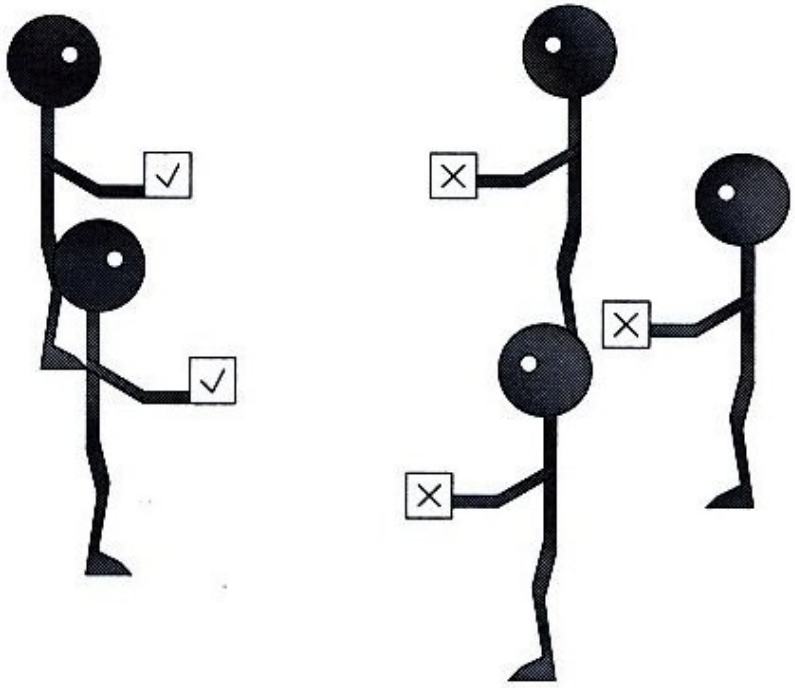
- exemplo7
- Vetor é gerado no nó mestre e dividido entre os demais processos
- Após processar os valores, o vetor é reunido no nó mestre

Exercício

- Alterar o exemplo5, para que use a função MPI_Scatter no envio das tarefas
- Usar vetor de tamanho 12

Operações de Redução

- Aplicar uma operação aos dados espalhados entre os participantes, reduzindo-o a um único dado.
- Soma, média, concatenação...
- Ex: Votação.



Operações de Redução

- MPI_Reduce(send_buf, recv_buf, count, datatype, op, root, comm)
  - Aplica uma operação (op), reunindo o resultado nó origem.
- Operações (op) podem ser: MPI_MAX, MPI_SUM, MPI_MIN, MPI_PROD...

Operações de Redução

- MPI_Allreduce(send_buf, recv_buf, count, datatype, op, comm)
  - O mesmo que MPI_Reduce, porém envia o resultado para todos os membros.

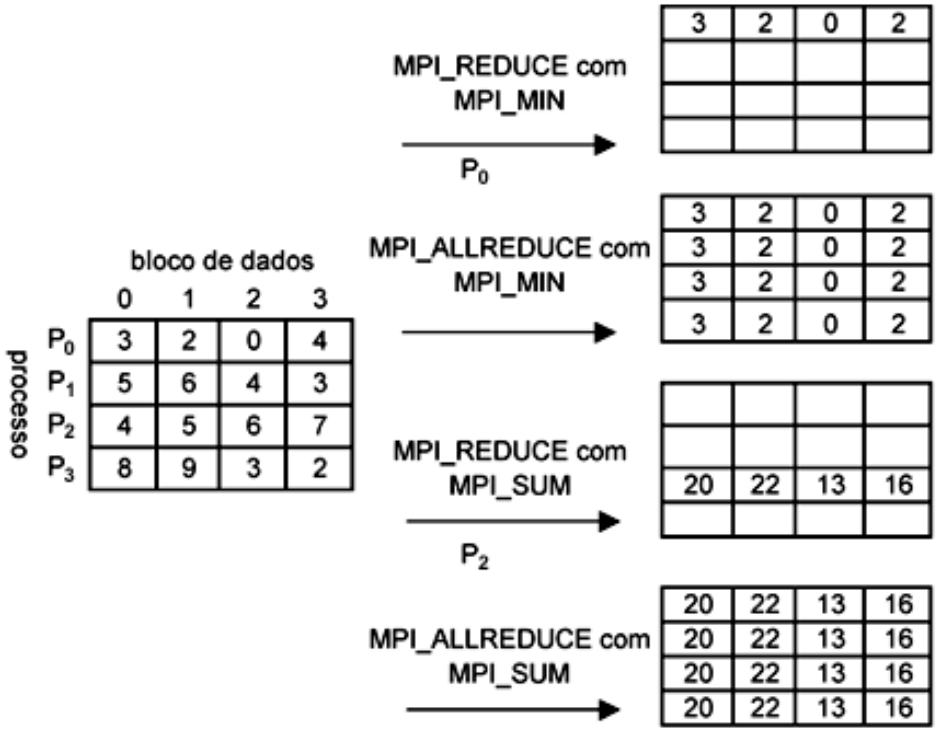


Figura 2 – Operações de Redução

Exemplo

- exemplo8
- Cada nó preenche um vetor de tamanho 3
- Todos os valores são somados e reunidos no nó mestre

Exercício

- Alterar último exercício para que a média seja calculada usando-se a função MPI_Reduce

Links úteis

- http://www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/fortran.html
- http://www-teaching.physics.ox.ac.uk/Unix+Prog/hargrove/tutorial_77/
- http://www.ead.cpdee.ufmg.br/cursos/C/
- https://computing.llnl.gov/tutorials/parallel_comp/
- https://computing.llnl.gov/tutorials/mpi/