MPI: Comunicação coletiva

- Comunicação entre diversos processo
  - 1 → N ou N→ 1
  - Rotinas
    - MPI_Barrier
    - MPI_Bcast
    - MPI_Gather
    - MPI_Scatter
    - MPI_Reduce

MPI: Comunicação coletiva

- Comunicação estruturada
  - Envolve + de 2 processos
- Definição de um processo 'raiz'
  - De onde dados serão copiados
    - MPI_Bcast
    - MPI_Scatter
  - Para onde dados serão reunidos
    - MPI_Reduce
    - MPI_Gather

MPI_Barrier

- Sincroniza processos
- Interrompe execução até que todos os processos o executem
- Protótipo

```
int MPI_Barrier(MPI_Comm communicator)
```

- Exemplo

```
MPI_Barrier(MPI_COMM_WORLD);
```

Exemplo de MPI_Barrier

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

int main(int argc, char **argv){
int i, meurank;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &meurank);

printf("Processo %d antes\n", meurank);
if (meurank == 0)
   for(i=0; i< 100000000; i++);

MPI_Barrier(MPI_COMM_WORLD);
printf("Processo %d depois\n", meurank);

MPI_Finalize();
return(0); }
```

Executar com e sem

Exemplo de MPI_Barrier: execução

```
$ mpirun -np 4 barreira
```

Sem MPI_Barrier

```
Processo 0 antes
Processo 1 antes
Processo 1 depois
Processo 2 antes
Processo 2 depois
Processo 3 antes
Processo 3 depois
Processo 0 depois
```

Com MPI_Barrier

```
Processo 0 antes
Processo 1 antes
Processo 2 antes
Processo 3 antes
Processo 0 depois
Processo 1 depois
Processo 2 depois
Processo 3 depois
```

MPI_Bcast

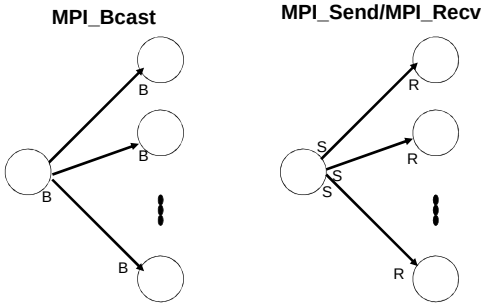
- Envia dado(s) a todos os processos
  - Utiliza o mesmo formato de dados
    - Referência (endereço)
    - Quantidade de elementos
    - Tipo de dados MPI
  - Não utiliza MPI_Send ou MPI_Recv
    - Todos os processos devem executar a mesma linha de comando

MPI_Bcast

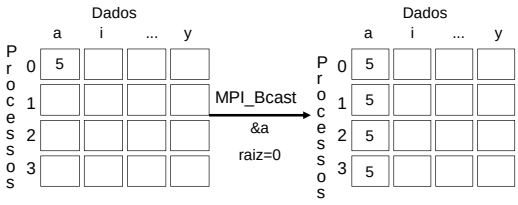
- Processos
  - Raiz: de onde os dados serão copiados
  - Demais: para onde os dados serão copiados
- Protótipo

```
int MPI_Bcast(
        void *buffer,
        int elementos,
        MPI_Datatype tipo_MPI,
        int raiz,
        MPI_Comm communicator)
```

MPI_Bcast: comparação com MPI_Send e MPI_Recv



MPI_Bcast: ilustração



MPI_Bcast: códigos equivalentes com MPI_Send/MPI_Recv

```
MPI_Bcast(&a, 1, MPI_INT, 0, MPI_COMM_WORLD);

- - - - - - - - - - - - - - - - - - - - - - - -

if (meurank == 0)
   for(i=1;i<procs;i++)
      MPI_Send(&a, 1, MPI_INT, i, tag,
               MPI_COMM_WORLD);
else
   MPI_Recv(&a, 1, MPI_INT, 0, tag,
            MPI_COMM_WORLD, &status);
```

Exemplo de MPI_Bcast

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char **argv){
int valor;
int meurank;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &meurank);
if (meurank == 0)
   valor = 20;
MPI_Bcast(&valor, 1, MPI_INT, 0, MPI_COMM_WORLD);
printf("Processo %d possui valor valendo %d\n",
        meurank, valor);
MPI_Finalize();
return(0);
}
```

Exemplo de MPI_Bcast: execução

```
$ mpirun -np 4 broad

Processo 0 possui valor valendo 20
Processo 1 possui valor valendo 20
Processo 2 possui valor valendo 20
Processo 3 possui valor valendo 20
```

MPI_Scatter

- Divide informações
- Envia mensagens coletivas
  - Processo = raiz
- Realiza também o recebimento
  - Processo != raiz
- Permite segmentação automática de mensagem
  - Utilização com estruturas de dados
    - Vetores
    - Matrizes

MPI_Scatter

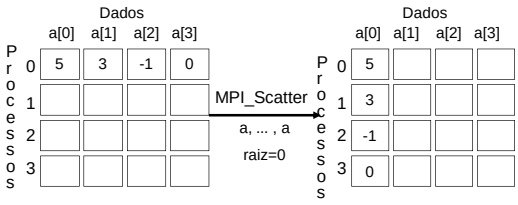
- Linha de comando deve ser a mesma em todos os processos
- Principais argumentos
  - Raiz
  - Buffer de envio
  - Buffer de recebimento

MPI_Scatter

- Protótipo

```
int MPI_Scatter (
        void *send_buffer,
        int send_elementos,
        MPI_Datatype send_tipo_MPI,
        void *receive_buffer,
        int receive_elementos,
        MPI_Datatype receive_tipo_MPI,
        int raiz,
        MPI_Comm communicator)
```

MPI_Scatter: ilustração



MPI_Scatter: códigos equivalentes com MPI_Send/MPI_Recv

```
processos = 4;

int a[12];
parte = 12/processos;
int b[parte];

if (meurank == 0)
   for(i=0; i<12; i++)
      a[i] = i;
```

MPI_Scatter: códigos equivalentes com MPI_Send/MPI_Recv

```
MPI_Scatter(a, parte, MPI_INT, b, parte,
            MPI_INT, 0, MPI_COMM_WORLD);
- - - - - - - - - - - - - - - - - - - - - - -
if (meurank == 0){
   ind = 0;
   for(i=1;i<procs;i++){
      MPI_Send(&a[ind], parte, MPI_INT, i, tag,
               MPI_COMM_WORLD);
      ind = ind + parte;
}
else
   MPI_Recv(&b[0], parte, MPI_INT, 0, tag,
            MPI_COMM_WORLD, &status);
```

Não faz cópia no processo 0

Exemplo de MPI_Scatter

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

#define TAM 4

int main(int argc, char **argv){
   int i;
   int mat[TAM][TAM]= {
       {  1,  2,  3,  4},
       {  5,  6,  7,  8},
       {  9, 10, 11, 12},
       { 13, 14, 15, 16}  };
   int vet[TAM];
   int meurank;                 /* continua … */
```

Exemplo de MPI_Scatter

```
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &meurank);

   MPI_Scatter(mat, TAM, MPI_INT,
               vet, TAM, MPI_INT,
               0, MPI_COMM_WORLD);

   printf("Processo %d [ ",meurank );
   for(i=0; i < TAM; i++)
      printf("%d ",vet[i]);
   printf("] Final do processo %d\n", meurank);
   fflush(stdout);

   MPI_Finalize();

   return(0);
}
```

Exemplo de MPI_Scatter: Execução

```
$ mpirun -np 4 divide

Processo 0 [ 1 2 3 4 ] Final do processo 0
Processo 1 [ 5 6 7 8 ] Final do processo 1
Processo 2 [ 9 10 11 12 ] Final do processo 2
Processo 3 [ 13 14 15 16 ] Final do processo 3
```

MPI_Gather

- Reúne informações
- Recebe mensagens coletivas
  - Processo = raiz
- Realiza também o envio
  - Processo != raiz
- Permite união de dados em estruturas
  - Vetores
  - Matrizes

MPI_Gather

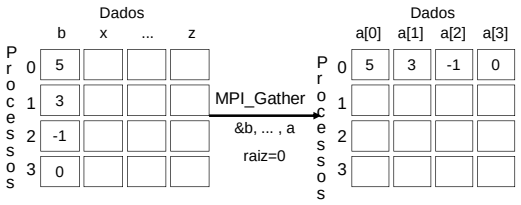
- Linha de comando deve ser a mesma em todos os processos
- Principais argumentos
  - Raiz
  - Buffer de envio
  - Buffer de recebimento

MPI_Gather

- Protótipo

```
int MPI_Gather (
        void *send_buffer,
        int send_elementos,
        MPI_Datatype send_tipo_MPI,
        void *receive_buffer,
        int receive_elementos,
        MPI_Datatype receive_tipo_MPI,
        int raiz,
        MPI_Comm communicator)
```

MPI_Gather: ilustração



MPI_Gather: códigos equivalentes com MPI_Send/MPI_Recv

```
processos = 4;
int meurank, vet[processos];

MPI_Gather(&meurank, 1, MPI_INT, vet, 1,
           MPI_INT, 0, MPI_COMM_WORLD);

- - - - - - - - - - - - - - - - - - - - - - - -

if (meurank == 0){
   for(i=1;i<procs;i++)
      MPI_Recv(&vet[i-1], 1, MPI_INT, i, tag,
               MPI_COMM_WORLD, &status);
else
   MPI_Send(&meurank, 1, MPI_INT, 0, tag,
            MPI_COMM_WORLD, &status);
```

Não coleta no processo 0

Exemplo de MPI_Gather

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

#define TAM 4

int main(int argc, char **argv){
   int vet[TAM];
   int i, meurank;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &meurank);


                                  /* continua … */
```

Exemplo de MPI_Gather

```
 MPI_Gather(&meurank, 1, MPI_INT, vet, 1,
            MPI_INT, 0, MPI_COMM_WORLD);

   if (meurank == 0){
      printf("Processo %d [ ",meurank);
      for(i=0; i < TAM; i++)
         printf("%d ",vet[i]);
      printf("] Final processo %d\n", meurank);
      fflush(stdout);

      MPI_Finalize();
   }

   return(0);
}
```

Outro exemplo de MPI_Gather

junta2.c

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

#define TAM 4

int main(int argc, char **argv){
   int vet[TAM*2];
   int ped[TAM/2];
   int i, meurank;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &meurank);

   ped[0] = meurank;
   ped[1] = meurank * 10;

                                  /* continua … */
```

Outro exemplo de MPI_Gather

junta2.c

```
 MPI_Gather(ped, TAM/2, MPI_INT, vet, TAM/2, MPI_INT,
            0, MPI_COMM_WORLD);

   if (meurank == 0){
      printf("Processo %d [ ",meurank);
      for(i=0; i < TAM*2; i++)
         printf("%d ",vet[i]);
      printf("]\nFinal processo %d\n", meurank);
      fflush(stdout);

      MPI_Finalize();
   }

   return(0);
}
```

Exemplo de MPI_Gather: Execução

```
$ mpirun -np 4 junta
Processo 0 [ 0 1 2 3 ] Final processo 0

- - - - - - - - - - - - - - - - - - - -

$ mpirun -np 4 junta2
Processo 0 [ 0 0 1 10 2 20 3 30 ]
Final processo 0
```

MPI_Reduce

- Realiza uma computação global
  - Todos os processos executam uma operação
  - Combina resultados parciais
    - rank != raiz
  - Devolve o resultado final para um processo específico
    - rank == raiz
- Diversos tipos de operação

MPI_Reduce

- Realiza automaticamente
  - Trocas de mensagens
  - Computação da operação
- Linha de comando deve ser a mesma em todos os processos
- Principais argumentos
  - Buffers de envio e recebimento
  - Raiz
  - Operação (padronizada pelo MPI)

MPI_Reduce: operações (principais)

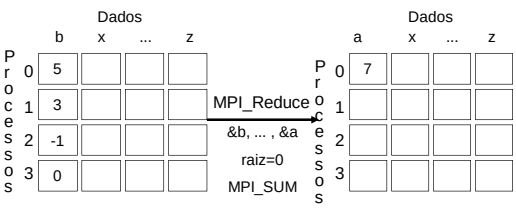
Operação	Significado
MPI_MAX	Valor máximo
MPI_MIN	Valor mínimo
MPI_SUM	Somatório dos valores
MPI_PROD	Produtório dos valores

MPI_Reduce

- Protótipo

```
int MPI_Reduce (
        void *send_buffer,
        void *receive_buffer,
        int elementos,
        MPI_Datatype tipo_MPI,
        MPI_Op operacao,
        int raiz,
        MPI_Comm communicator)
```

MPI_Reduce: ilustração



5 + 3 + (-1) + 0 = 7

MPI_Reduce: códigos equivalentes com MPI_Send/MPI_Recv

```
int soma;       /*processos = 4; */

MPI_Reduce(&meurank, &soma, 1, MPI_INT, MPI_SUM,
           0, MPI_COMM_WORLD);

- - - - - - - - - - - - - - - - - - - - - - - - -

if (meurank == 0){
   soma = meurank;
   for(i=1;i<procs;i++){
      MPI_Recv(&rec, 1, MPI_INT, i, tag,
               MPI_COMM_WORLD, &status);
      soma = soma + rec;
   }
else
   MPI_Send(&meurank, 1, MPI_INT, 0, tag,
            MPI_COMM_WORLD, &status);
```

Exemplo de MPI_Reduce

```
#include <stdio.h>
#include <mpi.h>
#include <sys/time.h>

int main(int argc, char **argv){
   int numero, maior;
   int meurank;
   struct timeval tempo;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &meurank);

   gettimeofday(&tempo, NULL);
   srandom(tempo.tv_usec);
   numero = random() % 50000;
                                  /* continua … */
```

Exemplo de MPI_Reduce

```
   printf("Processo %d gerou numero %d\n",
          meurank, numero);
   fflush(stdout);

   MPI_Reduce(&numero, &maior, 1, MPI_INT,
              MPI_MAX, 0, MPI_COMM_WORLD);

   printf("Processo %d encontrou maior
          valor%d\n", meurank, maior);
   fflush(stdout);

   MPI_Finalize();

   return(0);
}
```

Exemplo de MPI_Reduce: Execução

```
$ mpirun -np 4 maior

Processo 0 gerou numero 5672
Processo 0 encontrou maior valor 47976
Processo 2 gerou numero 7949
Processo 2 encontrou maior valor 1073786240
Processo 1 gerou numero 47976
Processo 1 encontrou maior valor 1073786240
Processo 3 gerou numero 22093
Processo 3 encontrou maior valor 1073786240
```