MPI.

- Ortega Ibarra Jaime Jesús.
- Martiñón Luna Jonathan José.
- José de Jesús Tapia López.

MPICC

- Este comando se puede usar para compilar y vincular programas MPI escritos en C. Proporciona las opciones y las bibliotecas especiales que se necesitan para compilar y vincular programas MPI.

```
(base) joiortega1@joiortega1-GL553VD:~/Escritorio/Concurrente$ mpicc holamundo.c
 -o holamundo
(base) joiortega1@joiortega1-GL553VD:~/Escritorio/Concurrente$
```

MPIRUN

Mpirun es un comando que controla varios aspectos de la ejecución del programa en Open MPI. Mpirun utiliza un entorno de tiempo de ejecución abierto para iniciar trabajos. Podemos administrar los recursos.

Único programa:

```
% mpirun [opciones ] [nombre-programa]
```

Múltiple programa:

```
% mpirun[ opciones ] [ nombre-programa ] :[ opciones2 ] [ nombre-programa2 ] ...
```

Hola Mundo

Creamos Hola mundo con C, utilizando MPI.

Obtendremos, el número del proceso, así como

Su nombre, esto imprimirá “Hola mundo”,

Desde el [Nombre del proceso] y su [Rango]

Seguido del [Total de procesos].

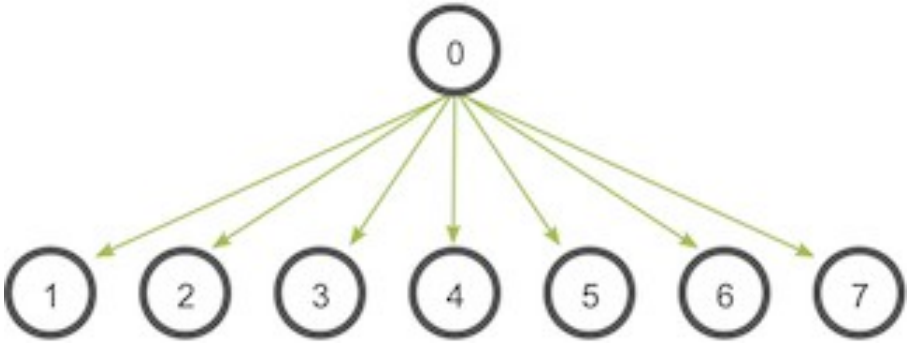
```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Inicializa MPI
    MPI_Init(NULL, NULL);

    // Numero de procesos
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Obtener el rango del proceso
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

     // Obtener el nombre del procesador
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

   // Imprime un mensaje de hello world
    printf("Hola mundo desde el proceso %s, rango %d de %d procesos\n",
           processor_name, world_rank, world_size);

    // Finaliza MPI
    MPI_Finalize();
}
```

Ejecución.

```
(base) joiortega1@joiortega1-GL553VD:~/Escritorio/Concurrente$ mpirun -np 4 ./ho
lamundo
Hola mundo desde el proceso joiortega1-GL553VD, rango 2 de 4 procesos
Hola mundo desde el proceso joiortega1-GL553VD, rango 3 de 4 procesos
Hola mundo desde el proceso joiortega1-GL553VD, rango 0 de 4 procesos
Hola mundo desde el proceso joiortega1-GL553VD, rango 1 de 4 procesos
(base) joiortega1@joiortega1-GL553VD:~/Escritorio/Concurrente$
```

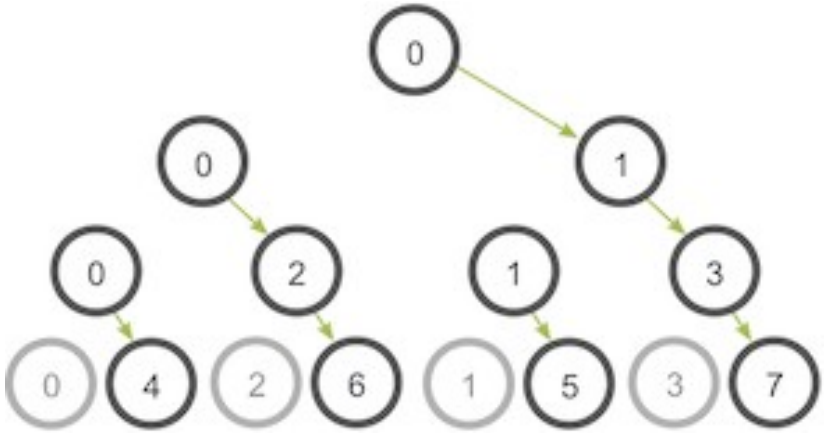
MPI Broadcast.

Una transmisión es una de las técnicas estándar de comunicación colectiva. Durante una transmisión, un proceso envía los mismos datos a todos los procesos en un comunicador. Uno de los principales usos de Broadcast es enviar la entrada del usuario a un programa paralelo o enviar parámetros de configuración a todos los procesos.



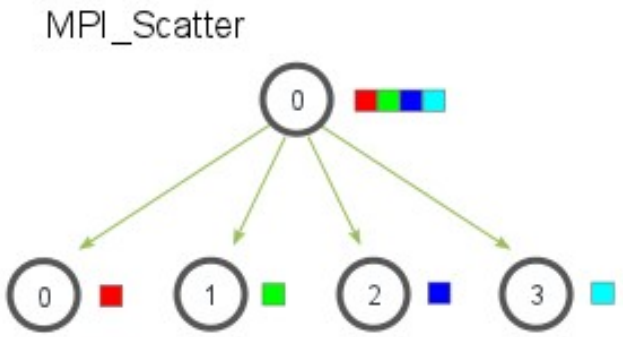
Envío y Recepción (MPI_send y MPI_recv)

Las llamadas de envío y recepción de MPI funcionan de la siguiente manera. En primer lugar, el proceso A decide que necesita un mensaje que se enviará al proceso B . El proceso A luego empaqueta todos sus datos necesarios en un búfer para el proceso B.



Scatter y Gather

MPI_Scatteres una rutina colectiva muy similar a MPI_Bcast. MPI_Scatter implica un proceso raíz designado que envía datos a todos los procesos en un comunicador. La diferencia principal entre MPI_Bcasty MPI_Scatteres pequeña pero importante. MPI_Bcastenvía la misma pieza de datos a todos los procesos mientras MPI_Scatter envía fragmentos de una matriz a diferentes procesos. Consulte la siguiente ilustración para obtener más aclaraciones.



Gather

Gather funciona como una inversa de Scatter, es regreso de los valores a la raíz.