Universidad Del Valle De Guatemala

Oscar Canek

Computación paralela


**Hoja de Trabajo 2: OpenMPI y Computación Distribuida**

Javier Mombiela 20067

Guatemala, 4 de septiembre del 2023

**mpiHello.c**

```c
main (int argc, char **argv)
{
  //----(1) INICIO DEL ENTORNO----//
  MPI_Init (&argc, &argv);

  //----(2) CAPTURA DE DATOS DEL COMUNICADOR----//
  int rank, num, i;
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &num);

  //----(3) DISTRIBUCION DEL TRABAJO----//
  printf ("Hello from process %i of %i\n", rank, num);

  //----(4) CLAUSURA DEL ENTORNO----//
  MPI_Finalize ();
  return 0;
}
```

```
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
cc mpiHello.c -o mpiHello
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
run -np 4 ./mpiHello
Hello from process 0 of 4
Hello from process 1 of 4
Hello from process 2 of 4
Hello from process 3 of 4
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$
```

Con el screenshot anterior podemos observar que estamos ejecutando el programa con 4 procesos (-np 4) y que cada proceso ha impreso su propio mensaje con su número de rank que va desde 0 hasta 3 en este caso. Esto nos demuestra que los procesos MPI se están ejecutando y comunicando correctamente en paralelo.

## Intercambio de Mensajes

```c
//----(3) DISTRIBUCION DEL TRABAJO----//
  if (rank == 0)
    {
        char mess[] = "Hello World";
        printf("%i sent %s\n", rank, mess);

        for (i = 1; i < num; i++)
        {
            MPI_Send(mess, strlen(mess) + 1, MPI_CHAR, i, MESSTAG, MPI_COMM_WORLD);
        }
    }
    else
    {
        char mess[MAXLEN];
        MPI_Status status;

        MPI_Recv(mess, MAXLEN, MPI_CHAR, 0, MESSTAG, MPI_COMM_WORLD, &status);
        printf("%i received %s\n", rank, mess);
    }
```

```
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
cc mpiHello4.c -o mpiHello
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
run -np 4 ./mpiHello
0 sent Hello World
1 received Hello World
2 received Hello World
3 received Hello World
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$
```

Como podemos observar, en el paso 3 del programa, en donde distribuimos el trabajo, ahora hay un if que indica que si el rank es 0, se envía un mensaje, si es algo más, se recibe el mensaje que envío el rank 0. En la captura de pantalla del output podemos ver que este comportamiento sí se cumple, ya que solo el primer proceso (el 0) envió el mensaje y todos los demás procesos lo recibieron. Este output sí demuestra la comunicación y coordinación exitosa entre los 4 procesos MPI.

## Mensajes con bloque y correspondencia

```c
if (rank %2 == 0)
    {
        char mess[] = "Hello World";
        printf("%i sent %s\n", rank, mess);

        MPI_Send(mess, strlen(mess) + 1, MPI_CHAR, rank+1, MESSTAG, MPI_COMM_WORLD);
    }
    else
    {
        char mess[MAXLEN];
        MPI_Status status;

        MPI_Recv(mess, MAXLEN, MPI_CHAR, 0, MESSTAG, MPI_COMM_WORLD, &status);
        printf("%i received %s\n", rank, mess);
    }
```

```
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
cc mpiHello5.c -o mpiHello
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
run -np 4 ./mpiHello
2 sent Hello World
0 sent Hello World
1 received Hello World
```

Como mencionas las instrucciones, podemos ver que uno de los procesos se queda en espera, lo que hace que el programa se pare por completo, por eso se realizan los siguientes cambios:

```c
//----(3) DISTRIBUCION DEL TRABAJO----//
  if (rank %2 == 0)
    {
        char mess[] = "Hello World";
        int dest_rank = (rank + 1) % num;
        printf("%d sent to %d: %s\n", rank, dest_rank, mess);

        MPI_Send(mess, strlen(mess) + 1, MPI_CHAR, dest_rank, MESSTAG, MPI_COMM_WORLD);
    }
    else
    {
        char mess[MAXLEN];
        MPI_Status status;
        int source_rank = (rank - 1 + num) % num;

        MPI_Recv(mess, MAXLEN, MPI_CHAR, source_rank, MESSTAG, MPI_COMM_WORLD, &status);
        printf("%d received from %d: %s\n", rank, source_rank, mess);
    }

  //Esperar que se sincronicen los procesos
  MPI_Barrier(MPI_COMM_WORLD);
```

```
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
cc mpiHello5.c -o mpiHello
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
run -np 4 ./mpiHello
0 sent to 1: Hello World
1 received from 0: Hello World
2 sent to 3: Hello World
3 received from 2: Hello World
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$
```

Para poder hacer que los procesos funcionaran de manera correcta, sin que ninguno se quedara esperando y por ende trabara todo el mensaje, se hicieron un par de cambios. El cambio significativo y el que en realidad hizo que todos los mensajes fueran recibidos de forma correcta, es el código que se agregó en la línea 21, que es el MPI_Barrrier. Esta es una función que crea una barrera de sincronización que asegura que todos los procesos esperen hasta que todos hayan alcanzado este punto antes de continuar. Esto garantiza que todas las comunicaciones se completen antes de que el programa finalice. Luego se hicieron un par de cambios pequeños en los prints, para poder especificar a qué proceso se enviaba el mensaje y de qué proceso se recibía el mensaje.

Como se puede observar en la última imagen, el output fue el esperado, ya que los procesos con un rank par son los que envían los mensajes, en este caso el proceso 0 y 2, y los procesos con rank impar son los que reciben, en este caso los procesos 1 y 3.

## Captura de correspondencia de mensajes con mi nombre y carnet

```
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
cc mpiHello5.c -o mpiHello
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$ mpi
run -np 4 ./mpiHello
0 sent to 1: Hello World: Javier Mombiela 20067
1 received from 0: Hello World: Javier Mombiela 20067
2 sent to 3: Hello World: Javier Mombiela 20067
3 received from 2: Hello World: Javier Mombiela 20067
javimombiela@javimombiela-VirtualBox:~/Documents/GitHub/Compu-Paralela/HDT2$
```