PRACTICA 3. ARQUITECTURA DE COMPUTADORES

DIEGO GONZALEZ SANZ Y JOSE IGNACIO MORENO MONTESINOS

INDICE

Ejercicio 1 3

Ejercicio 2 4

Ejercicio 3 4

Ejercicio 4 4

Ejercicio 1:

Parte 1:

Comunicación colectiva: es la que tiene de origen un emisor único y se dirige a un receptor masivo. Una de las características destacadas de este tipo de comunicación es que es unilateral. En estos casos el emisor emite un mensaje que va dirigido a un número elevado de individuos y/o procesos y estos lo reciben, pero sin que haya reciprocidad.

Las primitivas que existen en MPI son:

- MPI_Comm_Size: devuelve en "cuantos" el número de procesos del "grupo" indicado.
- MPI_Comm_rank: esta función devuelve en "yo" el identificador del proceso que le invoca dentro del "grupo" indicado.
- MPI_Finalize: sirve para informar al entorno MPI de que este proceso termina.
- MPI_Init: inicializa la ejecución del entorno MPI. Su efecto es arrancar los procesos indicados en la línea de comando.
- MPI_Send: el envío puede bloquear al proceso que hace el envío hasta que se haya recibido el mensaje.
- MPI_Recv: el proceso quedará bloqueado hasta recibir el mensaje indicado.
- MPI_Get_count: devuelve en "cuantos" el número de elementos del tipo "tipoDato" recibidos por la operación de recepción que devolvió el "estado" indicado como primer parámetro.
- MPI_Bcast: esta primitiva envía un mensaje desde el proceso "emisor" al resto de procesos que pertenezcan al "grupo" indicado, en principio MPI_Comm_WORLD.
- MPI_Reduce: realiza una operación de reducción global.

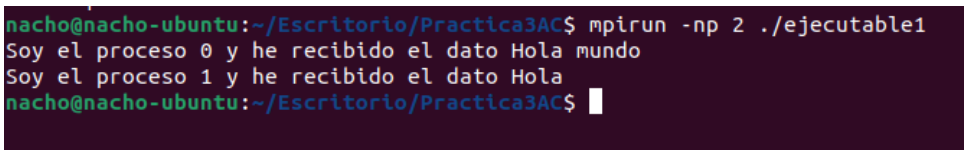
Ventajas y desventajas:

VENTAJAS	DESVENTAJAS
Conexión permanente	Estrés potencial permanente
Permiten la fusión de distintas tecnologías	Impiden la independencia de las herramientas
Interacción elevada	Subjetividad elevada
Automatización de determinados procesos	Industrialización de las relaciones humanas

Parte 2 (Programa):

En este ejercicio lo que nos pide es implementar un programa donde le nodo 0 inicializa el mensaje Hola mundo y lo envía al resto de los procesos del comunicador. Posteriormente deberá imprimirlo por pantalla.

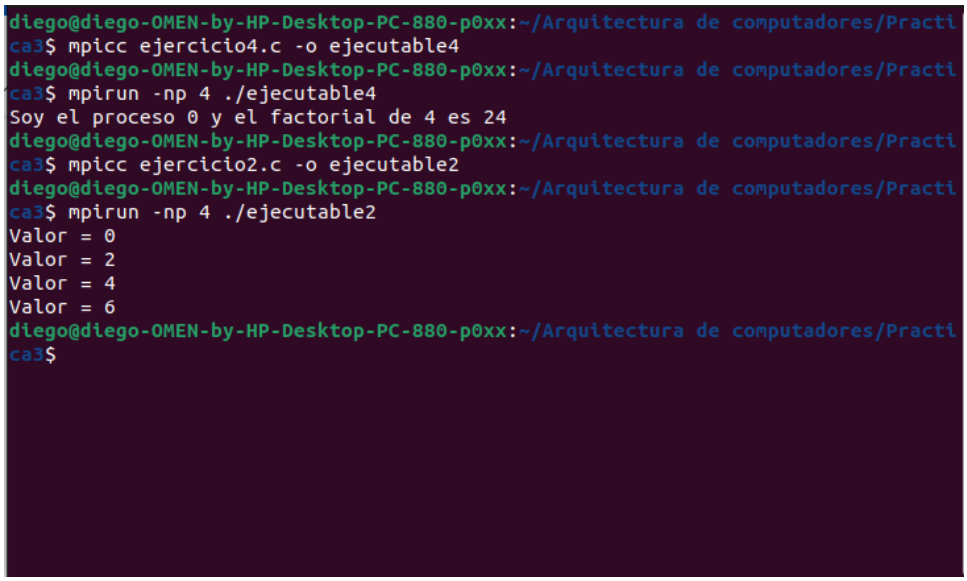
Primero creamos una variable array de mensaje para guardar todos los caracteres del mensaje y después lo creamos dentro del proceso 0. Luego con MPI_Bcast lo que hacemos es enviar el mensaje desde el proceso emisor al resto de procesos del mismo grupo. Luego lo mostramos por pantalla y terminamos los procesos.



Ejercicio 2:

En este ejercicio primero inicializamos la variable índice que es donde estará toda la información. Hacemos un bucle for para rellenar todo el array de los datos de las posiciones y después utilizamos la funcion MPI_Scatter que envía la información de la posición del rango, después hacemos la suma del valor que hemos cogido más el rango y con la funcion MPI_Gather la enviamos al nodo 0 todos los resultados. Y para finalizar cuando este en el proceso 0 que muestre por pantalla todos los resultados

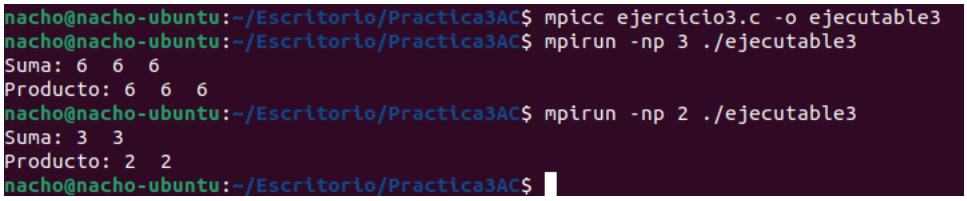
De forma similar a la función de broadcast, un nodo designado como root envía datos al resto de nodos del comunicador. La diferencia principal consiste en que los datos enviados a cada nodo son distintos. La función es la siguiente:



Ejercicio 3:

En este ejercicio se nos pedía un programa donde se inicializa un array de una dimensión, asignando a todos los elementos el valor de su rank+1. Después el proceso 0 ejecuta dos operaciones de reducción (suma y producto) sobre los arrays de todos los procesos.

En el código lo que hacemos es inicializar un array del tamaño de size, y luego otros dos que posteriormente asignaremos a cada operación. Después cogemos un for donde, como hemos dicho anteriormente, asignamos a todos los elementos el valor del rank+1. Con MPI_Reduce realizamos las operaciones para cada caso, en uno la suma y en otro el producto (MPI_SUM, MPI_PROD). Después, lo que hacemos es mostrar por pantalla la suma y la multiplicación, asignando un for para cada caso. Y para finalizar cerramos los procesos.



Ejercicio 4:

En este ejercicio estamos intentando sacar el factorial de un número. Primero tenemos la variable resultado que empieza valiendo 1, después tengo la variable rango que es equivalente a rango + 1

porque se empieza desde el proceso 0 si multiplicamos por 0 siempre nos dará 0 entonces controlamos eso. Por eso el programa lo que hará es iterar número de rango y lo va multiplicando a resultado y se va cambiando el resultado de la variable resultado hasta que llegue al factorial .