Solucionario: 1ra práctica calificada del curso de Programación Paralela

Alumno: SOTO BEJAR DIEGO ELIAS

1. **¿Qué es un proceso de una computadora?**

Un proceso de una computadora se refiere a la ejecución de diversas instrucciones por parte del procesador, en relación a las instrucciones de un programa. Ejemplo de proceso de una computadora: Abrir Excel.

1. **Explique a qué se refieren cuando hablamos de una comunicación punto a punto entre 2 procesos, proponer un ejemplo en código.**

Cuando hablamos de comunicación punto a punto entre 2 procesos hace referencia al momento el que 2 procesos funcionan como emisor y receptor de un mensaje. El proceso 0 envía data al proceso 1. Para ello el Proceso 0 utiliza el método MPI\_send() y el proceso 1 utiliza el método MPI\_recv().

Ejemplo:

**int main(int argc, char \*argv[])**

**{**

**int rank, contador;**

**MPI\_Status estado;**

**MPI\_Init(&argc, &argv);**

**MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);**

**//Envia y recibe mensajes**

**MPI\_Send(&rank //referencia al vector de elementos a enviar**

**,1 // tamaño del vector a enviar**

**,MPI\_INT // Tipo de dato que envias**

**,rank +1// pid del proceso destino**

**, 1 //etiqueta**

**,MPI\_COMM\_WORLD); //Comunicador por el que se manda**

**Ig(rank=1)**

**MPI\_Recv(&contador // Referencia al vector donde se almacenara lo recibido**

**,1 // tamaño del vector a recibir**

**,MPI\_INT // Tipo de dato que recibe**

**,rank-1 // pid del proceso origen de la que se recibe**

**,1// etiqueta**

**,MPI\_COMM\_WORLD // Comunicador por el que se recibe**

**,&estado); // estructura informativa del estado**

**cout<< "Soy el proceso "<<rank<<" y he recibido "<<contador<<endl;**

**MPI\_Finalize();**

**return 0;**

**}**

1. **Qué es una memoria Ram, Cache y virtual. ¿Cómo funcionan?**

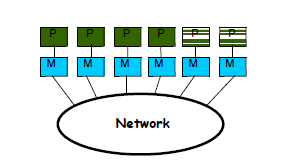
Memoria Ram: La memoria RAM es la memoria principal de un dispositivo donde se almacena programas y datos informativos. Esta memoria ayuda a tener varios programas abiertos al mismo momento.

Memoria Cache: Es un tipo de memoria volátil, pero mucho más rápida. La caché almacena instrucciones y datos a lo que el procesador debe acceder continuamente.

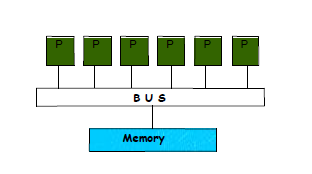
Memoria Virtual: Este tipo de memoria ayuda al SO a utilizar una mayor cantidad de memoria de la físicamente disponible, recurriendo a soluciones cuando se agota la memoria RAM.

1. **¿En qué consiste la programación en memoria distribuida y la programación en memoria compartida?**

Programación en memoria distribuida: Consiste en que cada procesador tiene su propia memoria local. Se utiliza el paso para el intercambio de datos



Programación en memoria compartida: Consiste en un único espacio de memoria, en el cual todos los procesadores tienen acceso a la memoria a través de una red de conexión



1. **Describa en 3 líneas como máximo e indicar los parámetros de los siguientes comandos**
2. MPI\_send()

Esta función nos permite enviar mensaje de un proceso a otro proceso, para esto se utiliza etiquetas en cada proceso.

MPI\_Send(

void\* data, // El mensaje

int count, // el número de elementos del mensaje

MPI\_Datatype datatype, //El tipo de dato del mensaje

int destination, //El proceso destino

int tag, //Etiqueta

MPI\_Comm communicator // el comunicador

)

1. MPI\_recv()

Esta función nos permite recibir mensaje que fue enviado por otro mensaje, para esto se utiliza etiqueta en ambo procesos.

MPI\_Recv(

void\* data, //Acá se almacena el mensaje

int count, // el número de elementos del mensaje

MPI\_Datatype datatype, //tipo de dato del mensaje

int source, //el proceso origen

int tag, //etiqueta

MPI\_Comm communicator, // comunicador

Status// estructura informativa del estado

)

1. MPI\_reduce()

Esta función nos permite realizer operación sobre los procesos, como por ejemplo una suma, promedio, producto, estas operaciones se harán al dato de cada proceso y el resultado se almacenará en el proceso root.

MPI\_Reduce(

void\* send\_data, //data sobre la cual se realiza la operación

void\* recv\_data, //Acá se almacena el resultado de la operación

int count, // el número de elementos

MPI\_Datatype datatype, // el tipo de dato

MPI\_Op op, //Acá se define la operación

int root, // el proceso donde enviará el resultado

MPI\_Comm communicator // el comunicador

)

1. MPI\_allreduce()

Este método es muy similar al reduce, la diferencia es que en el mpi\_reduce el resultado de la operación se envía al proceso raíz; en este caso el resultado se enviará a todos los procesos involucrados.

MPI\_Reduce(

void\* send\_data, //data sobre la cual se realiza la operación

void\* recv\_data, // donde se almacena el resultado de la operación

int count, // el número de elementos

MPI\_Datatype datatype, // el tipo de dato

MPI\_Op op, // la operación

MPI\_Comm communicator //el comunicador

)

1. **Suponga que COMM\_SZ = 8 y la cantidad de elementos es N = 16**

* Diseñe un programa que explique cómo **MPI\_Scatter** puede ser implementado usando comunicaciones basadas en árboles. Puede suponer que el origen del scatter es el proceso con rank 0.

El vector de 16 elementos es separado en 8 partes consecutivamente y en orden. Primero el proceso envía 8 elementos al proceso 0 y 4. Luego el proceso 0 envía 4 elementos al proceso 0 y 2, a su el proceso 4 envía 4 elementos al proceso 4 y 6. Luego el proceso 0 envía 2 elementos al proceso 0 y 1, el proceso 2 envía 2 elementos al proceso 2 y 3, el proceso 4 envía 2 elementos al proceso al proceso 4 y 5, y por último el proceso 6 envía 4 elementos al proceso 5 y 6. Con esto todos los procesos tienen 2 elementos.

* Hacer lo mismo para el **MPI\_Gather,** en este caso con el proceso 0 como destino.

Todos los procesos tienen 2 elementos. El proceso 0 y 1 envían sus elementos al proceso 0, el proceso 2 y 3 envían sus elementos al proceso 2, el proceso 4 y 5 envían sus elementos al proceso 4, el proceso 6 y 7 envían sus elementos al proceso 6. Luego el proceso 0 y 2 tienen 4 elementos cada uno, cada proceso envía sus elementos al proceso 0, el proceso 4 y 6 tienen 4 elementos cada uno, cada proceso envía sus elementos al proceso 4. Por último el proceso 0 y 4 tiene 8 elementos cada uno, cada proceso envía sus elementos al proceso 0. El proceso 0 tiene 16 elementos.