1ra. Práctica Calificada del curso de Programación Paralela

Alumno: Colan Torres Manuel Alejandro

Código: 14200013

Profesor: Herminio Paucar

Teoría

1. ¿Qué es una memoria RAM, Cache y Virtual? E indicar ¿Cómo funcionan?

Un proceso es un conjunto de instrucciones y datos que interactúan entre si con el fin de resolver una tarea en común.

2. ¿Qué es una memoria RAM, Cache y Virtual? E indicar ¿Cómo funcionan?

Se refiere a la comunicación exclusiva entre 2 proceso, en el cual uno envía información y el otro recibe ésta, mediante un identificador, una etiqueta y un comunicador.

```
int main(int argc, char *argv[])
{
    int rank, contador;
    MPI_Status estado;

    MPI_Init(&argc, &argv); // Inicializamos la comunicacion de los procesos
    MPI_Comm_rank(MPI_COMM_WORLD, &rank); // Obtenemos el valor de nuestro identificador
    if(rank==0){
    //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
            ,1 // pid del proceso destino
            ,0 //etiqueta
            ,MPI_COMM_WORLD); //Comunicador por el que se manda
    }
    else if(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
            ,0 // pid del proceso origen de la que se recibe
            ,0 // etiqueta
            ,MPI_COMM_WORLD // Comunicador por el que se recibe
            ,&estado); // estructura informativa del estado

        cout<< "Soy el proceso "<<rank-1<<" y he recibido "<<contador<<endl;

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

3. ¿Qué es una memoria RAM, Cache y Virtual? E indicar ¿Cómo funcionan?

Memoria virtual

Memoria Virtual es el uso combinado de memoria RAM en su computadora y espacio temporero en el disco duro. Cuando la memoria RAM es baja, la memoria virtual mueve datos desde la memoria RAM a un espacio llamado archivo de paginación. El movimiento de datos desde y hacia los archivos de paginación crea espacio en la memoria RAM para completar su tarea.

Si su computadora está falta de la memoria RAM necesaria para ejecutar una operación o programa, el sistema operativo utiliza la memoria virtual para compensar.

Memoria principal

Memoria primaria (MP), memoria principal, memoria central o memoria interna es la memoria de la computadora donde se almacenan temporalmente tanto los datos como los programas que la unidad central de procesamiento (CPU) está procesando o va a procesar en un determinado momento

Memoria caché

La memoria caché de un procesador es un tipo de memoria volátil (como la memoria RAM), pero muy rápida. Su función es almacenar instrucciones y datos a los que el procesador debe acceder continuamente. ¿Cuál es su finalidad? Pues que este tipo de datos sean de acceso instantáneo para el procesador, ya que se trata de información relevante y que debe estar a la mano de manera muy fluida. Los sistemas de hardware y software llamados caché, almacenan este tipo de datos de manera duplicada y por esta razón su acceso es tan veloz.

4. ¿En qué consiste la programación en Memoria Distribuida y la programación en Memoria Compartida?

En la memoria distribuida cada procesador tiene su propia memoria asignada localmente, la comunicación de datos es a través de buses de mensajería (paso de mensajes).

5. Describa en 3 líneas como máximo e indicar los parámetros de los siguientes comandos del MPI:

a) MPI_Send
Hace el envió de un mensaje a otro o muchos procesadores.

```
MPI_Send(
    void* data, //referencia al vector de elementos a enviar
    int count,  // tamaño del vector a enviar
    MPI_Datatype datatype, // Tipo de dato que envias
    int destination, // pid del proceso destino
    int tag, // etiqueta
    MPI_Comm communicator) // comunicador por el que se manda
```

b) MPI_Recv
Hace la recepción del mensaje enviado por otro proceso.

```
MPI_Recv(
    void* data, // Referencia al vector donde se almacenara lo recibido
    int count, // numero de elementos máximo a recibir
    MPI_Datatype datatype, // Tipo de dato que recibe
    int source, // pid del proceso origen de la que se recibe
    int tag, // etiqueta
    MPI_Comm communicator, // Comunicador por el que se recibe
    MPI_Status* status) // estructura informativa del estado
```

c) MPI_Reduce
Reduce un valor de un conjunto de procesos en un solo proceso raíz.

```
MPI_Reduce(
    void* send_data, // Dirección inicial del buffer en envío.
    void* recv_data, // Referencia al dato donde se reducira el valor de entrada
    int count, // Número de elementos que se va a enviar del buffer de envío (int).
    MPI_Datatype datatype, // Tipo de datos de los elementos del buffer de envío (por ejemplo MPI_INT).
    MPI_Op op, // Operación de reducción, constante definida por MPI.
    int root, // Rango del proceso raíz, el proceso receptor (int).
    MPI_Comm communicator // Comunicador por el que se realiza la comunicación
```

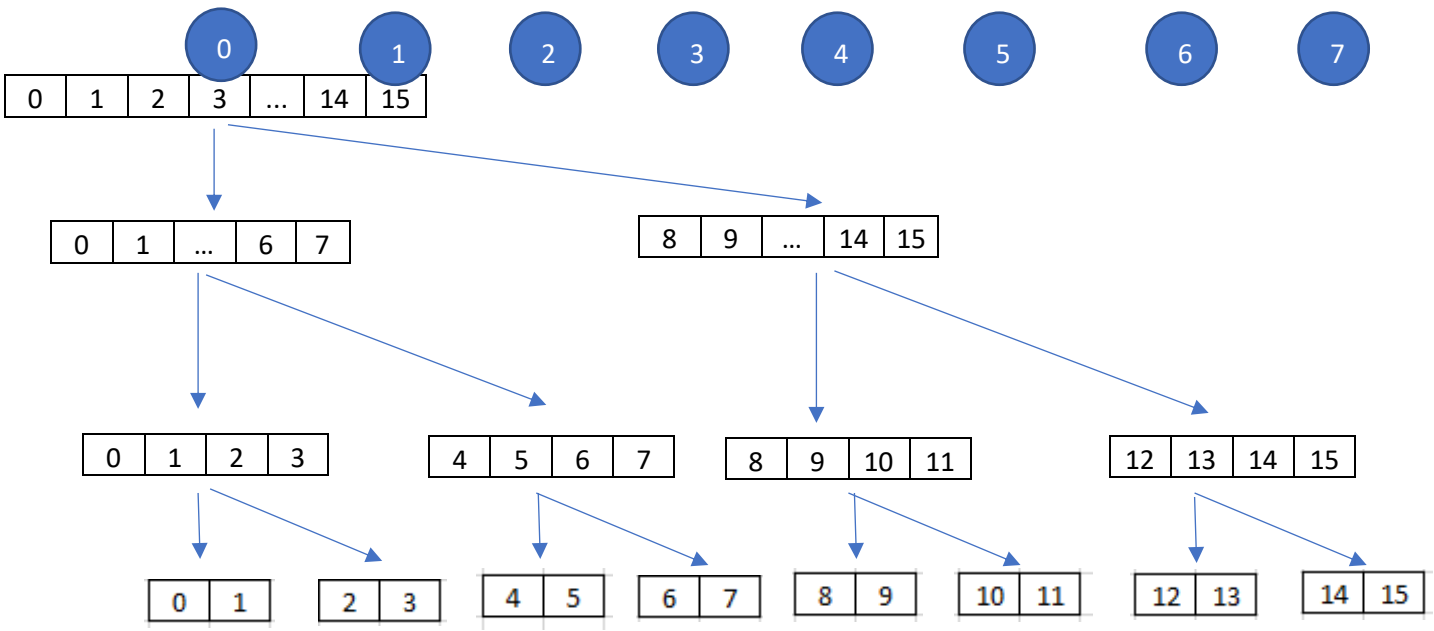
d) MPI_AllReduce
Reduce un valor de un conjunto de procesos y lo distribuye en todos los procesos.

```
MPI_AllReduce(
    void* send_data, // Dirección inicial del buffer en envío.
    void* recv_data, // Referencia al dato donde se reducira el valor de entrada
    int count, // Número de elementos que se va a enviar del buffer de envío (int).
    MPI_Datatype datatype, // Tipo de datos de los elementos del buffer de envío (por ejemplo MPI_INT).
    MPI_Op op, // Operación de reducción, constante definida por MPI.
    MPI_Comm communicator // Comunicador por el que se realiza la comunicación
```

Practica

8. Suponga que comm_sz=8 y la cantidad de elementos es n=16

a) Diseñe un diagrama 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.



b) Hacer lo mismo para el MPI_Gather, en este caso con el proceso 0 como destino.