SOLUCIONARIO DE LA PRIMERA PRACTICA CALIFICADA

1. **Explique con sus palabras ¿Qué es un proceso de una computadora?**

Es la ejecución de diversas instrucciones por parte del microprocesador, de acuerdo a lo que indica un programa. Se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados El sistema operativo de la computadora (ordenador) se encarga de gestionar los procesos.

2. Explique a que se refieren cuando hablamos de una comunicación punto a punto entre 2 procesos, proponer un ejemplo de código

Es la comunicación que hay entre pares de procesos. La selección de los mensajes es mediante ranking dentro de un grupo y tag (contexto). El ranking y tag están asociados a un comunicador. Ejemplo:

```
if (world_rank == 0) {
   number = -1;
   MPI_Send(&number, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
} else if (world_rank == 1) {
   MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,   MPI_STATUS_IGNORE);
   printf("Proceso  1 recibio el numero %d del proceso 0\n",  number);
}
```

3. ¿Qué es una memoria Ram (principal), Cache y Virtual? E indicar ¿Cómo funcionan?

MEMORIA RAM: Significa «Memoria de Acceso Aleatorio». La Memoria RAM en todos los casos, actúa como una memoria de corto plazo ya que esta se usa para guardar todos los procesos y datos generados para el funcionamiento de aplicaciones que se mantengan abiertas al mismo tiempo en el equipo.

Existen tres tipos de señales que el controlador de memoria RAM debe gestionar, señales de datos, señales de direccionamiento y señales de control. Estas señales circulan principalmente por los buses de datos y de direcciones y otras líneas de control

MEMORIA CACHE: Un caché es una colección de ubicaciones de memoria a las que se puede acceder en menos tiempo que otras ubicaciones de memoria. Cuando hablamos de cachés, generalmente nos referimos a un caché de CPU, que es una colección de ubicaciones de memoria a las que la CPU puede acceder más rápidamente de lo que puede acceder a la memoria principal.

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.

La memoria virtual opera en bloques de datos e instrucciones. Estos bloques se denominan comúnmente páginas, y dado que el acceso al almacenamiento secundario puede ser cientos de

miles de veces más lento que el acceso a la memoria principal, las páginas son relativamente grandes

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

MEMORIA DISTRIBUIDA: Cada núcleo tiene su propia memoria privada y los núcleos deben comunicarse explícitamente haciendo algo como enviar mensajes a través de la red

MEMORIA COMPARTIDA: Los núcleos pueden compartir el acceso a la memoria de la computadora, cada núcleo puede leer y escribir cada ubicación de memoria. Podemos coordinar los núcleos haciendo que examinen y actualicen las ubicaciones de memoria compartida

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

a) MPI_Send()

int MPI_Send (void∗ msg_buf_p, int msg_size, MPI_Datatype msg_type , int dest, int tag , MPI_Comm communicator);

Este argumento envía datos a otro proceso. msg_buf_p es el puntero que contiene el bloque de memoria donde se almacena el dato, msg_size es el tamaño del dato a enviar, msg_type es el tipo de dato, dest es el rango del proceso al que se enviara el mensaje y tag es la etiqueta.

b) MPI_Recv()

int MPI_Recv(void∗ msg_buf_p, int buf_size, MPI_Datatype buf_type, int source , int tag , MPI Comm communicator, MPI Status∗ status_p);

Este argumento recibe datos a otro proceso. msg_buf_p es el puntero que contiene el bloque de memoria donde se almacena el dato, bug_size es el tamaño maximo del dato, msg_type es el tipo de dato, source es el rango del proceso del que se recibirá el mensaje y tag es la etiqueta.

c) MPI_Reduce()

int MPI_Reduce(void∗ input_data p, void∗ output data_p, int count, MPI_Datatype datatype, MPI_Op operator, int dest_process , MPI_Comm comm);

Sirve para que cualquiera operación como suma, minimo, máximo, etc. se implemente en una sola funcion, input_data_p es el valor al que se aplicara la operación, output_data_p es donde se almacenara el resultado, operator es el operador predefinido del MPI.

d) MPI_Allreduce()

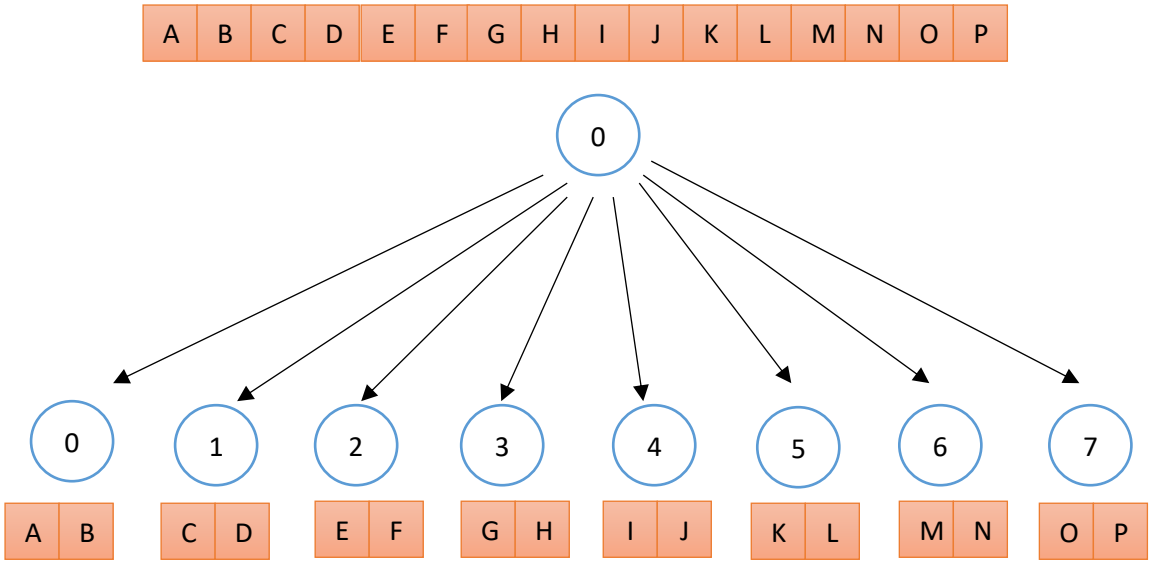
int MPI_Allreduce(void∗ input_data_p, void∗ output_data_p, int count, MPI_Datatype datatype, MPI_Op operato, MPI_Comm comm);

Se almacena el resultado en todos los procesos del comunicador. Tiene los mismos argumentos que el MPI_Reduce a excepción de que no hay proceso destino ya que todos los procesos deberían obtener el resultado

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 0

El MPI_Scatter lee la data completa que está en el proceso 0 pero solo envía los componentes necesarios a cada uno de los otros procesos. Si se tiene comm_sz procesos, esta función divide la data en comm_sz procesos, y cada proceso pasa a tener un vector local con tamaño igual a n/comm_sz. En este caso seria 16 / 8 = 2



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

El MPI_Gather puede recopilar todos los componentes del vector en el proceso 0 y luego el proceso 0 puede imprimir todos los componentes