1ra. Practica Calificada del curso de Programación Paralela

**Teoría**

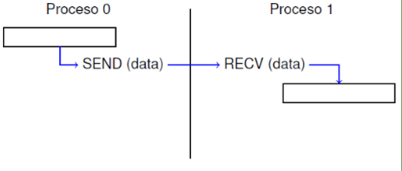
1. Explique con sus palabras ¿Qué es un proceso de una computadora?

Respuesta: Un **proceso** puede entenderse como una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y que utiliza un conjunto de recursos del sistema asociados.

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

Respuesta: 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.

Existe un emisor y un remitente especifico.



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

Respuesta:

Memoria RAM: La **memoria RAM** es la **memoria** principal de un dispositivo donde se almacena programas y datos informativos.

Su función principal es **almacenar datos e instrucciones** para que puedan ser accedidos por otros componentes básicos, de manera que evita que tengan que volver a pasar por el procesador o incluso por la tarjeta gráfica.

Cache:  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.

Virtual: En informática, la memoria virtual es una técnica de gestión de la memoria que se encarga de que el sistema operativo disponga, tanto para el software de usuario como para sí mismo, de mayor cantidad de memoria que esté disponible físicamente.

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

Respuesta:

Memoria Compartida. - Se puede conectar todos los procesadores directamente a la memoria principal trabajando con una sola memoria global.

Memoria Distribuida. - Los sistemas de memoria distribuida pueden ser de dos tipos básicos. El primer de ellos consta de un único computador con múltiples CPUs comunicadas por un bus de datos mientras que en el segundo se utilizan múltiples computadores, cada uno con su propio procesador, enlazados por una red de interconexión más o menos rápida.

1. Describe en 3 líneas como máximo e indicar los parámetros de los siguientes comandos del MPI:
   1. MPI\_Send (…): Realiza el envío de un mensaje de un proceso fuente a otro destino.

MPI\_Send(

void\* data, //buffer de datos

int count, //cantidad de elementos a enviar

MPI\_Datatype datatype, // tipo de datos a enviar

int destination, //El rango de proceso de envio

int tag, //etiqueta del mensaje

MPI\_Comm communicator) //Comunicador

* 1. MPI\_Recv (…): Rutina de recibimiento de un mensaje desde un proceso.

MPI\_Recv(

void\* data, // buffer de datos de recepcion

int count, //cantidad de elementos a recibir

MPI\_Datatype datatype, //tipo de datos a recibir

int source, //Rango de proceso de recepcion

int tag, // etiqueta del mensaje

MPI\_Comm communicator, //Comunicador

MPI\_Status\* status) //Estado del mensaje recibido

* 1. MPI\_Reduce (…): Reduce un valor de un grupo de procesos en un único proceso raíz.

MPI\_Reduce(

void\* send\_data, //matriz de elementos que cada proceso quiere reducir

void\* recv\_data, //

int count, // tamaño de la matriz

MPI\_Datatype datatype, //el tipo de dato

MPI\_Op op, // Operacion que se desea hacer

int root, //El rango de proceso

MPI\_Comm communicator) //Comunicador

* 1. MPI\_Allreduce (…): Reduce un valor de un grupo de procesos y lo redistribuye entre todos.

MPI\_Allreduce(

void\* send\_data, // matriz de elementos que cada proceso quiere reducir

void\* recv\_data,

int count, //tamaño de la matriz

MPI\_Datatype datatype, //El tipo de dato

MPI\_Op op, //Operación que se desea realizar

MPI\_Comm communicator) //Comubusquenicador

**Práctica**

1. Utilizando MPI, implemente un algoritmo que determine el número de veces que un elemento “**x”** aparezca en un vector A con “**n”** elementos enteros. Se puede asumir que su algoritmo comienza con los elementos ya distribuidos entre los “**p”** procesos (**n/p** para cada uno).
2. Desarrolle un algoritmo en MPI, utilizando procesadores para calcular n!.

**Respuesta:**

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[]){

int myRank, size;

int numero; //numero factorial

int inferior, superior;

int i;

double resultado\_local = 1.0;

double total;

/\* inicializar MPI \*/

MPI\_Init(&argc,&argv);

/\*obtener mi rango y el tamaÃ±o del comunicador \*/

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myRank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

// obtener la entrada. (solo si tengo rango 0)

if(myRank==0){

printf("Ingrese un numero:");

scanf("%d",&numero);

}

/\* ya que solo el proceso con rango tiene la entrada

\* debemos pasar a todos los demas procesos. \*/

MPI\_Bcast(&numero, // parametro de entrada/salida

1, // contador

MPI\_INT, // tipo de datos

0, // raiz

MPI\_COMM\_WORLD); // communicador

/\* calcular los limites superior e inferior

\* para cada proceso \*/

if(myRank==0){

inferior = 1;

}else

inferior = myRank \* (numero / size) + 1;

if(myRank==(size-1))

superior = numero;

else

superior = (myRank + 1) \* (numero / size);

/\* ahora que sabemos el superior e inferior,

\* haga la multiplicacion en nuestra area local \*/

for(i=inferior;i<=superior;i++){

resultado\_local = resultado\_local \* (double)i;

}

// combina todos los resultados locales multiplicandolos juntos

MPI\_Reduce(&resultado\_local, // operando

&total, // resultado

1, // contador

MPI\_DOUBLE, // tipo de datos

MPI\_PROD, // operador

0, // rango raiz

MPI\_COMM\_WORLD); // communicador

// dar la salida al usuario

if(myRank==0){

printf("El factorial de %d es %lf, y ha sido calculado usando %d procesos\n",numero,total,size);

}

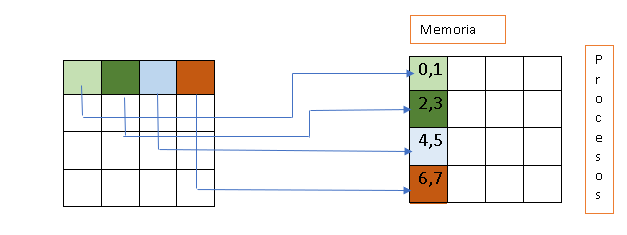
// finaliza el MPI

MPI\_Finalize();

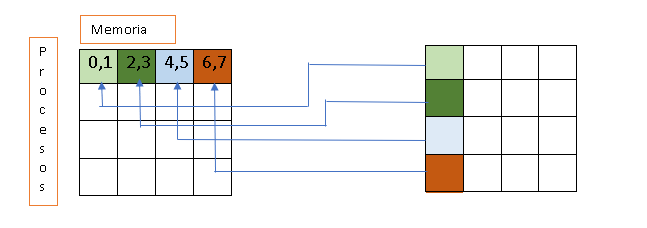
return 0;

}

1. Suponga que **comm\_sz=8** y la cantidad de elementos es **n=16**.
   1. Diseñe un diagrama que explique como **MPI\_Scatter** puede ser implementado usando comunicaciones basadas en árboles. Puede suponer que el origen del scatter es el proceso con Rank 0.

**Respuesta:** MPI\_Scatter establece una operación de distribución, en la cual un dato (arreglo de algún tipo de datos) se distribuye en diferentes procesos.

* 1. Hacer lo mismo para el **MPI\_Gather**, en este caso con el

**Respuesta:** MPI\_Gather establece una operación de recolección, en la cual los datos son recolectados en un solo proceso.