Teoría

1. ¿Qué es un proceso de una computadora?

Conjunto de tareas o instrucciones que un procesador ejecuta en una computador y que pertenecen a un determinado programa

1. Comunicación punto a punto

ES la comunicación entre dos procesos MPI diferentes. Un proceso realiza una operación de ***envío*** mientras que el otro realiza una lectura coincidente. MPI garantiza que cada mensaje llegará intacto sin errores.

MPI\_Send () y MPI\_Recv () son métodos de bloqueo comúnmente utilizados para enviar mensajes entre dos procesos MPI.

1. Explique:

Memoria RAM

RAM son las siglas de “Random Access Memory”. 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.

Memoria Caché

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.

Memoria Virtual

La memoria virtual es una técnica utilizada por los sistemas operativos para acceder a una mayor cantidad de memoria de la físicamente disponible, recurriendo a soluciones de almacenamiento alternativas cuando se agota la memoria RAM instalada.

1. Explique:

Programación en Memoria Distribuida

Cada procesador tiene su propia memoria local. Se utiliza paso de mensajes para intercambiar datos.

Programación en Memoria Compartida

Un único espacio de memoria. Todos los procesadores tienen acceso a la memoria a través de una red de interconexión:

* Bus
* Red de barras cruzadas
* Red multietapa

1. Describa e indique los parámetros de los siguientes comandos MPI:

MPI\_Send (

void\* msg buf p /in/, //data

int msg size /in/, //tamanio

MPI Datatype msg type /in/, //tipo de dato MPI

int dest /in/, //proceso destino

int tag /in/, //etiqueta

MPI Comm communicator /in/); //comunicador

MPI\_Recv (

void\* msg buf p /out/, //data

int buf size /in/, //tamanio

MPI Datatype buf type /in/, //tipo de dato MPI

int source /in/, //proceso origen o fuente

int tag /in/, //etiqueta

MPI Comm communicator /in/, //comunicador

MPI Status\* status p /out/); //estado

MPI\_Reduce (

void\* input data p /in/, //data de entrada

void\* output data p /out/, //data de salida

int count /in/, //tamanio

MPI Datatype datatype /in/, //tipo de dato MPI

MPI Op operator /in/, //operador MPI

int dest process /in/, //proceso destino

MPI Comm comm /in/); //comunicador

MPI\_Allreduce (

void\* input data p /in/, //data de entrada

void\* output data p /out/, //data de salida

int count /in/, //tamanio

MPI Datatype datatype /in/, //tipo de dato MPI

MPI Op operator /in/, //operador MPI

MPI Comm comm /in /); //comunicador