UTE EC

EDUCACIÓN EN LÍNEA

Computación Paralela y Distribuida

MPI

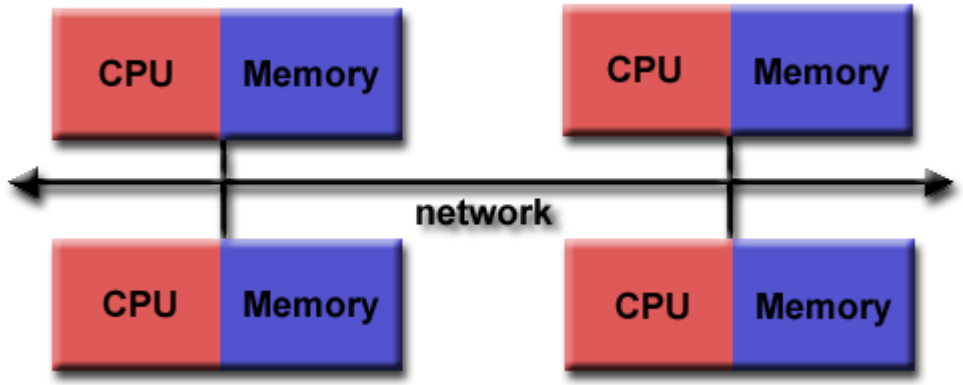
Grupo N°2: Carolyn Quilca,Dario Rodríguez

MPI (Message Passing Interface)

¿Qué es?

MPI que significa Interfaz de paso de mensajes es un estándar de paso de mensajes y portátil diseñado por un grupo de investigadores de la academia y la industria para funcionar en una amplia variedad de arquitecturas de computación paralela.

El primer estándar MPI 1.0 fue acabado y publicado en mayo 1994. El estándar ha sido actualizado desde entonces, estando actualmente en desarrollo el MPI 2.



MPI (Message Passing Interface)

Estándar

El estándar define la sintaxis y la semántica de un núcleo de rutinas de biblioteca útiles para una amplia gama de usuarios que escriben programas portátiles de paso de mensajes en C , C ++ y Fortran.

```
#include "mpi.h"
```

MPI (Message Passing Interface)

- MPI es una especificación para programación de paso de mensajes, que proporciona una librería de funciones para C, C++ o Fortran que son empleadas en los programas para comunicar datos entre procesos.

- MPI es la primera librería de paso de mensajes estándar y portable, especificada por consenso por el MPI Forum, con unas 40 organizaciones participantes, como modelo que permita desarrollar programas que puedan ser migrados a diferentes computadores paralelos.

MPI (Message Passing Interface)

Características de MPI

- Estandarización.
- Portabilidad: Es portable a cualquier plataforma paralela. Multiprocesadores, multicomputadores, redes, heterogéneos, ...
- Buenas prestaciones.
- Amplia funcionalidad.
- Existencia de implementaciones libres (mpich, LAM-MPI, ...)
- Es simple, con solo 6 funciones se puede implementar cualquier programa
- Soporta el modelo SPMD.

MPI (Message Passing Interface)

Características de programación

El usuario escribirá su aplicación como un proceso secuencial del que se lanzarán varias instancias que cooperan entre sí.

Los procesos invocan diferentes funciones MPI que permiten:

- iniciar, gestionar y finalizar procesos MPI
- comunicar datos entre dos procesos
- realizar operaciones de comunicación entre grupos de procesos
- crear tipos arbitrarios de datos

MPI (Message Passing Interface)

Modelos de programación

Se pueden agrupar en:

- Comunicación punto a punto: Intercambio de info entre pares de procesos.
- Comunicación colectiva: Intercambio de info entre conjunto de procesos
- Gestión de datos: Definir tipos de datos derivados, se puede enviar datos de info no contiguos en memoria.
- Operaciones de alto nivel: Grupos, contextos, comunicadores, topologías.
- Operaciones avanzadas (MPI-2, MPI-3): Entrada-salida, creación de procesos, comunicación unilateral

MPI (Message Passing Interface)

Ventajas

- Universalidad
- Fácil comprensión
- Gran expresividad
- Mayor eficiencia

Desventajas

- Programación compleja
- Control total de las comunicaciones.

MPI (Message Passing Interface)

Funciones básicas

- Todas ellas empiezan por MPI_ y obligan a que los programas MPI tengan #include "mpi.h“.
- Los programas MPI deben ser obligatoriamente inicializados y finalizados en MPI (MPI_Init, MPI_Finalize).
- Los procesos en ejecución pueden saber cuántos procesos MPI forman parte de un grupo de procesos -***communicator*** en la terminología MPI- (MPI_Comm_size) y qué número de orden -empezando por 0- tiene en ese grupo de procesos (MPI_Comm_rank).

```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, size;

  MPI_Init(&argc, &argv);

  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  printf("Hola! Soy el %d de %d\n", rank, size);

  MPI_Finalize();

  return 0;
}
```

MPI (Message Passing Interface)

Funciones básicas

- Los mensajes punto a punto deben ser enviados explícitamente por el emisor y recibidos explícitamente por el receptor ,para lo cual pueden emplearse dos funciones básicas (MPI_Send y MPI_Recv).
- **Fichero cabecera:**

```
#include <mpi.h>
```

- **Formato de las funciones:**

```
codigo_error = MPI_nombre( parámetros ... )
```

- **Inicialización:**

```
int MPI_Init ( int *argc , char ***argv )
```

MPI (Message Passing Interface)

Funciones básicas

- **Comunicador:** Conjunto de procesos que se intercomunican. Por defecto podemos utilizar MPI_COMM_WORD , en cuyo caso el grupo de procesos es el conjunto de procesos lanzados conjuntamente para resolver un problema
- **Identificación de procesos:**

  MPI_Comm_rank (MPI_Comm comm , int *rank)
- **Procesos en el comunicador:**

  MPI_Comm_size (MPI_Comm comm , int *size)
- **Finalización:**

  int MPI_Finalize ( )

MPI (Message Passing Interface)

Funciones básicas

- **Mensajes:**

Un mensaje estará formado por un cierto número de elementos de un mismo tipo MPI.

- **Tipos MPI básicos:**

MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHOT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	
MPI_PACKED	

MPI (Message Passing Interface)

Funciones básicas

- **Envío de un mensaje a otro proceso:**

  int MPI_Send (void *posicion_de_memoria , int contador , MPI_Datatype tipo , int destino , int etiqueta , MPI_Comm comunicador)

- **Recepción de un mensaje de otro proceso:**

  int MPI_Recv (void *posicion_de_memoria , int contador , MPI_Datatype tipo , int origen , int etiqueta, MPI_Comm comunicador , MPI_Status *estado)

- El receptor puede emplear MPI_ANY_TAG y/o MPI_ANY_SOURCE

MPI (Message Passing Interface)

Ejemplo

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, size;

  MPI_Init(&argc, &argv);

  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  printf("Hola! Soy el %d de %d\n", rank, size);

  MPI_Finalize();

  return 0;
}
```

MPI (Message Passing Interface)

LAM-MPI

- LAM-MPI (LAM viene de Local Area Multicomputer) es una implementación libre de MPI. Es un simple pero poderoso entorno para ejecutar y monitorizar aplicaciones MPI sobre redes de ordenadores.
- El sistema LAM-MPI está compuesto de tres partes

➢una librería de funciones

➢un *daemon*, lamd, que se ejecutará en todos los procesadores del multicomputador (virtual)

➢una serie de órdenes para gestionar y monitorizar el multicomputador (virtual)

- Las diferentes órdenes son explicadas con su manual mediante el correspondiente **man**

MPI (Message Passing Interface)

Arranque de LAM

- El usuario crea un fichero que indique las máquinas que vayan a formar parte del multicomputador virtual.
- **lamboot** [-v ficherohosts]: Arranca LAM (el multicomputador virtual) en una máquina o en un conjunto de máquinas.
- **recon** [-v ficherohosts]: Verifica que se puede arrancar LAM en una máquina o en un conjunto de máquinas.
- **Lamnodes:** muestra los nodos (procesadores) que forman parte del multicomputador virtual.

MPI (Message Passing Interface)

Compilación de programas MPI en LAM

mpicc: es un "atajo" para ejecutar el compilador cc que además encuentra los ficheros "include" de MPI y enlaza con las librerías MPI necesarias.

ejemplo: mpicc -o programa programa.c

Ejecución de programas MPI en LAM

mpirun. Una aplicación MPI con el modelo SPM se lanza con una simple orden que indica cuántas instancias del programa se ejecutarán.

- **ejemplo:** mpirun -np 4 programa

MPI (Message Passing Interface)

Motorización de aplicaciones MPI en LAM

- El estado de los diferentes procesos MPI y de los mensajes que se intercambian puede ser monitorizado en todo momento.
- **Mpitask**

muestra el estado de los procesos en ejecución.

- **Mpimsg**

muestra los mensajes que estén enviándose o que no hayan sido recibidos todavía

MPI (Message Passing Interface)

Limpieza de LAM

- **lamclean** [-v]

Elimina todos los procesos MPI lanzados por el usuario y que no hayan terminado.

Terminación de LAM

- **lamhalt**

Termina la sesión de LAM, eliminando el *daemon* lamd, y debe ser ejecutado cuando no se vaya a seguir trabajando con LAM-MPI

MPI (Message Passing Interface)

Bibliografía

- https://www.youtube.com/watch?v=uiZU05CfZUg
- http://informatica.uv.es/iiguia/ALP/materiales2005/2_2_introMPI.htm
- https://en.wikipedia.org/wiki/Message_Passing_Interface

UTE EC

¡GRACIAS!

TRAS CENDE MOS