UAEM | Universidad Autónoma del Estado de México

Programación Paralela y Distribuida

Centro Universitario UAEM Valle de México

Centro Universitario UAEM Valle de México

Unidad 1 Paragdigmas y Bases del Cómuto Paralelo

1.2 Paradigmas de la Programación en Paralelo

Ingeniería en Computación (ICO)

Ph. D. Victor Manuel Landassuri Moreno

vmlandassurim@uaemex.mx

landassuri@gmail.com

PROGRAMA DE ESTUDIOS POR COMPETENCIAS

PROGRAMACIÓN PARALELA Y DISTRIBUIDA

I. IDENTIFICACIÓN DEL CURSO

ORGANISMO ACADÉMICO: Facultad de Ingeniería						
PROGRAMA EDUCATIVO: Ingeniería en Computación				ÁREA DE DOCENCIA: Programación e Ingeniería del Software		
APROBACIÓN POR LOS H.H. CONSEJOS ACADÉMICO Y DE GOBIERNO		FECHA:		PROGRAMA ELABORADO POR: Ing. Luis E. Ledezma Fuentes Ing. Élfego Gutiérrez Ocampo	PROGRAMA REVISADO POR: Integrantes de la Academia de Programación e Ingeniería de Software	
				FECHA DE ELABORACIÓN : Enero 2011	FECHA DE REVISIÓN : Mayo 2011	
CLAVE	HORAS DE TEORÍA	HORAS DE PRÁCTICA	TOTAL DE HORAS	CRÉDITOS	TIPO DE UNIDAD DE APRENDIZAJE	NÚCLEO DE FORMACIÓN
L41057	2	3	5	8	Curso	Obligatoria
PRERREQUISTOS: Programación Avanzada, Estructuras de Datos		UNIDAD DE APRENDIZAJE ANTECEDENTE: Ninguna		UNIDAD DE APRENDIZAJE CONSECUENTE: Ninguna		
PROGRAMAS EDUCATIVOS O ESPACIOS ACADÉMICOS EN LOS QUE SE IMPARTE: Licenciatura en Ingeniería en Computación (Facultad. de Ingeniería, Centros Universitarios: Atlacomulco, Ecatepec, Texcoco, Valle de Chalco, Valle de México, Valle de Teotihuacán, Zumpango)						

Índice de Contenidos

- Descripción de la unidad de aprendizaje
- Introducción
- Unidad 1.
  - Paradigmas y bases del cómputo paralelo

    1.2 Paradigmas de la Programación en paralelo Vistas en este juego de diapositivas
- Guion explicativo
- Referencias

Descripción de la unidad de aprendizaje

Identificación del Curso

Ingeniería en Computación – 6º semestre

Horas de Teoría: 2 hrs.

Horas de Práctica: 3 hrs.

Créditos: 8

Unidad de Aprendizaje Antecedente:
Ninguna

Unidad de Aprendizaje Consecuente:
Ninguna

Lineamientos

DEL DOCENTE

- Establecer las políticas del curso al inicio del mismo.
- Respetar el horario del curso y la forma de evaluarlo.
- Cumplir el temario y el número de horas asignadas al curso o justificar la ausencia por adelantado (asistencia a conferencias, etc.)
- Asesorar y guiar el trabajo de las unidades de aprendizaje.
- Retroalimentar el trabajo de los alumnos.
- Fomentar la creatividad en los alumnos a través del desarrollo de proyectos.
- Evaluar y Calificar a los alumnos.
- Preparar el material didáctico para las clases y prácticas.

DEL DISCENTE

- Contar con la asistencia establecida en el reglamento de Facultades:
  - 80% para examen ordinario
  - 60% para examen extraordinario
  - 30% para examen a titulo de suficiencia
- Cumplir con las actividades encomendadas entregando con calidad en tiempo y forma los trabajos requeridos.
- Participar activa y críticamente en el proceso de enseñanza-aprendizaje.
- Hacer uso adecuado de las instalaciones y equipo de cómputo.
- Realizar las evaluaciones que se establezcan.
- Mantener unas pautas de comportamiento socialmente aceptables cuando se encuentre en clases y laboratorio.
- Cuando se requiera, entregar a tiempo y forma los trabajos requeridos.

Propósito

Presentar al alumno con la tecnología de vanguardia en el diseño de algoritmos y programación en paralelo haciendo uso de computadoras con 2 o más procesadores o haciendo uso de procesadores de doble núcleo, con vistas a capacitar al estudiante a su egreso en el análisis, diseño, desarrollo y construcción de sistemas de resolución de problemas de gran envergadura en donde se requiera el uso de arquitecturas paralelas o sistemas distribuidos.

Competencias genéricas

- Analizar y diseñar algoritmos y programas haciendo uso de técnicas de programación en paralelo aplicables a la tecnología computacional.
- Analizar y diseñar proyectos donde se requiera programación en paralelo y distribuida.
- Comunicarse con expertos de otras áreas.
- Utilizar eficazmente computadoras con 2 o más procesadores.
- Analizar soluciones del entorno y problemas propios de ser tratados mediante sistemas computacionales paralelos y distribuidos.
- Proponer soluciones eficaces y eficientes.

Competencias genéricas

- Crear nuevas ideas para la solución de problemas.
- Aplicar los conocimientos mediante prácticas.
- Conocer la temática básica de la profesión que desempeña en la práctica.
- Especificar arquitecturas de computadoras de arquitectura paralelas o sistemas distribuidos.
- Diseñar, desarrollar y dar mantenimiento a sistemas paralelos o distribuidos.
- Conocer la temática básica sobre programación paralela y distribuida.

Ámbito de desempeño

- Empresas públicas y privadas
- Investigación de nuevas soluciones computacionales
- Docencia a cualquier nivel de aprendizaje escolarizado
- Desarrollo de proyectos.
- Análisis, diseño, implementación y mantenimiento de sistemas computacionales

Estructura

Unidad 1.	• **Paradigmas y bases del cómputo paralelo**. Dar a conocer las bases del cómputo paralelo en general.
Unidad 2.	• **Arquitecturas paralelas y Sistemas de interconexión**. Dar a conocer las distintas formas de conectar procesadores con procesadores, procesadores con memoria.
Unidad 3.	• **Técnicas de Diseño de algoritmos paralelos y distribuidos.** Comprender y entender técnicas de diseño de algoritmos en paralelo y distribuidos.

Estructura

Unidad 4.	• **Balanceo de Carga.** Comprender y aplicar técnicas de balanceo de carga computacional durante el diseño de programas en paralelo y distribuidos.
Unidad 5.	• **Sistemas de memoria Compartida.** Modelar el comportamiento de un sistema de memoria compartida en paralelo.
Unidad 6.	• **Sistemas de Memoria Distribuida.** Modelar el comportamiento de un sistema de memoria distribuida en paralelo.

Estructura por unidad

Unidad 1. Paradigmas y bases del cómputo paralelo y distribuido

- Conceptos generales.
- Paradigmas de la Programación en paralelo
- Constitución de una computadora paralela.
- Modelos de arquitectura (MIMD, SIMD, SISD, MISD).
- Cómputo Paralelo y Programación en Paralelo.
- Memoria Compartida y Distribuida.
- Red de Conexión

Estructura por unidad

Unidad 2. Arquitecturas Paralelas y Sistemas de interconexión

- Sistemas de arquitectura (SMP, MPP, COW, DSM).
- Modelos de acceso a memoria (UMA, NUMA, COMA, NORMA).
- Ley de Amdahl.
- Ley de Gustafson.

Estructura por unidad

Unidad 3. Técnicas de Diseño de algoritmos paralelos y distribuidos.

- Técnicas de algoritmos paralelos (PRAM, APRAM, C3).

- Particionamiento, Comunicación, Aglomeración y Mapeo (PCAM)

Estructura por unidad

Unidad 4. Balanceo de carga.

- Medición de balance de carga.
- Asignación dinámica de procesos.
- Balanceo de carga dinámico, robusto y no centralizado

Unidad 5. Sistema de memoria compartida.

- Uso de Java
- Open MP

Estructura por unidad

Unidad 6. Sistema de memoria distribuida

- Algoritmo paralelo diseñándolo con memoria distribuida.
- Instrucciones de paralelización (mpi, pvm, java) aplicados a problemas numéricos.

Procedimientos de Evaluación

- Derecho a examen
  - ordinario: al menos un 80% de asistencia.
  - Extraordinario: al menos un 60 % de asistencia.
  - Título de suficiencia: al menos un 30% de asistencia

- **Evaluación Ordinaria**
- Manejo de aspectos teóricos (1º y 2º parcial) 40%
- Evaluación continua 20%
- Proyecto Final 40%
- Exentos: Calificación total mayor o igual a 80%.

- **Calificación final extraordinaria y a Título:**
  - 30 % del examen escrito
  - 70 % proyecto de investigación

Proyecto final

- Realizar un programa, en el que estudiante escoja la tarea a resolver, y en donde se usen diferentes comandos de las librerías de openMPI.
  - El problema a resolver tiene que ser de mayor complejidad a los vistos en el curso,
  - Ejemplo: superior a un “hola mundo” o un simple deadlock.

Proyecto final

- Para tu proyecto final
  - Queda muy bien implementar
  - Algoritmos de procesamiento de imagénes
    - CUDA
  - Cómputo intensivo requerido para
    - Análisis estadísticos o de algoritmos de inteligenci artíficial
      - Redes Neuronales Artificiales
      - Algoritmos Evolutivos
      - Etc.
- Se te puede entregar alguno de ellos, en código serial, para que tu lo implementes en el lenguaje adecuado.

Proyecto final - Aspectos a evaluar

- Si tu escojes el proyecto, este debe ser:
  - Un problema interesante de paralelizar o realizar en cómputo dsitribuido
  - Que tan optima fue la paralelización
    - Programación, recursos de procesamiento y memoria
  - No contiene Warnings
  - Se ejecuta sin bugs el código
  - Se emplean estructuras de datos adecuadas y/o memoria dinámica
  - Se emplean librerías de openMPI, openMP o CUDA

Introducción

Descripción

- En estas diapositivas se abordarán una parte de los Paradigmas de la Programación en paralelo
  - Esto de forma teórica en su mayoría, las siguientes unidades tendrán más práctica.

- Anterior a esto, debes de revisar el juego de diapositivas de conceptos generales.

- Favor de revisar el guión explicativo al final del éstas diapositivas.

Unidad 1
Paradigmas y bases del cómputo Paralelo y Distribuido

Tema 1.2
Paradigmas de la programación en Paralelo

Contenido

- Computación concurrente
  - Coherencia y seguridad
    - Problema: Cena de los filósofos
- Programación Paralela
- Programación Distribuida
  - Comparación entre Paralelo y Distribuido
  - Ejemplos

1.2.1 Computación concurrente

- Se puede decir que:
  - Es la ejecución de múltiples tareas interactivas al mismo tiempo
  - Es la base de la computación paralela
  - Tareas que se ejecutan al mismo tiempo
  - Colección de procesos o hilos secuenciales en paralelo
  - Pseudo-paralelismo: cuando la tareas se ejecutan en una maquina secuencial

¿Cómo se puede implementar la concurrencia?

- Multiprogramación
  - Los hilos se multiplexan durante ejecución sobre el equipo secuencial
- Multiprocesamiento
  - Los hilos se multiplexan durante ejecución sobre equipos multi-núcleo o multiprocesadores
- Procesamiento distribuido
  - Los procesos se multiplexan durante su ejecución a lo largo de diferentes equipos

Coherencia y Seguridad

- Hilos que accedan recursos compartidos
- Seguridad
  - Todos los accesos no tienen efecto sobre los recursos
    - Ejemplo: Variables
    - Solo un acceso a la vez (Exclusión mutua)
- ***Un problema clásico***
  - ***Se ejemplifica con la cena de los filósofos***

Problema: Cena de los filósofos

- Es un problemas en ciencias de la computación
- Ejemplo del problema de concurrencia de algoritmos que ilustra la sincronización
- Originalmente formulado en 1965 por Edsger Dijkstra
  - Para mostrar como las computadoras pueden competir por acceso a una cinta magnetica

Cena de los filósofos

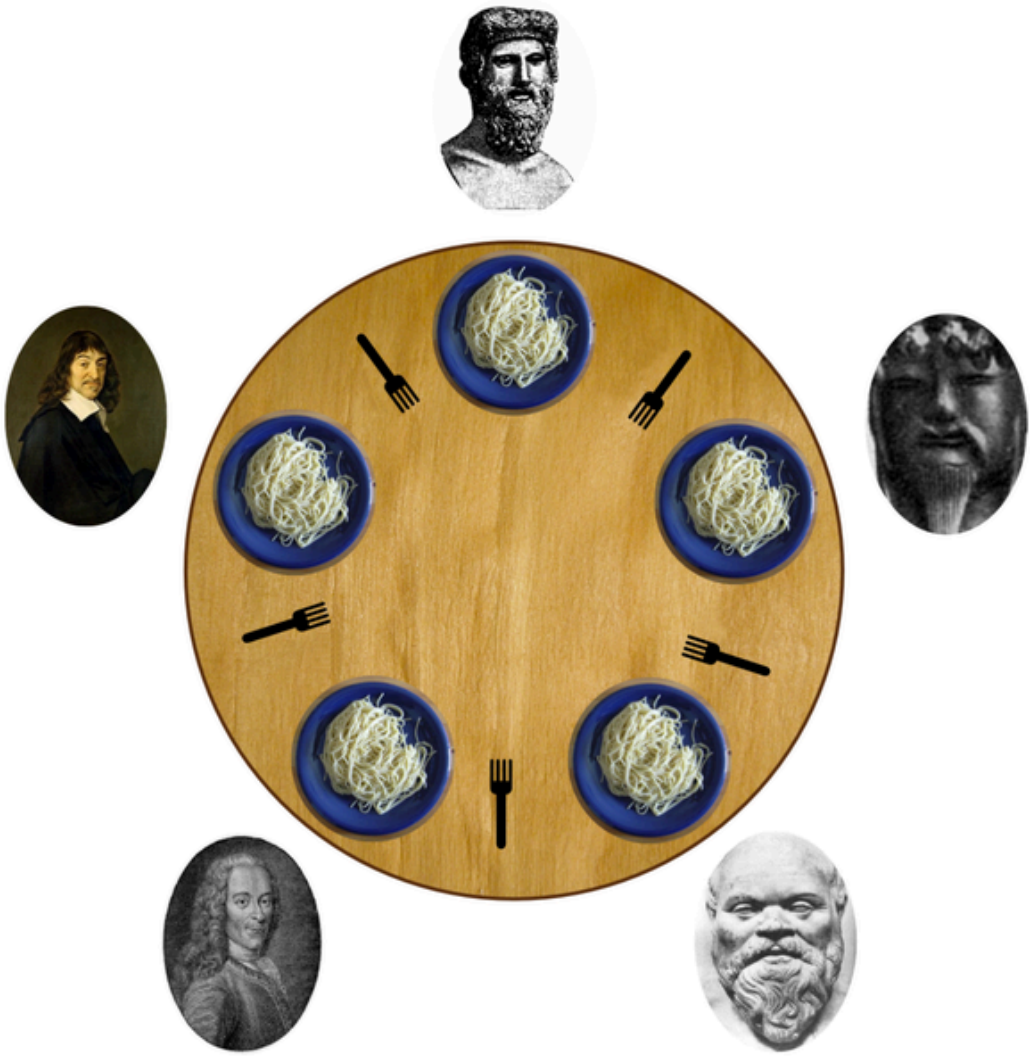


Imagen recuperada de http://en.wikipedia.org/wiki/File:Dining_philosophers.png

Cena de los filósofos

- Hay 5 filósofos sentados en una mesa redonda
- Entre cada filósofo hay un tenedor
- Cada filósofo tiene dos actividades a realizar
  - Pensar
  - Comer
- El filósofo piensa por un rato
  - Cuando tiene hambre, toma el tenedor derecho e izquierdo
  - No puede comer si no estan ambos tenedores en la mesa

Cena de los filósofos

- Libera los tenedores cuando termina de comer

- Problema: Competencia, coherencia y seguridad de recursos
  - Semáforos, Exculsión Mutua, etc.

- A continuación explicaremos los paredigmas de Computao Paralelo y Distribuido

1.2.2 Sistema Paralelo

- Cada procesador puede tener su propia memoria
  - En la mayoría de los casos se encuentran servidores con más de un procesador
    - Memoria compartida (Temas vistos mas adelante en el curso)
- La direcicón de memoria de un procesador puede mapear en la memoria de otro procesador
  - Ahí, hay direcciones globales de memoria a través de los procesadores
  - Cada procesador opera independientemente de forma concurrente
  - El acceso a los datos de otro procesador en otro servidor lo define explicitamente el usuario
    - Paso de mensajes, sincronización, etc.

Sistema Paralelo - Ventajas

- Muy poco tiempo en transmitir mensajes entre procesadores, o servidores
  - Dado que está fuertemente conectado
  - Recursos muy cercanos entre ellos
    - Usualmente no mayor a una habitación.
- Sistemas dedicados a cómputo intensivo
  - Muy rápido en procesamiento

Sistema Paralelo - Desventajas

- Desventajas
  - El consumo electríco es un problema, dado que recae en una sola institución (por lo regular)
  - Por ejemplo, la carga de trabajo podría repartirse en usuarios de todo el mundo (en sus PCs)
  - Un ejemplo de ello se dio con el PayStation 2, donde usuarios de todo el mundo podían ejecutar programas para el análisis de proteinas.
    - https://en.wikipedia.org/wiki/Folding@home

1.2.3 Sistema Distribuido

- Cada procesador tiene su propia memoria
- La direcicón de memoria de un procesador no mapea en la memoria de otro procesador
  - No hay direcciones globales de memoria a través de los procesadores
  - Cada procesador opera independientemente
  - El acceso a los datos de otro procesador lo define explicitamente el usuario
    - Paso de mensajes, sincronización, etc.

Sistema Distribuido

- Ventajas
  - Memoria escala con el número de procesadores
  - Cada procesador accede a su propia memoria sin interferencia ni sobrecarga (overhead) causado por mantener la coherencia de la memoria
- Desventajas
  - El programa es responsable de la comuniación
  - Difícil mapear datos basados en memoria global

Cosas en común

- Múltiples procesadores
- Los procesadores están interconectados por alguna red
- Múltiples procesos están en progreso al mismo tiempo, y pueden cooperar los unos con los otros

Prog. paralela VS Prog. distribuida

- Por un lado, una aplicación es dividida en tareas y ejecutada de forma simultanea
  - Programación Paralela

- Por otro lado, se divide la aplicación en diversas tareas y se ejecutan usando diferentes recursos
  - Programación Distribuida

Prog. paralela VS Prog. distribuida

Computación paralela

- Divide una aplicación en tareas que se ejecutan al mismo tiempo (fuertemente acoplado)
- Se considera una aplicación a la vez
- **Objetivo**: Acelerar la ejecución de una aplicación
- Se ejecutan sobre arqeuitecturas homogéneas con memoria compartida

Computación distribuida

- Utiliza diferentes recursos físicamente separados
- Se consideran varias aplicaciones a la vez (pueden pertenecer a diferentes usuario)
- **Objetivo**: permitir que varios usuarios trabajen en forma cooperativa
- Se ejecutan sobre arquitecturas heterogéneas, abiertas y dinámicas

Ejemplo: Hola Mundo MPI

- Dentro del ámbito de cómputo paralelo
  - Podemos ir haciendo ejercicios básicos para entender el funcionamiento
- Veamos entonces, como programar en MPI
- A continuación se presenta un programa Hola Mundo
- Debes nombrar al archivo como:

  MPI_HelloW.c

Hola Mundo MPI

```
/* compile: mpicc MPI_HelloW.c -o MPI_HelloW, ejecute: mpiexec -n 5 MPI_HelloW,  alternative, ejecute with mpirun -np 5 MPI_HelloW, where 5 is for the number of process to run */

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

int main(int argc, char** argv) {
   int myrank, nprocs;

   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
   MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

   printf("Hello from processor %d of %d\n", myrank, nprocs);

   MPI_Finalize();
   return 0;
}
```

Hola Mundo MPI

```
/* compile: mpicc MPI_HelloW.c -o MPI_HelloW, ejecute: mpiexec -n 5 MPI_HelloW,  alternative,
ejecute with mpirun -np 5 MPI_HelloW, where 5 is for the number of process to run */

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

int main(int argc, char** argv) {
   int myrank, nprocs;

   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
   MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

   printf("Hello from processor %d of %d\n", myrank, nprocs);

   MPI_Finalize();
   return 0;
}
```

Comentario

Librerías – bibliotecas de funciones

Inicia el código en paralelo

Termina el código en paralelo

Hola Mundo MPI

- Nótese que se usa la **librería** de MPI
  - #include <mpi.h>
- Es buen momento de mencionar que **no** es el termino correcto decir librería
  - Más apropiado sería decir biblioteca de funciones
  - Sin embargo, por uso comun se le seguira llamando librerías, y el alumno debe tener encuenta esa distinción.

Hola Mundo MPI

- MPI_Init(&argc, &argv);
  - Despliega todos los hilos de procesamiento, o procesadores, que ejecutarán cada uno de los procesos
  - Si te fijas con cuidado, le estamos pasando la dirección de memoria de las dos variables que recibe el método ***main*** en sus argumentos.

int main(int argc, char argv)**  **MPI_Init(&argc, &argv);**

Hola Mundo MPI

- A partir de este punto, se tienen copias identicas de todo el código en cada uno de los hilos
  - La diferencia es que ellos trabajarán de forma concurrente, y ejecutarán el código que les corresponde

Hola Mundo MPI

- MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
  - Comando para determinar el número de procesos en ejecución
  - El resultado se almacena en **nprocs**
    - Se puede usar otros nombres de variables
  - MPI_COMM_WORLD
    - Es el comunicador por default, el encargado de poner en contacto a todos los hilos

MPI_COMM_WORLD

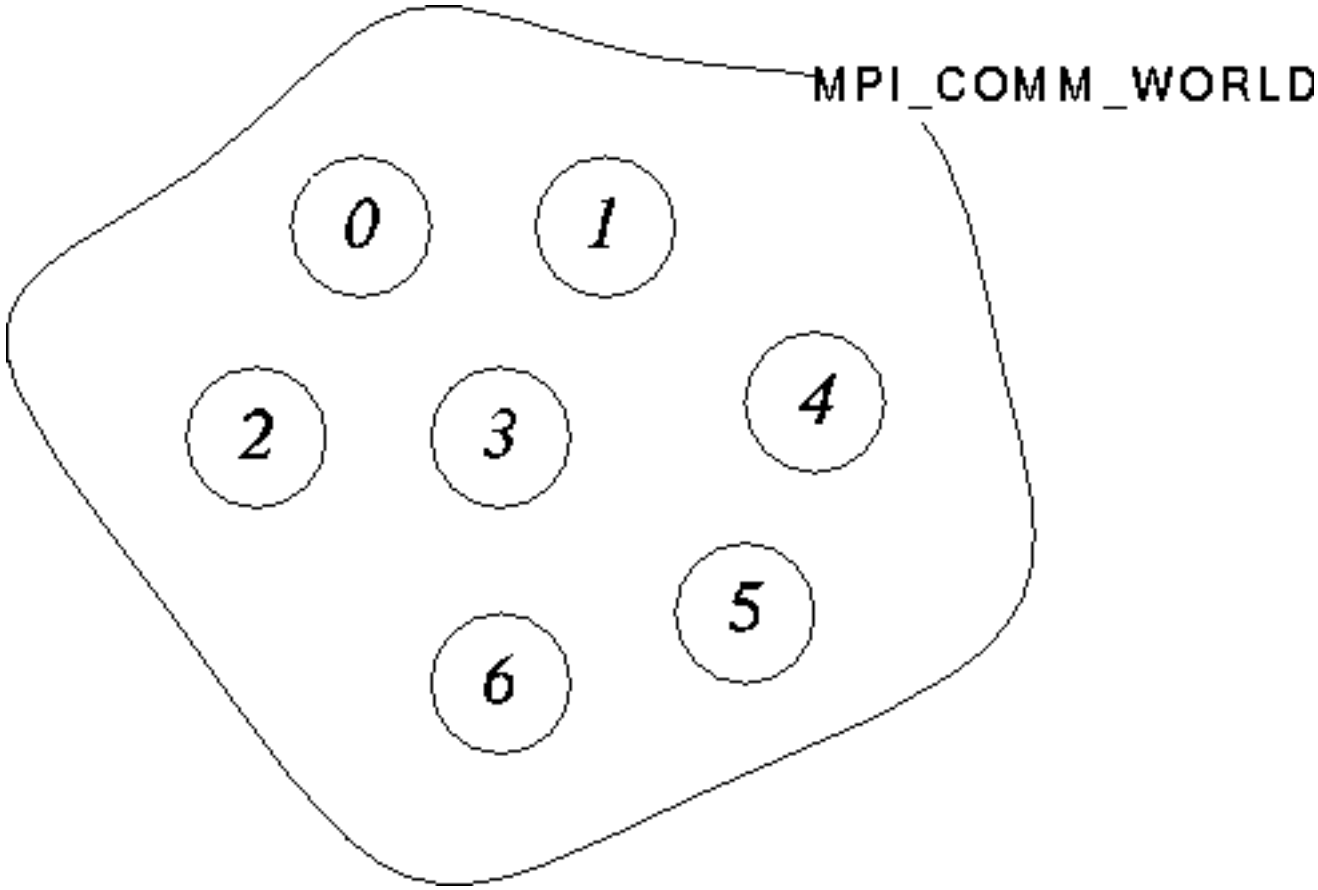


Imagen tomada del NCI National Facility https://nf.nci.org.au/training/MPIProg/slides/slides.028.html

MPI_Comm_rank

- MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  - Función para obtener el identificador de procesador en ejecución
    - También puede ser llamado hilo generado por MPI
  - El resultado se lamacena en **myrank**
  - Sirve para distiguir a un procesador de otro
  - Así, sirve para saber quíen ejecurará qué código

- MPI_Finalize();
  - Se cerciora de que todos los hilos hayan terminado, y regresa de nuevo un solo hilo de ejecución.

Otro jemplo del hola mundo en MPI

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char *argv[]) {
   int rank;
   int size;
   MPI_Init(&argc, &argv);

MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   if (rank == 0) {
      printf("Hola MPI desde C\n");
   }
MPI_Comm_size(MPI_COMM_WORLD, &size);

   printf(" Hay %d procesos en el mundo de MPI in my world, y mi Identificador (rank) es %d\n",  size, rank);

   MPI_Finalize();
return 0;
}
```

Ejemplo: Multiplicar un vector por una matriz

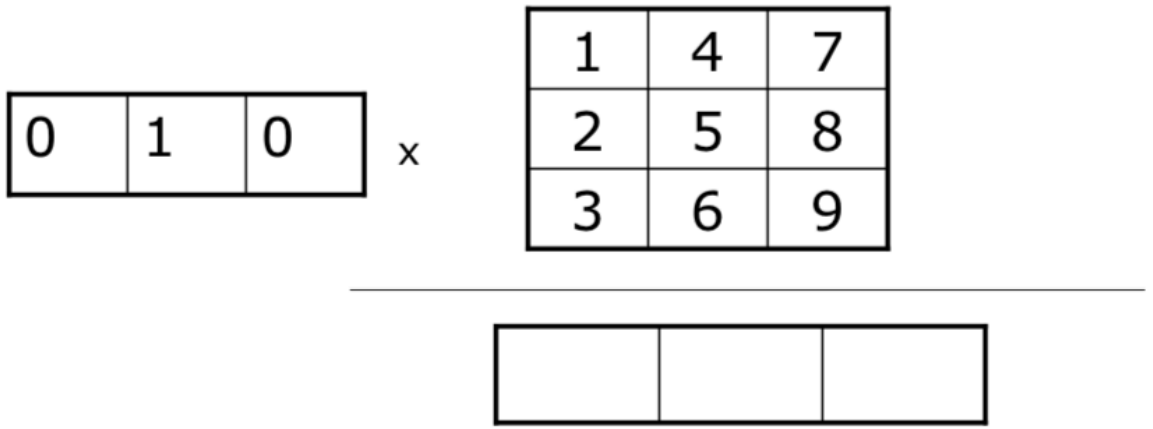
$$D= \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{pmatrix}, \quad F= \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

$$D \times F= \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{pmatrix} \times \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} d_{11}f_1 + d_{12}f_2 + \cdots + d_{1n}f_n \\ d_{21}f_1 + d_{22}f_2 + \cdots + d_{2n}f_n \\ \vdots \\ d_{m1}f_1 + d_{m2}f_2 + \cdots + d_{mn}f_n \end{pmatrix}$$

D es de m x n

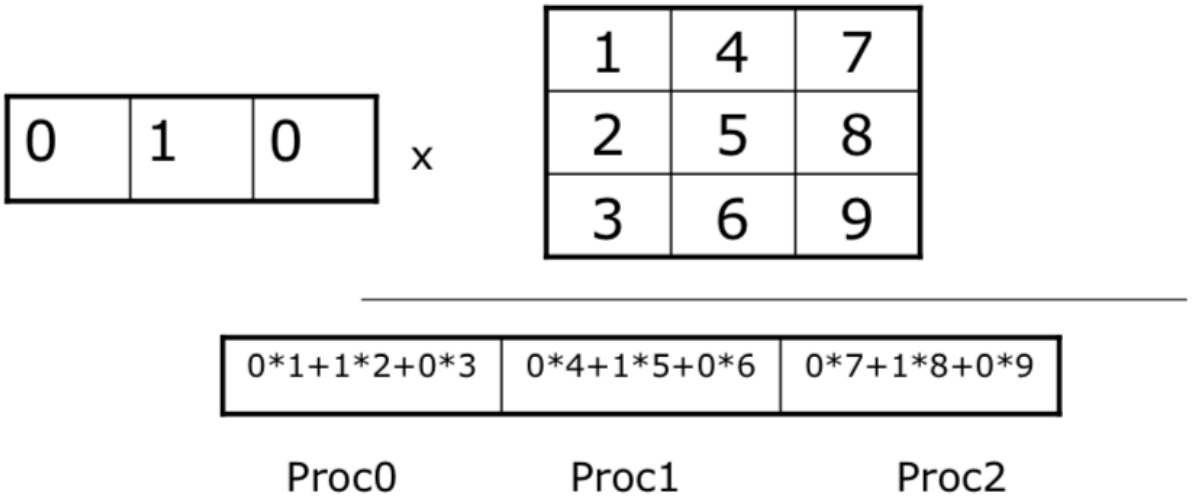
F de n x 1

Multiplicar un vector por una matriz – 1 procesador



- Se tiene tres resultados finales
- 5 operaciones por resultado
- Total 15 operaciones, ejecutado en en un procesador, podríamos decir 15 unidades de tiempo
- ¿Recuerdas como hacer esta operación?

Multiplicar un vector por una matriz – n procesadores



- Si lo realizamos en paralelo
  - Se tiene tres resultados finales (igual que antes)
  - 5 operaciones por resultado **ajecutadas al mismo tiempo en tres procesadores**

Resumen

- Aquí se mostro un panarama general de la Concurrencia, así como una descripción y comparación de la Programación Paralela y Distribuida.

- Así, se puede ver que ambos dividen el problema en tareas pequeñas, donde:

Resumen

- La programación paralela ejecuta masivamente las tareas en equipos fuertemente conectados (cientos o miles de procesadores sin mucha distancia entre ellos)
- La programción distribuida, podría en un caso, repartir las tareas en diversas partes del mundo.

- Aquí fuen presentado como hacer un programa en MPI, con el avance del curso, iremos profundizando más en éste y otros lenguajes de programación.

Guion Explicativo

- Las diapositivas presentadas debe leerse en el orden que aparecen deseado.

- Se recomienda compilar y ejecutar los dos programas aquí mostrados.

- Se recomienda al estudiante repasar conceptos de programación en C como:
  - Memoría dinámica, estructuras de datos, y estructuras (struct).

BIBLIOGRAFÍA

- BÁSICA:
  - Parte de estas diapositivas estan basadas en las diapositivas de Veronica Gil-Costa. Paradigmas de Computación Paralela, concurrente y Distribuida.
  - Así como del curso: “Introduction to programming with MPI”. CSAR Applications Support. Impartido en la Universidad de Birmingham, B15 2TT, UK en 2009.
  - O. Bonorden et all - PUB library, Release 6.0 - User guide and function reference. 1998.
  - O. Bonorden et all - The Puderborn University BSP (PUB) Library-Desing, Implementation and
  - performance. 1999.
  - M. Goudreau et all - Towards Efficiency and Portability: Programming with the BSP model. 1996.
  - J. Keller et all - Practical PRAM programming. John Wiley & Sons inc.. 2001.

BIBLIOGRAFÍA

- COMPLEMENTARIA:
  - C. Leopold - Parallel and Distributed Computing: A survey of models, paradigms, and approaches. 2001.
  - W.F. Mccoll - BSP Programming. 1994.
  - M. Quinn - Parallel Computing. Theory and Practice. Second Edition. McGraw-Hill. Inc. 1994.
  - L.G. Valiant - A Bridging Model for Parallel Computation. 1990.
  - General Purpose parallel Architectures. 1990.
  - B. Wilkinson, et all - Parallel Programming: Tecniques and Aplications using Networked Workstations and Parallel Computers. 1999.
  - OpenMPI. http://www.open-mpi.org/
  - OpenMP. http://www.openmp.org/blog/