Introducción a la Programación Paralela

Dr. Mario Rossainz López
FCC- BUAP

Programación Concurrente y Paralela
Otoño 2022
NRC: 60898

Introducción
Taxonomía
Paradigmas
Diseño y Comportamiento

Programación Paralela

Introducción:

Objetivo del paralelismo: Conseguir ejecutar un programa inicialmente secuencial en menos tiempo utilizando para ello varios procesadores.

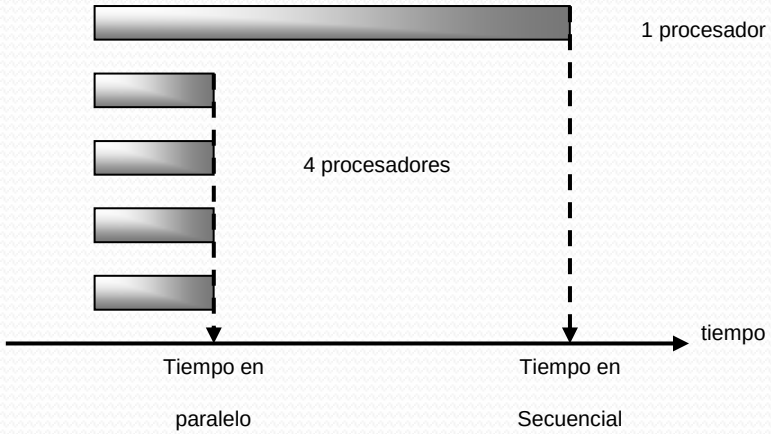
Idea: Dividir un problema grande en varios más pequeños y repartirlos entre los procesadores disponibles.

Problemas: Estrategia de diseño adecuada, elementos no paralelizables, reparto de trabajo entre los procesadores.

Programación Paralela

Introducción:

Esquema Ideal de la paralelización

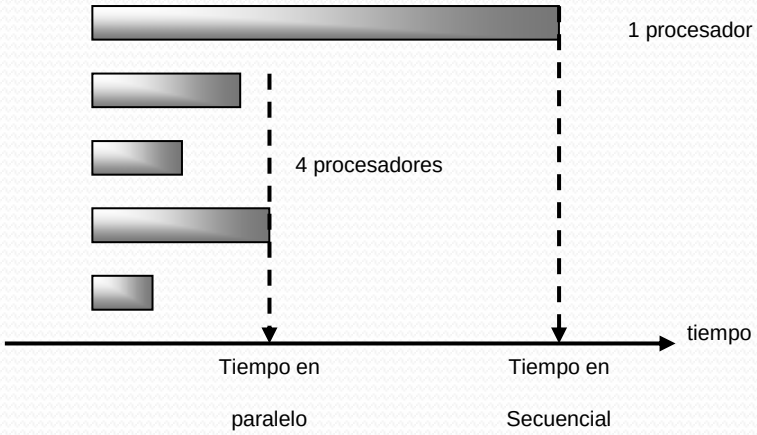


Suponiendo que todos los procesadores llevan a cabo el mismo tipo y número de operaciones, el tiempo de ejecución de un programa paralelo sería: t/n con t=tiempo de ejecución secuencial y n=número de procesadores disponibles

Programación Paralela

Introducción:

Esquema REAL de la paralelización



Existen dependencias secuenciales entre las variables del programa, por tanto la unidad de paralelización son las entidades que encapsulan instrucciones y no las instrucciones por sí mismas.

Programación Paralela

Introducción:

Factor SpeedUp o Aceleración:

- Es una medida comparativa del rendimiento de un sistema multiprocesador respecto de un sistema monoprocesador.

- Mide qué tan efectiva es la paralelización de un programa frente a su versión secuencial.

Programación Paralela

Introducción:

Factor SpeedUp o Aceleración:

$$SpeedUp = \frac{T_{sec}}{T_{par}}$$

- T_{sec}: Tiempo de ejecución del programa secuencial
- T_{par}: Tiempo de ejecución del programa paralelo

Programación Paralela

Introducción:

Factor SpeedUp o Aceleración:

$$SpeedUp = \frac{T_{sec}}{T_{par}}$$

CONSIDERACIONES:

1. Se debe utilizar el mejor algoritmo secuencial conocido (del problema a resolver).
2. Deberá ser por tanto el algoritmo secuencial más rápido que pueda ejecutarse en un solo procesador.
3. El algoritmo utilizado para la programación paralela suele ser diferente.

Programación Paralela

Introducción:

Ejemplo de SpeedUp:

Suposiciones:

- Se tienen 4 procesadores para la ejecución paralela de la aplicación
- T_{par}= 3 hrs.
- T_{sec}= 8 hrs.

$$SpeedUp = \frac{8\ hrs}{3\ hrs} = 2.66\ hrs$$

SpeedUp Ideal = 4 hrs

Programación Paralela

Introducción:

Ejemplo de SpeedUp:

$$SpeedUp = \frac{8\ hrs}{3\ hrs} = 2.66\ hrs$$

¿Qué tan realista (bueno) es este SpeedUp?

Consideraciones:

1. Tiempo utilizado en paralelizar el programa
2. Calidad del código fuente
3. Naturaleza del algoritmo implementado
4. Porción de código intrínsecamente secuencial (qué tan paralelizable es el algoritmo)

Programación Paralela

Introducción:

Ejemplo de SpeedUp:

$$SpeedUp = \frac{8\ hrs}{3\ hrs} = 2.66\ hrs$$

¿Qué tan realista (bueno) es este SpeedUp?

RESPUESTA: El máximo speedUp que se puede alcanzar con la implementación paralela de un determinado algoritmo viene dada por la llamada *Ley de Amdahl.*

Programación Paralela

Ley de Amdahl:

Es la cota superior al SpeedUp que como máximo puede alcanzar un programa ejecutándose en paralelo, suponiendo que en todo programa hay una fracción de código que no se puede paralelizar

Sea *f*= fracción del cómputo que no puede ser dividida en tareas concurrentes

Sea *p*= el número de procesadores

S(p) = el factor de speedUp máximo dado por la ley de Amdahl y se define como:

Programación Paralela

Ley de Amdahl:

$$S(p) = \frac{p}{1+(p-1)f}$$

$$S(p) = \frac{1}{f+[(1-f)/p]}$$

Programación Paralela

Ley de Amdahl:

Ejemplo: Se tiene un programa con una fracción de código paralelo equivalente a un 75% y la aplicación se ejecuta utilizando 4 procesadores

Sea f= 0.25

Sea p= 4

$$S(4) = \frac{4}{1+(4-1)0.25} = 2.28$$

Programación Paralela

Ley de Amdahl:

Ejemplo: Se tiene un programa con una fracción de código paralelo equivalente a un 75% y la aplicación se ejecuta utilizando 4 procesadores

Sea $f = 0.25$

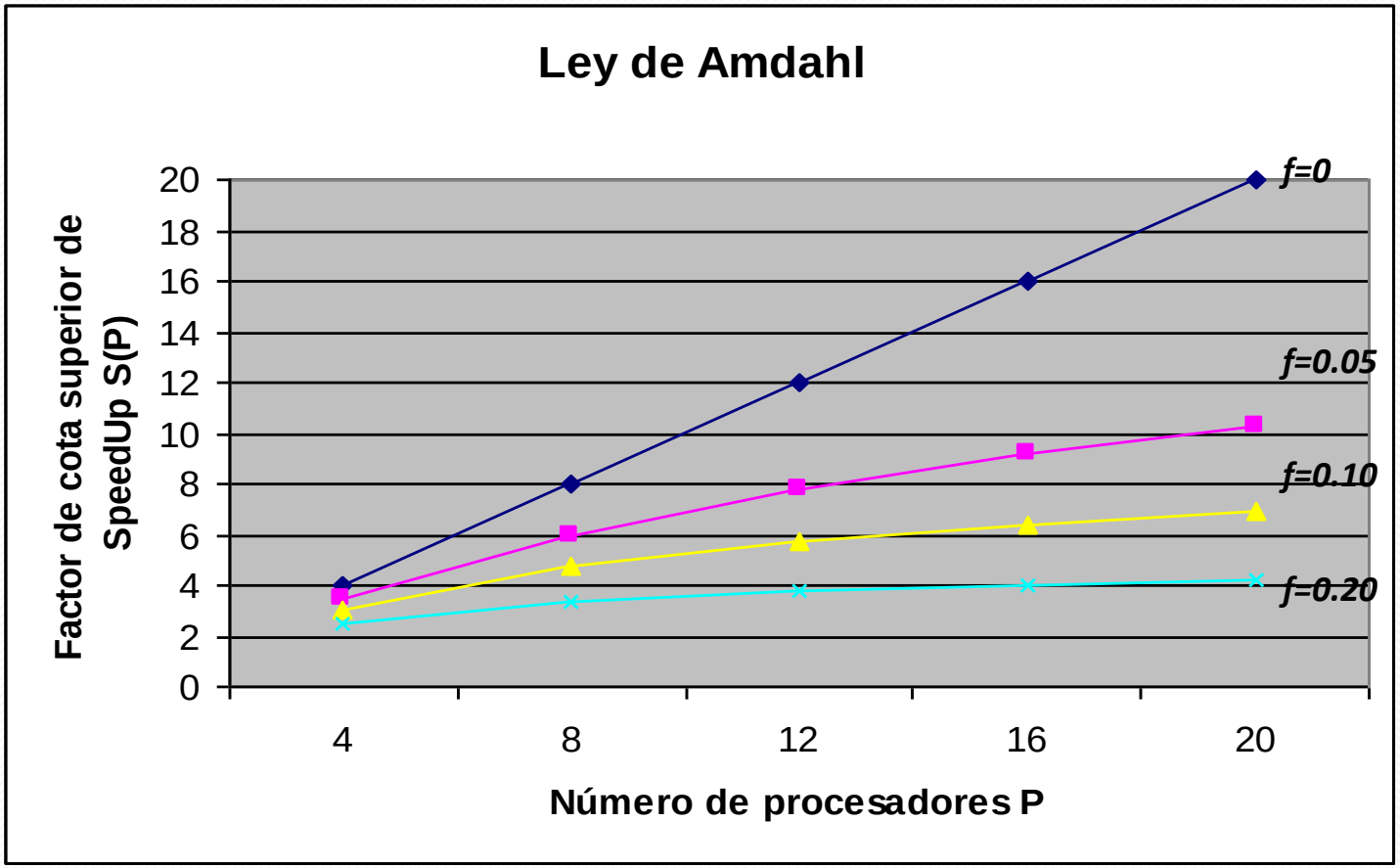
Sea $p = 4$

$$S(4) = \frac{1}{0.25 + [(1 - 0.25)/4]} = 2.28$$

SpeedUp <= S(p)

Programación Paralela

Ley de Amdahl:



Programación Paralela

Ley de Amdahl:

La cota superior del SpeedUp mejora a medida que se aumenta el número de procesadores

Con un número infinito de procesadores, el máximo speedUp que se puede alcanzar esta limitado por *1/f*

$$\lim_{p \to \infty} S(p) = \frac{1}{f}$$

Programación Paralela

Ciclos por instrucción (CPI):

Es una medida utilizada para reflejar la calidad de uso del procesador por parte de un programa

$$CPI = \frac{\#\ ciclos}{\#\ instrucciones}$$

Si por cada ciclo se ejecuta una instrucción, el CPI =1 lo cual se considera bueno

Los procesadores superescalares ejecutan más de una instrucción de un programa por ciclo, es decir, 0<CPI<=1

Programación Paralela

TAXONOMÍA:

Los multiprocesadores se clasifican de forma genérica en:

- Multiprocesadores de Memoria Compartida

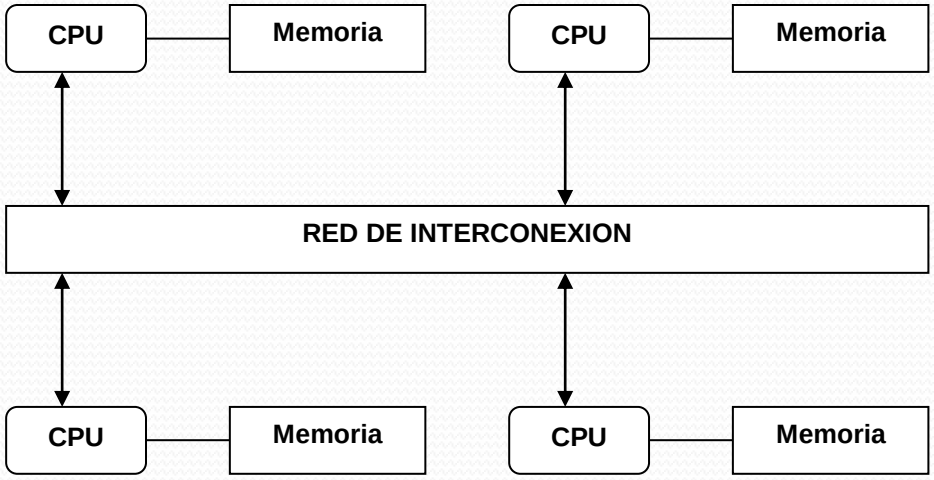
- Multiprocesadores de Memoria Distribuida

- Multiprocesadores de Memoria Compartida Distribuida

Programación Paralela

TAXONOMÍA:

Multiprocesadores de Memoria Distribuida: Computadora donde la memoria del sistema esta distribuida entre todos los procesadores del sistema, usando del paso de mensajes para la comunicación entre los procesadores



Programación Paralela

TAXONOMÍA:

Multiprocesadores de Memoria Compartida: Multicomputadoras donde todos los procesadores acceden a toda la memoria. Si los procesadores acceden a la memoria en un tiempo uniforme, entonces se tiene el modelo NUMA, de lo contrario, estaremos con un modelo UMA.

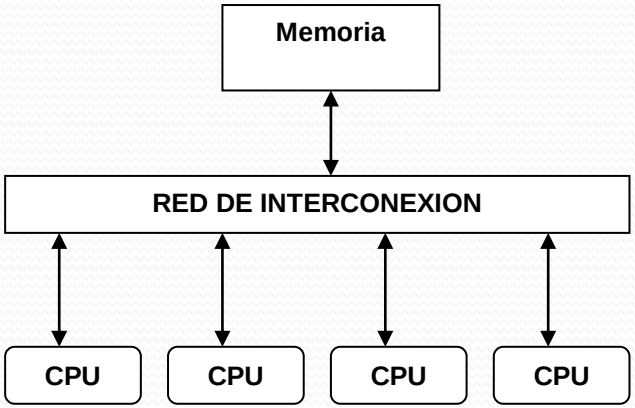


Fig. I.5. (a) Modelo NUMA

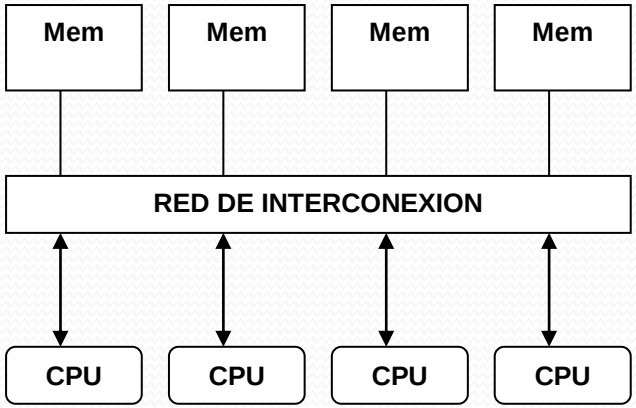
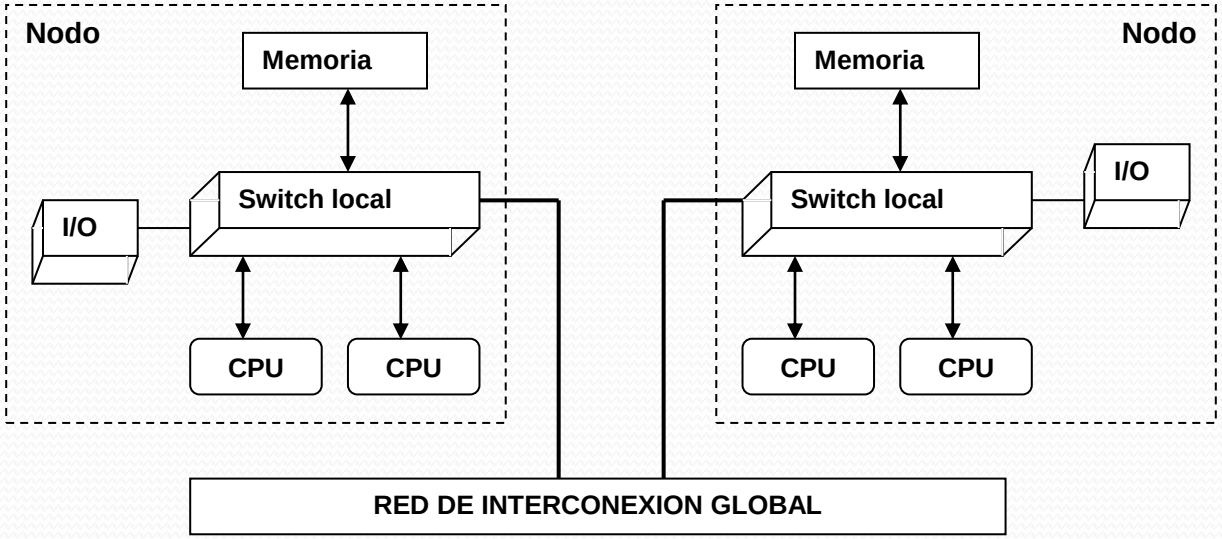


Fig. I.5. (b) Modelo UMA

Programación Paralela

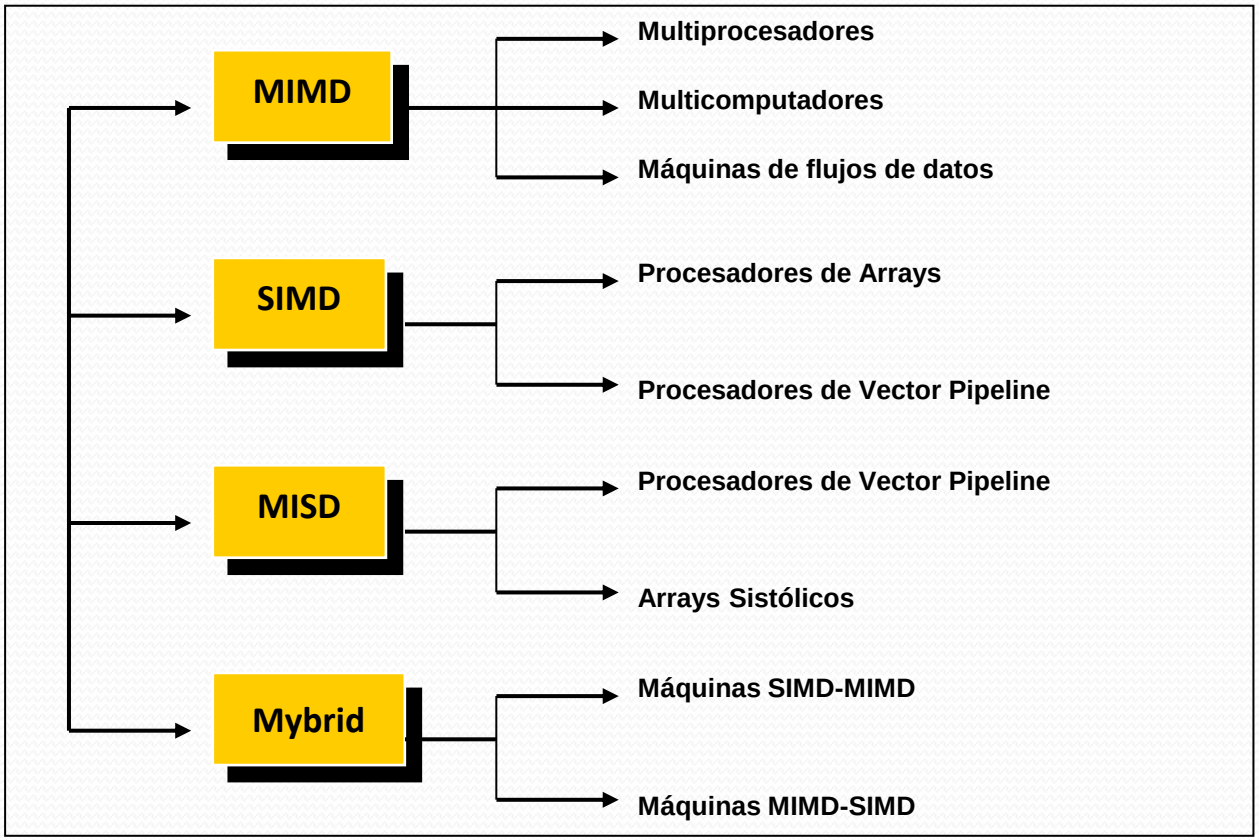
TAXONOMÍA:

Multiprocesadores de Memoria Compartida Distribuida: Computadora donde la memoria esta físicamente repartida entre los nodos del sistema compuesto por 2 procesadores c/u y solo hay un espacio de direcciones y todos los procesadores acceden a toda la memoria.



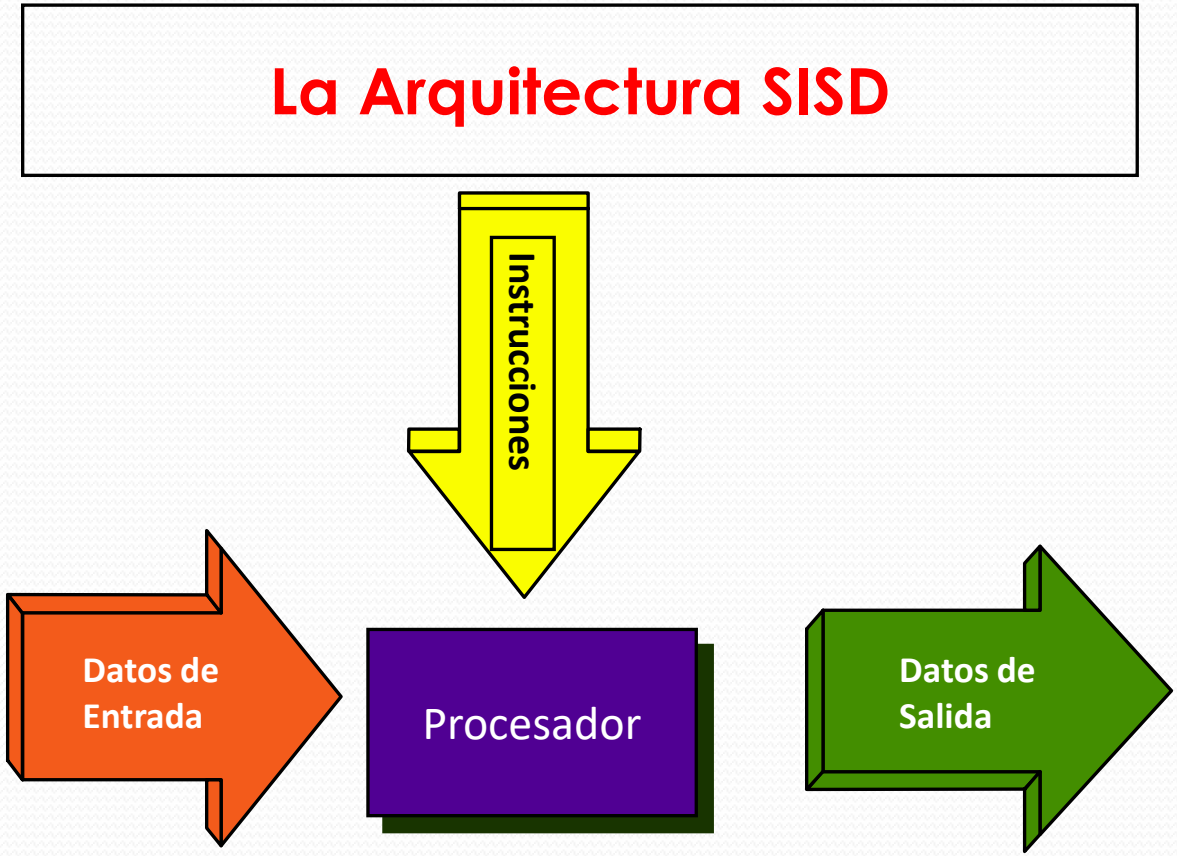
Programación Paralela

Clasificación de Michael Flynn:



Programación Paralela

Arquitectura SISD (Simple Instruction Stream, Simple Data Stream):



Programación Paralela

Arquitectura SISD (Simple Instruction Stream, Simple Data Stream):

1. Se basa en el modelo de Von-Neumman

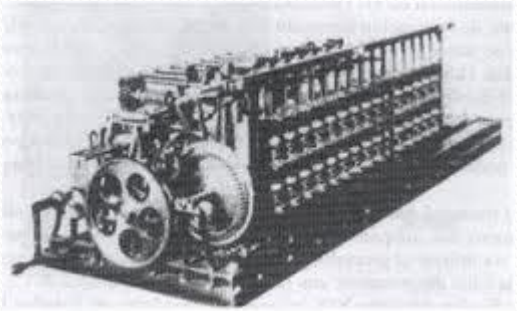
2. Cuenta con un CPU que ejecuta una instrucción a la vez e instancía un item de datos a la vez

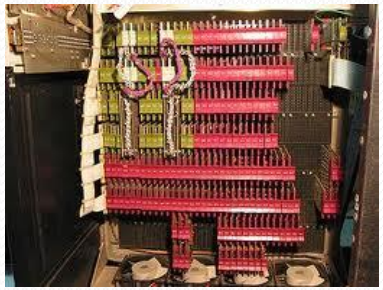
3. Utiliza un registro simple llamado contador de programa (CP) para llevar el conteo serial de las instrucciones

4. Las instrucciones son ejecutadas a través del ciclo-fetch de la máquina en orden serial

Programación Paralela

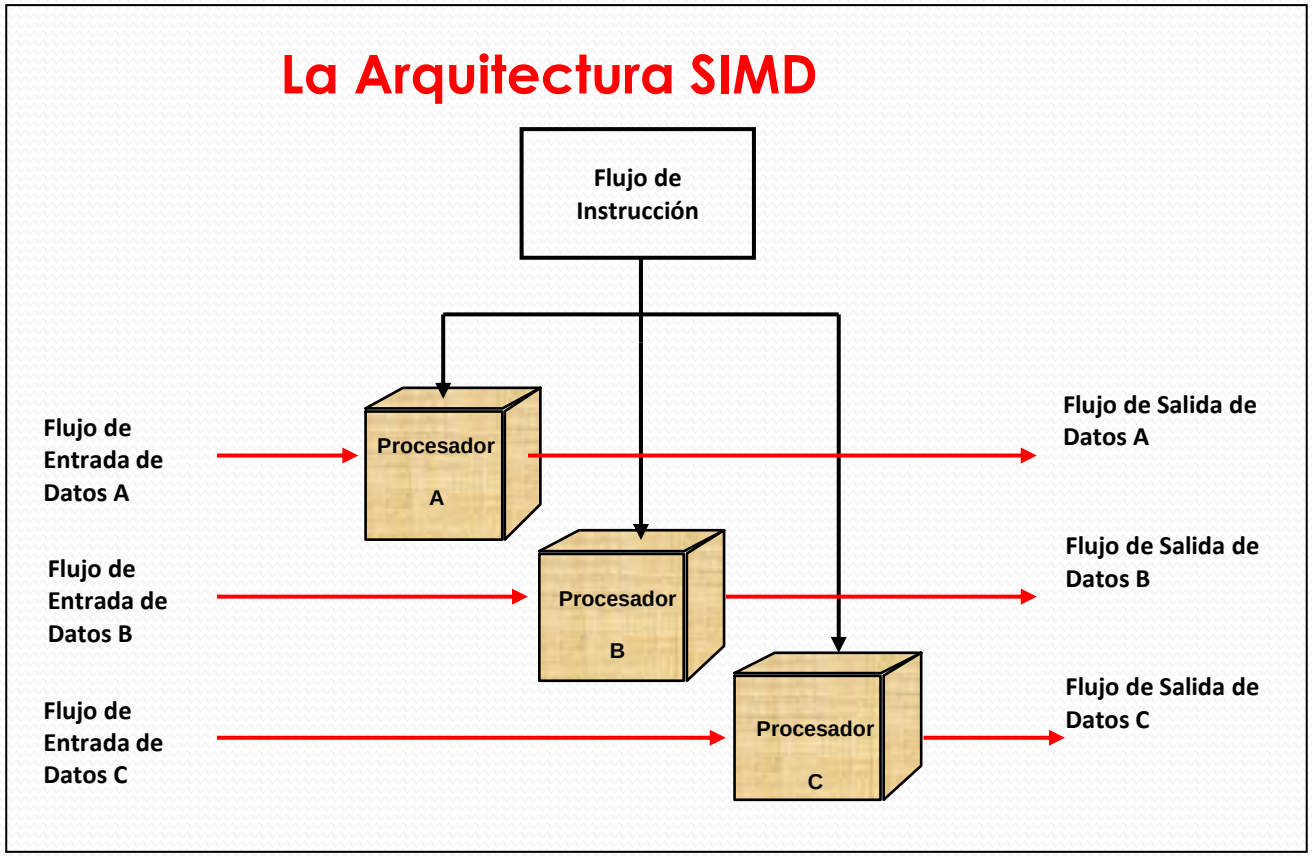
Arquitectura SISD (Simple Instruction Stream, Simple Data Stream):





Programación Paralela

Arquitectura SIMD (Simple Instruction Stream, Multiple Data Stream):



Programación Paralela

Arquitectura SIMD (Simple Instruction Stream, Multiple Data Stream):

1. Todos los CPUs reciben la misma instrucción a ejecutar por parte de la CU pero operan con distintos conjuntos de datos.

2. Distribuyen el procesamiento sobre hardware grande

3. Operan concurrentemente con muchos datos distintos

4. Realizan el mismo cálculo en todos los elementos de datos

Programación Paralela

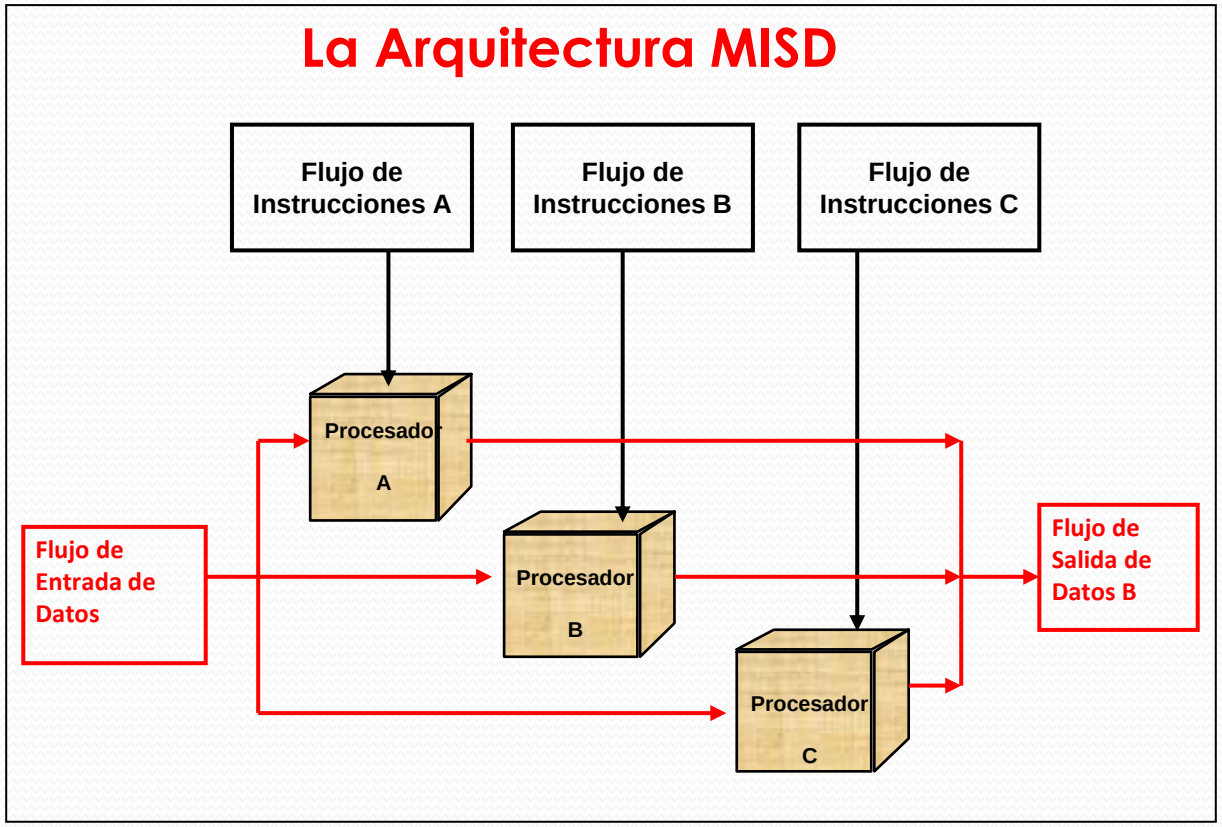
Arquitectura SIMD:

ILLIAC-IV

MasPar MP-1

Programación Paralela

Arquitectura MISD (Multiple Instruction Stream, Single Data Stream):



Programación Paralela

Arquitectura MISD (Multiple Instruction Stream, Single Data Stream):

1. Arquitectura de computadora que ejecuta varios programas distintos para el mismo conjunto de datos

2. Se pueden tener computadoras que requieren de distintas unidades de procesamiento que reciben distintas instrucciones para operar sobre los mismos datos

3. Se pueden tener computadoras tales que el flujo de datos circula sobre una serie de elementos de procesamiento

4. Ejemplo de ello son los array sistólicos o pipeline (cauces)

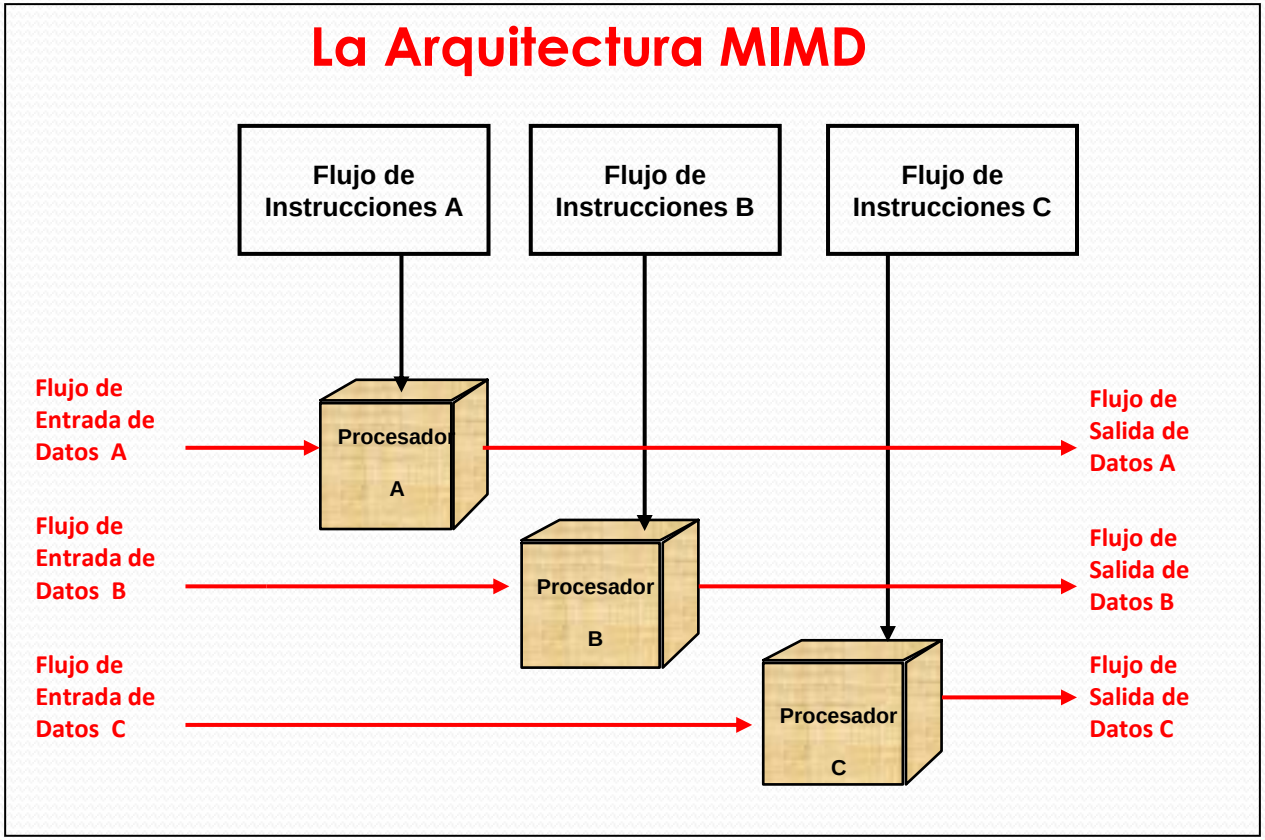
Programación Paralela

Arquitectura MISD:

NEC Nehalem Cluster

Programación Paralela

Arquitectura MIMD (Multiple Instruction Stream, Multiple Data Stream):



Programación Paralela

Arquitectura MIMD (Multiple Instruction Stream, Multiple Data Stream):

1. Son sistemas multiprocesadores o multicomputadoras donde cada procesador tiene su unidad de control y ejecuta su propio programa

2. Distribuyen el procesamiento sobre procesadores independientes

3. Comparten flujos o recursos entre procesadores

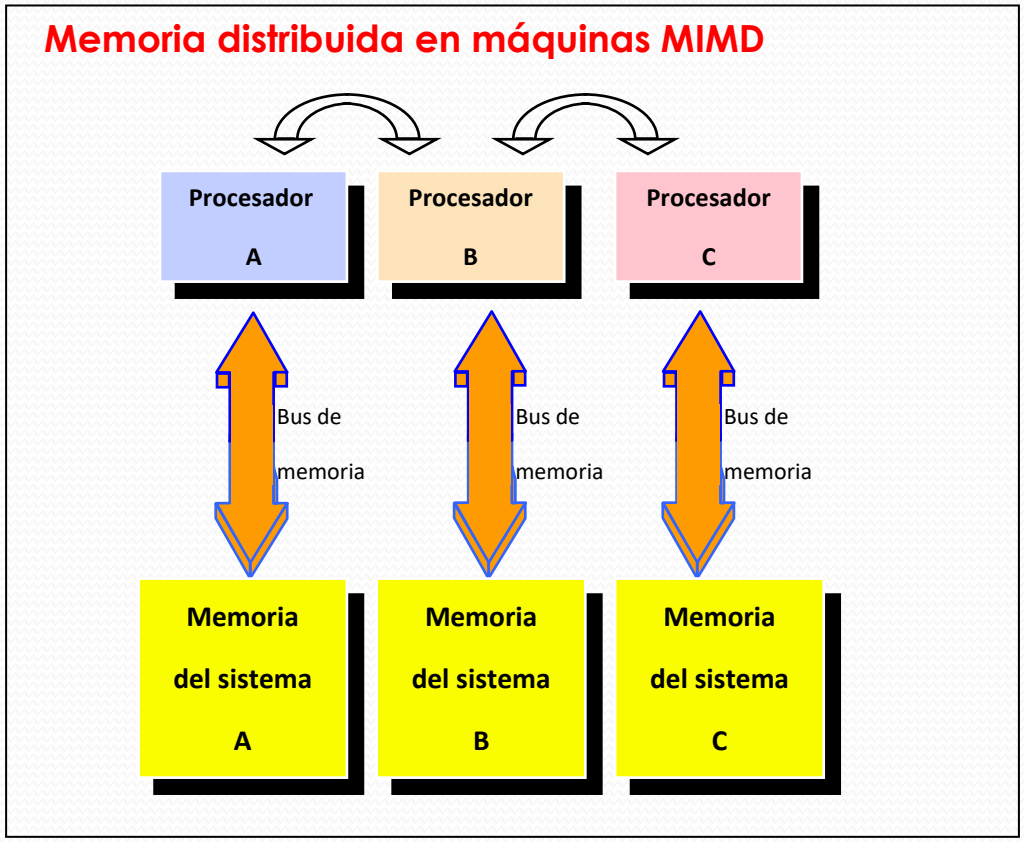
4. Son arquitecturas utilizadas para paralelismo de grano fino y obtener así más eficiencia

Programación Paralela

Arquitectura MIMD:

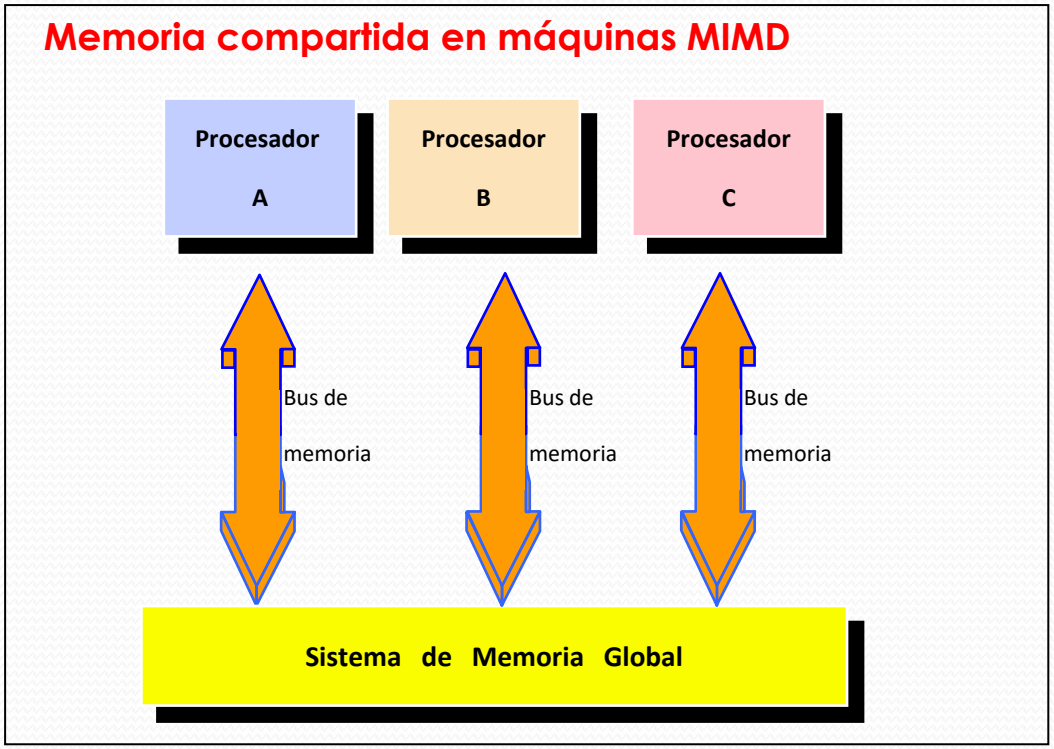
Programación Paralela

Arquitectura MIMD:



Programación Paralela

Arquitectura MIMD:



Programación Paralela

Diseño y Comportamiento:

Algoritmos Paralelos: Describen como puede resolverse un problema pensando en una determinada arquitectura paralela, mediante la división del problema en subproblemas, comunicando los procesadores y después uniendo las soluciones parciales para obtener la solución final y suelen basarse en el concepto de paradigma.

Paradigma: Es una clase de algoritmos que resuelven diferentes problemas que tienen la misma estructura de control.

- Por ejemplo: Divide y Vencerás, Multiplicación de Tuplas, Programación Dinámica, Ramificación y Poda, etc.
- Caen dentro del área de Teoría de algoritmos paralelos

Programación Paralela

Patrones de Comunicación Paralela:

Los Paradigmas definen una metodología de programación dentro de la Ciencia Computacional aplicando un proceso de derivación informal para su programación en donde se identifica un *patrón de comunicación paralela* para la interacción entre los procesos.

Patrones de comunicación paralela:

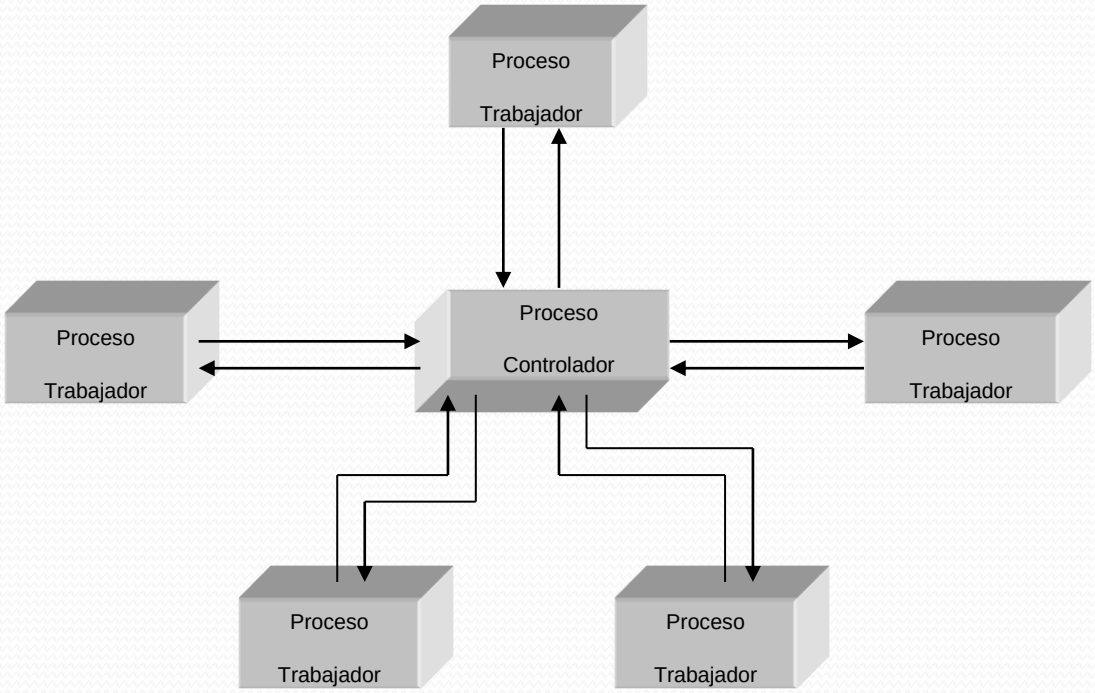
-Farms o granjas
-Pipelines o Cauces
-Trees o árboles
-Cubos
-Hipercubos
- Matrices o mallas

Programación Paralela

Diseño y Comportamiento:

Patrones de comunicación paralela:

Farms o Granjas: Formadas de un conjunto de procesos trabajadores y un proceso controlador. Los trabajadores se ejecutan en paralelo hasta alcanzar un objetivo común y el controlador distribuye el trabajo y controla el progreso del cómputo global.



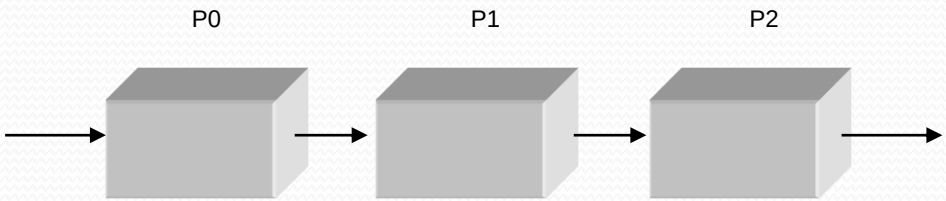
Farm con un proceso controlador y 5 procesos trabajadores

Programación Paralela

Diseño y Comportamiento:

Patrones de comunicación paralela:

Pipelines o Cauces: Esta compuesto por un conjunto de procesos (llamados stages, etapas o estados) conectados, uno detrás de otro y donde la información sigue un flujo o cauce de uno a otro stage

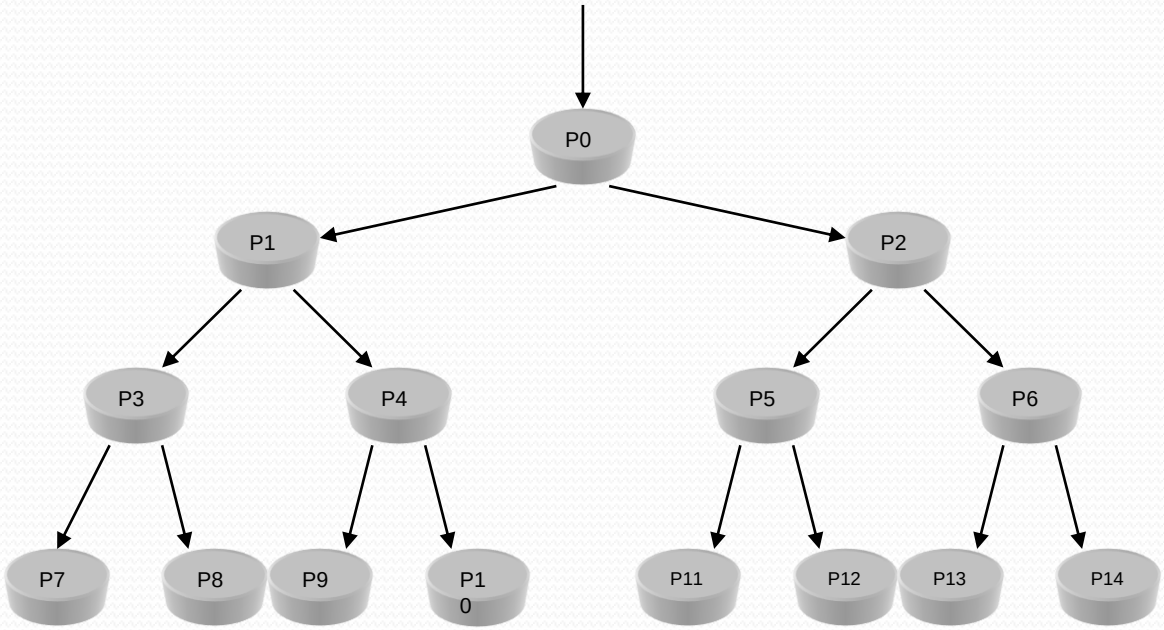


Programación Paralela

Diseño y Comportamiento:

Patrones de comunicación paralela:

Trees o Árboles: Hace que la información fluya desde el proceso raíz hacia los procesos hojas o viceversa, ejecutándose en paralelo los nodos que se encuentran en el mismo nivel del árbol

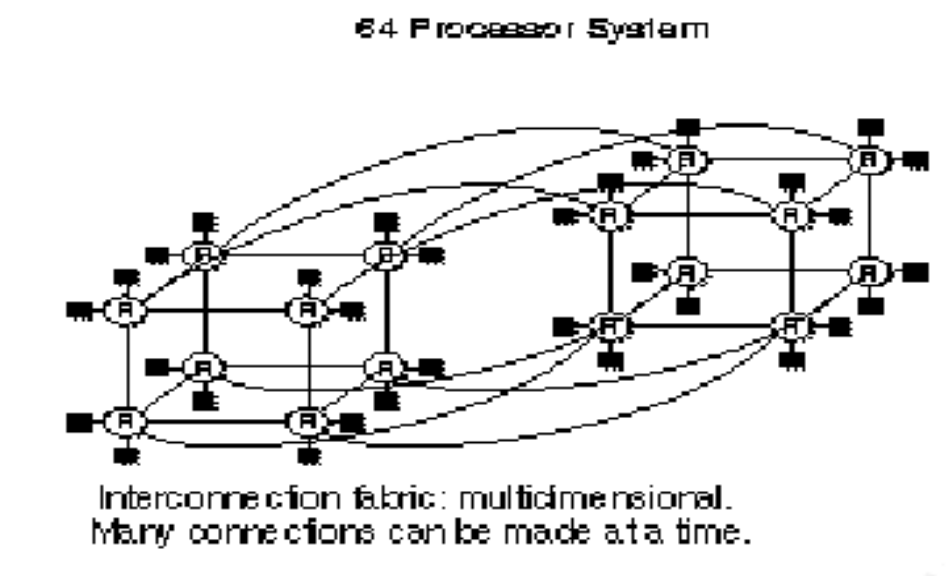


Programación Paralela

Diseño y Comportamiento:

Patrones de comunicación paralela:

Cubos o Hipercubos: Conjunto de procesos interconectados entre sí formando un cubo n-dimensional por ejemplo, dos cubos, cada uno con 8 nodos donde cada nodos esta formado por dos procesos, conectados a su vez con otros nodos formando aristas.

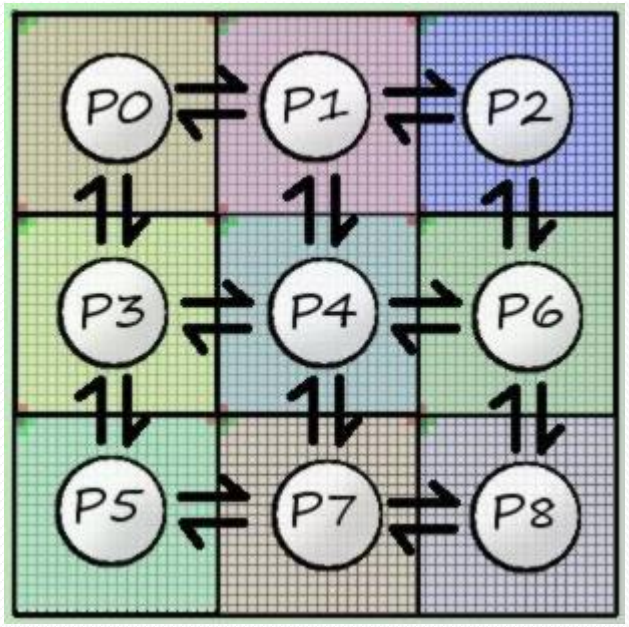


Programación Paralela

Diseño y Comportamiento:

Patrones de comunicación paralela:

Matrices o Mallas: En una malla cada proceso esta conectado con sus vecinos y todos contribuyen entre sí para resolver en común un problema, ejecutándose en paralelo.



Programación Paralela

Diseño y Comportamiento:

Metodología para definir un paradigma:

1. Identificar el comportamiento paralelo (patrón de comunicación entre procesos) de la aplicación en desarrollo.
2. Elaborar un bosquejo gráfico de su representación.
3. Realizar una definición sintáctica y/o semántica
4. Traducir dicha definición a un programa en el entorno de programación más adecuado para su implementación
5. Verificar que la semántica resultante sea la correcta probando con varios ejemplos distintos demostrando su genericidad
6. Observar el rendimiento de las aplicaciones

Programación Paralela

Diseño y Comportamiento:

Metodología para definir un paradigma:

PARADIGMA	PROGRAMA	ARQUITECTURA
MULTIPLICACION	Matrices	PIPELINE
	Rutas-Grafos	
DIVIDE Y VENCERAS	Ordenación	ARBOL
	Búsqueda	