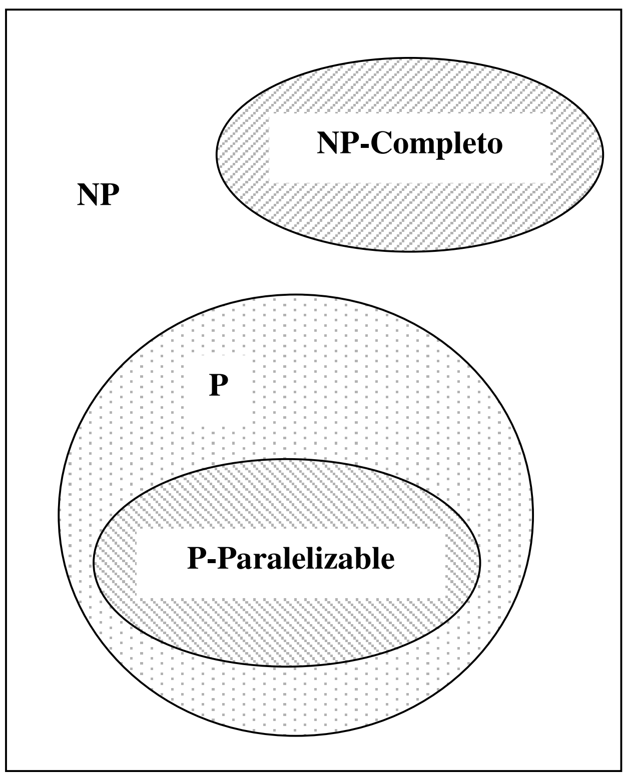
Administración de sistemas con multiprocesador

La necesidad de resolver problemas cada vez más complejos ha llevado a la electrónica a acercarse a los límites físicos de la materia. La velocidad de los procesadores tiene como límite máximo la velocidad con la que viajan los electrones a través de un conductor (velocidad de la luz).

El que existan algunos problemas que llegan a ser intratables con la arquitectura tradicional de computadoras de Von Neumann, aunado al avance más rápido de la electrónica con respecto al software, han obligado a pensar en arquitecturas paralelas de procesadores y a una nueva forma de hacer programas.

La mayoría de los sistemas actuales tienen un sólo procesador (CPU) principal. Sin embargo las nuevas tecnologías de arquitectura de computadoras permiten la existencia de máquinas con múltiples procesadores, que comparten los recursos de la computadora (memoria, bus, periféricos).

Esto permite mejorar el desempeño de los equipos, aunque no de manera lineal : una computadora con n procesadores no trabaja n veces mejor. Esto es debido a que se utiliza una parte del tiempo de los procesadores para intercambiar información de sincronización entre ellos. Además un problema "paralelizable" siempre tendrá partes intrínsecamente secuenciales. (Problemas polinomiales y no-polinomiales)



Ubicación de los problemas paralelizables en la clasificación de problemas por su complejidad computacional.

Algunos problemas tratados con máquinas paralelas son:

- Predicción del clima
- Diseño de medicinas.
- Diseño de jets.
- Diseño de mejores motores de combustión.
- Modelado del océano.
- Imagería medica, tomografía.
- Modelado de la contaminación.
- Estudio de la estructura de proteínas.
- Imágenes sintéticas.
- Apoyos educativos.

Los múltiples procesadores se usan también para reducir el efecto de las fallas en los sistemas : si un procesador falla, el sistema se hará más lento, pero seguirá funcionando (degradación progresiva o sistemas de *fallas suaves*).

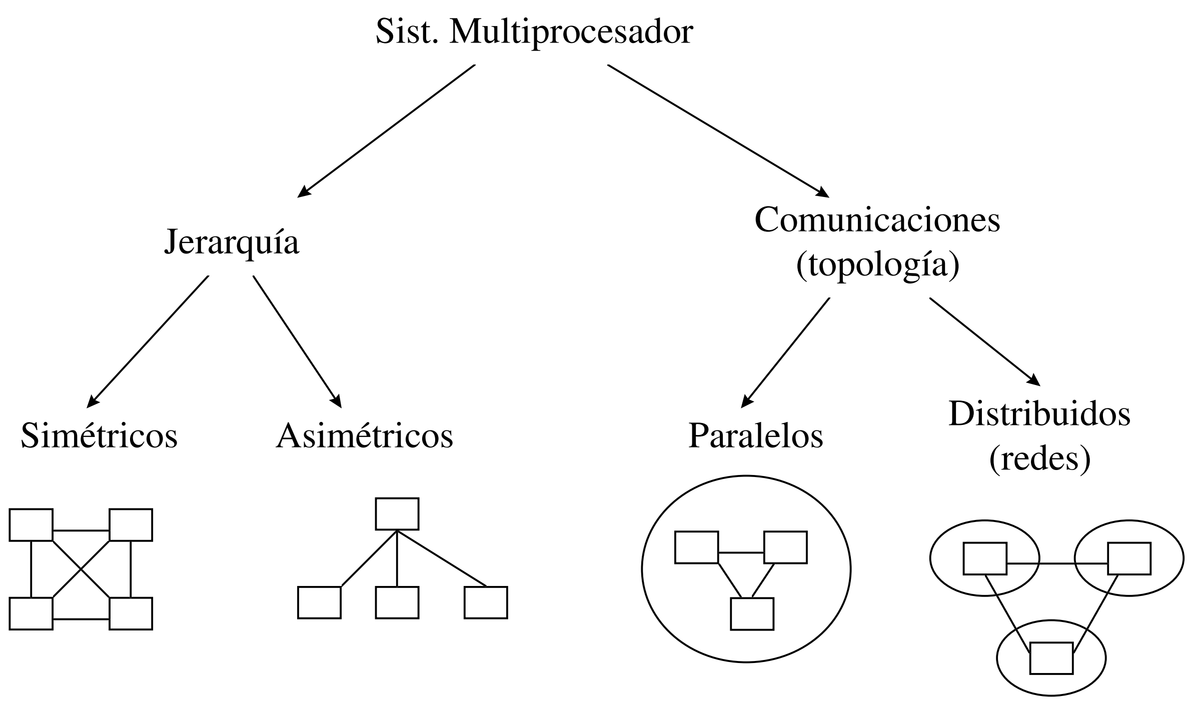
Arquitecturas paralelas

Clasificación de Flynn (1972) para sistemas de cómputo: basada en el número de flujos de instrucciones y el número de flujos de datos:

SISD Single Instruction, Single Data	Computadoras de un procesador
SIMD Single Instruction Multiple Data	Procesadores independientes ejecutando instrucciones similares sobre datos diferentes
MISD Multiple Instruction, Single Data	Procesadores ejecutando diferentes instrucciones sobre los mismos datos simultáneamente **(No existe en la realidad)**
MIMD Multiple Instruction, Multiple Data	Procesadores independientes ejecutando instrucciones diferentes (sistemas distribuidos)

Diferentes tipos de MIMD

<table>
<tr><td rowspan="2">Por la manera en que se establece la jerarquía de los procesadores</td><td>Multiprocesamiento Simétrico</td><td>Cada procesador tiene los mismos Privilegios y ejecuta tareas Equivalentes
(relación peer-to-peer).</td></tr>
<tr><td>Multiprocesamiento Asimétrico</td><td>Los procesadores tienen diferente jerarquía. Algunos actúan como controladores de otros y efectúan tareas no equivalentes
(relación maestro-esclavo)
[Común en sistemas de gran tamaño, donde la E/S consume mucho tiempo]</td></tr>
<tr><td rowspan="2">Por la forma en que se comunican los procesadores entre sí y con la memoria</td><td>Fuertemente acoplados</td><td>Comparten memoria física (Computadoras paralelas)</td></tr>
<tr><td>Débilmente acoplados</td><td>Cada procesador tiene su propia memoria (Sistemas distribuidos)</td></tr>
</table>



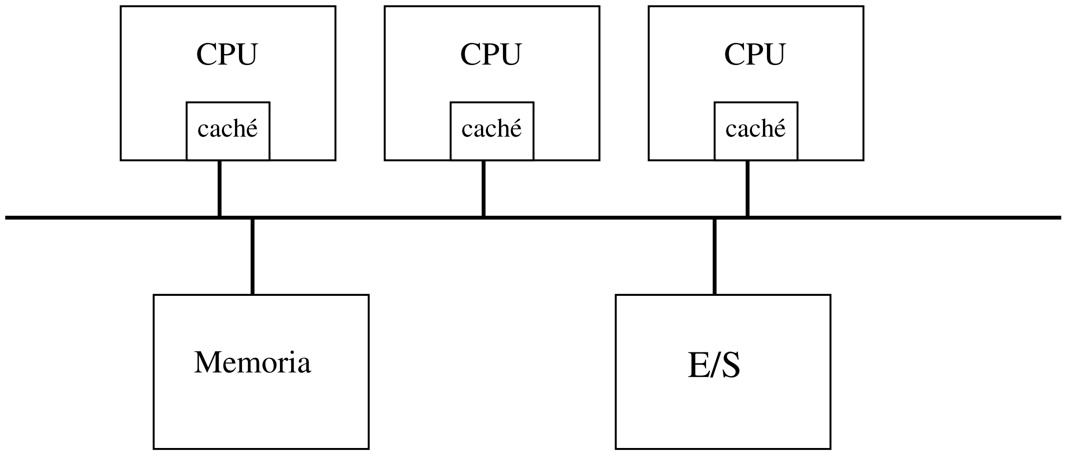
Taxonomía de arquitecturas paralelas

Según las diferentes maneras en que se controla el sistema y la forma en que se accesa a la memoria se pueden diferenciar las siguientes clases de arquitecturas paralelas:

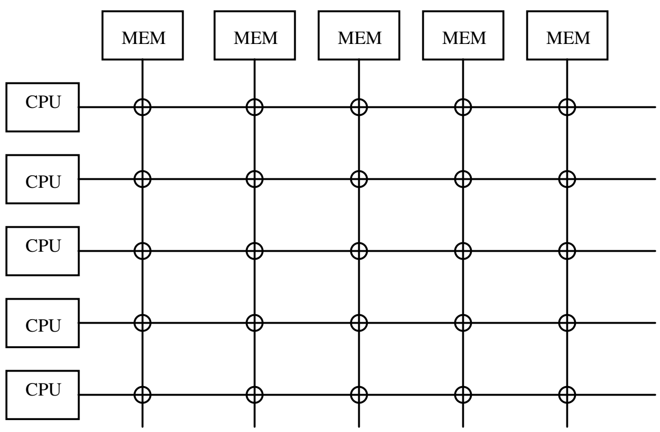
		Control	
		MIMD	**SIMD**
Memoria	**Compartida**	**Máquinas funcionales** NYU RP3 HEP Cm* Cedar Alliant	**Supercomputadoras** Cray CDC ES/9000
	Distribuida	**Paso de mensajes** Cosmic Cube Hypercube Intel SP1, SP2 *-Transputer	**Máquinas celulares y sistólicas** MPP Gapp Connection Machine

Arquitectura de comunicaciones

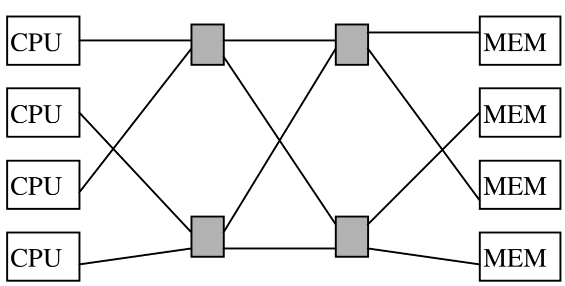
También podemos dividir a las arquitecturas paralelas por la manera en que se comunican entre sí los procesadores, la memoria y los dispositivos. Básicamente se tienen dos opciones: comunicación por buses o comunicación por conmutadores.



Bus



Conmutador de punto de cruce



Conmutador de red omega

La comunicación por bus es fácil de implementar pero difícil de controlar. A pesar de la potencia de los procesadores, el cuello de botella del procesamiento es precisamente la necesidad de compartir el bus, por lo que no es conveniente para cantidades grandes de procesadores.

El uso de memoria caché local a cada procesador evita la necesidad de utilizar el bus para llegar a la información, pero por otra parte, la actualización del caché para todos los CPUs se complica.

El uso de redes de conmutación libera de tráfico a los procesadores, permitiendo que se generen nuevos caminos para la comunicación entre pares diferentes de cpu-memoria.

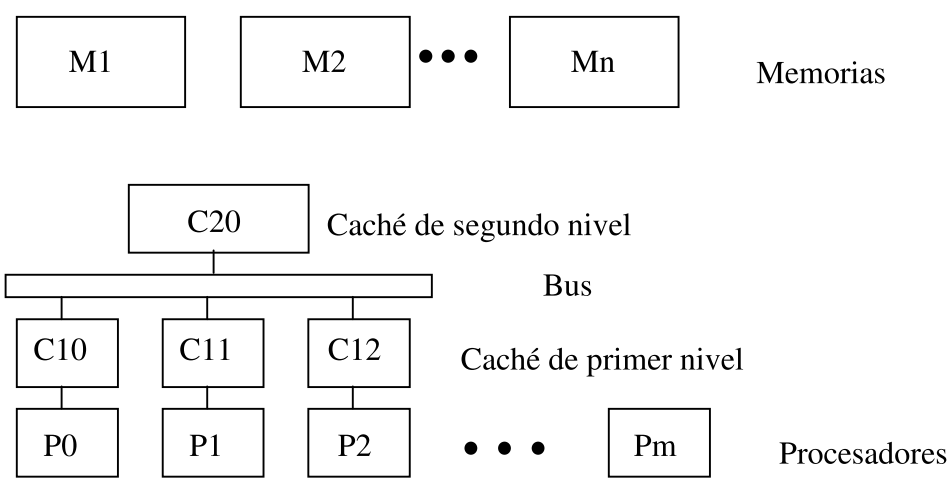
Las redes de conmutación de malla o punto de cruce son eficientes en este aspecto, pero muy costosas para números grandes de procesadores y/o memoria (n^2 cruces).

Una manera de reducir el número de conmutadores es la red omega, basada en arreglos de conmutadores 2X2, con 2 entradas y 2 salidas cada uno. Eligiendo los conmutadores adecuados, cualquier CPU puede accesar cualquier memoria ($n \log_2 n$ cruces)

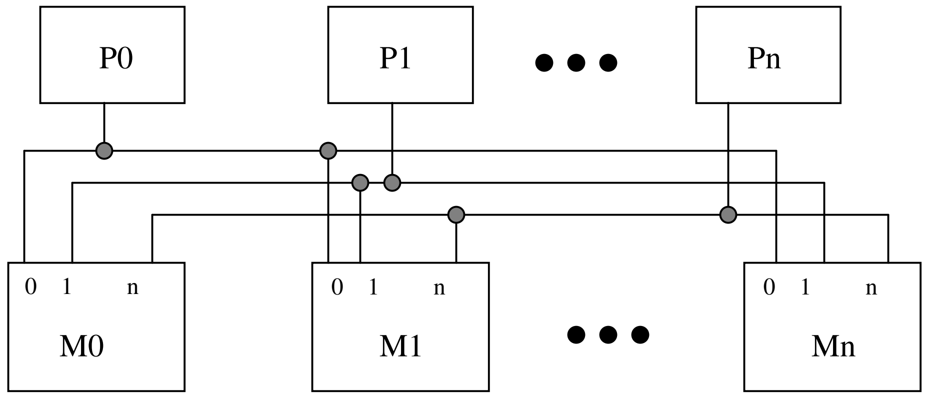
Acceso a memoria

Memoria Compartida

Se basa en el uso de memorias multipuerto y/o buses jerárquicos

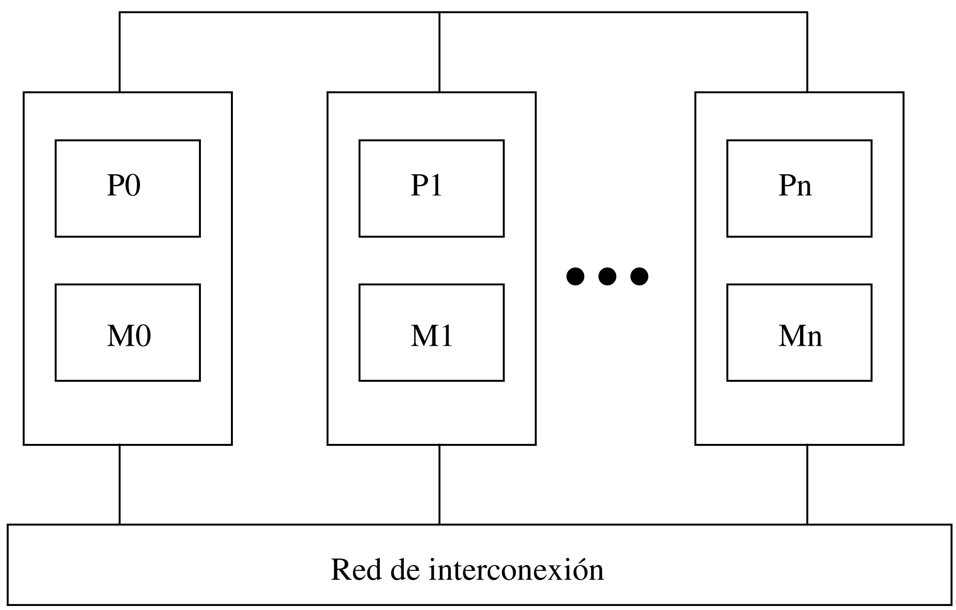


Buses jerárquicos



Memorias multipuerto

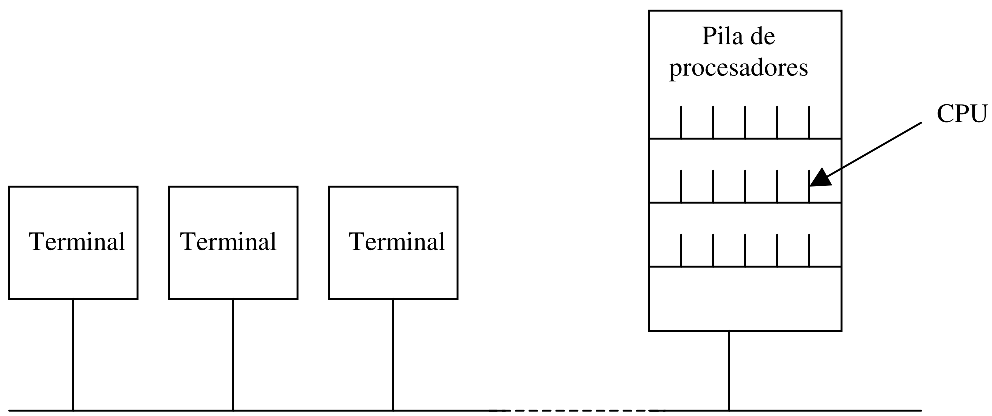
Memoria Distribuida



Administración de procesos paralelos

Asignación de procesadores

En general se utiliza el modelo de *pila de procesadores* para analizar la asignación de procesadores a los diferentes procesos



Modelo de Pila de procesadores

Utilizando elementos de teoría de colas se puede establecer el tiempo promedio entre la emisión de una solicitud de atención y la obtención de una respuesta completa:

$$T = \frac{1}{\mu - \lambda}$$

μ .- Promedio de procesamiento de solicitudes por unidad de tiempo
λ .- Promedio de solicitudes por unidad de tiempo

Podemos agrupar los diferentes esquemas de asignación de procesadores en dos categorías:

Asignación no migratoria.- Una vez decidido cuál procesador atenderá la solicitud, el proceso se ejecuta completo en este procesador. No puede moverse, no importa qué tan sobrecargado esté el procesador. Son sencillos, pero pueden ser ineficientes

Asignación migratoria.- El proceso puede iniciar su ejecución en un procesador, pero puede cambiarse a otro si la carga del procesador inicial es demasiada. Reparten mejor las cargas de trabajo pero son complejos y difíciles de implementar.

Diseño de algoritmos de asignación de procesadores

Las principales decisiones que deben tomar los diseñadores se resumen en cinco aspectos:

1. Algoritmos deterministas vs. heurísticos
2. Algoritmos centralizados vs. distribuidos
3. Algoritmos óptimos vs. subóptimos

4. Algoritmos locales vs. globales (política de transferencia: tomar el proceso o transferirlo)
5. Algoritmos iniciados por el emisor vs. iniciados por el receptor (política de localización: quién toma el proceso)

Ejemplos de algoritmos: Tanenbaum, S.O.D. p 203 – 210

Determimista con teoría de grafos: Trata de minimizar el tráfico entre procesadores agrupando procesos fuertemente acoplados en un CPU. Requiere info. Global del sistema

Heurístico centralizado (arriba-abajo):Un procesador principal mantiene una tabla que indica la carga de los procesadores

Jerárquico: (MICROS 1980) Etiqueta los procesadores para establecer jerarquías. Existe un coordinador o un comité de coordinadores que deciden las cargas de trabajo de sus subordinados. Los elementos pueden subir o bajar en la jerarquía de ser necesario. Es más escalable que los centralizados.

Heurístico distribuido iniciado por emisor: Un procesador muy cargado llama a otro al azar para transferirle un proceso. Si no tiene éxito en varios intentos lo ejecuta localmente. Existen versiones analíticas con teoría de colas.

Heurístico distribuido iniciado por receptor: Un procesador desocupado pregunta a otro al azar si tiene trabajo para él. S i no encuentra trabajo, deja de preguntar por cierto tiempo. Más eficiente por cargar menos al sistema.

Administración de recursos en un ambiente de multiprocesamiento