Cisneros Aguilar Sara Kenia
Moreno Diaz Rodrigo
Rios Lastiri Armando

Arquitecturas Paralelas

Computación Concurrente 2019/11/24

1. Introducción al Paralelismo

1.1. Procesamiento Paralelo

Es el uso de muchas unidades de proceso independientes para ejecutar distintas partes de una tarea en simultáneo y su principal objetivo es aumentar el **rendimiento**, y el aumento de la capacidad para resolver problemas computacionales grandes.
¿Cómo lo hace?

- División del trabajo en tareas mas pequeñas e independientes
- Asignación de las tareas a distintas unidades de proceso
- Resolución de las tareas en simultaneo.

¿Qué problemas se presentan?

- Sincronización de las tareas.
- Control de ejecución simultanea
- Conflictos debidos a dependencias

Presenta limitaciones, en algunos problemas el incremento del número de procesadores no mejora el rendimiento global, incluso empeora la eficiencia del sistema, esta eficiencia se mejora cuando se logra un balance de carga entre procesadores: igual número de tareas de igual tamaño y se minimiza la interacción entre tareas, se minimiza la comunicación o, al menos, se mejoran los canales de comunicación.

1.2. Sistema Paralelo

Un sistema paralelo es un conjunto de elementos de proceso que, operando juntos, permiten resolver problemas computacionales complejos de forma eficiente y cuenta con las siguientes características:

- Cantidad y potencia de los elementos de proceso
- Tipo y Tamaño de la memoria
- Forma de comunicación entre los elementos de proceso
- Rendimiento
- Escalabilidad del sistema: capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa la carga computacional del mismo.
- Recursos de potencia requeridos

El paralelismo puede estudiarse a varios niveles:

- Trabajo: Dos programas distintos pueden ejecutarse en paralelo
- Tarea: En este nivel se consideran varias tareas independientes entre si formando parte de un programa determinado. Es posible la interacción de las tareas
- Proceso: Varios procesos componen una tarea. Son bloques con funcionalidad bien definida.
- Variable: El paralelismo puede darse a nivel de variables ya que varias instrucciones pueden ser ejecutadas en paralelo siendo el punto de conflicto las variables en común
- Bit: Todos los computadores usan paralelismo a nivel de bit

1.3. Ventajas y limitaciones

Ventajas:

- Permite salvar el llamado *cuello de botella*: Si un proceso funciona lentamente, los demás deben esperar a que se termine para ejecutarse, con lo cual el rendimiento del ordenador se verá afectado en gran medida.
  Con un sistema en paralelo, aunque un proceso sea lento, el resto continúan ejecutándose.
- Resuelve problemas que no se podrían realizar en una sola CPU y en un tiempo razonable
- Permite ejecutar problemas de un orden y complejidad mayor de manera más rápida (aceleración)
- Permite la ejecución de varias instrucciones en simultáneo

Limitaciones:
El proceso en paralelo tiene ciertos inconvenientes, es mucho más complejo, básicamente porque, para que un sistema trabaje en paralelo se debe indicar dónde derivar cada proceso, y esto puede ralentizar el inicio de las tareas, al requerir cálculos previos.
Además algunas tareas pueden no ser divisibles, o el programa pudo haber sido diseñado para que los pasos se hagan uno a uno, con lo cual el procesamiento paralelo lejos de ayudar puede entorpecer la ejecución.

2. Medidas de performance

2.1. Speedup

Promedio entre el tiempo de proceso secuencial y paralelo en P procesadores

$$Sp = \frac{T1}{Tp}, Sp < P$$

Donde:

- Sp: Speedup para P procesadores
- P: número de procesadores
- Castilla la Mancha.
- $T1$: tiempo en 1 procesador
- Tp: tiempo en P procesadores

2.2. Eficiencia

Cociente entre Sp y P. Medida de la relación costo/efectividad de la computación

$$Ep = \frac{Sp}{P}, 0 < Ep < 1$$

Donde:

- Ep: Eficiencia para P procesadores
- Sp: Speedup con P procesadores
- P : número de procesadores

2.3. Redundancia

Promedio entre el número total de operaciones ejecutadas con P procesadores y el número de operaciones necesarias en 1 procesador

$$Rp = Op/O1 \frac{Op}{O1}, Rp > 1$$

Donde:

- Rp: Redundancia para P procesadores
- Op: Número de operaciones en P procesadores
- $O1$: Número de operaciones en un procesador

2.4. Utilización

Número de operaciones totales ejecutadas con P procesadores ponderada por la eficiencia de trabajo en esos P procesadores

$$Up = Rp(Ep) = \frac{Op}{P(Tp)}, Up < 1$$

Donde:

- Up: Utilización para P procesadores
- Rp: Redundancia para P procesadores
- Ep: Eficiencia para P procesadores
- Op: número de operaciones en P procesadores
- P : número de procesadores
- Tp: tiempo en P procesadores

2.5. Calidad del paralelismo (Qp)

La calidad de paralelismo es proporcional al Spedup y a la Eficiencia.La calidad de paralelismo decrece al aumentar la Redundancia.

$$Qp = \frac{Sp(Ep)}{Rp}, Qp < 1$$

Donde:

- Qp: Calidad del paralelismo
- Sp: Speedup con P procesadores
- Ep: Eficiencia para P procesadores
- Rp: Redundancia para P procesadores

2.6. Límites y costos de la programación paralela

La Ley de Amdahl establece que la aceleración (speedup) potencial del programa se define por la fracción de código (P) que se puede paralelizar:

$$speedup = \frac{1}{1-P}$$

Si ninguno de los códigos puede ser paralelo, $P = 0$ y la aceleración = 1 .
Si todo el código está en paralelo, $P = 1$ y la aceleración es infinita (en teoría).
Si se puede paralelizar el 50 % del código, la velocidad máxima = 2, lo que significa que el código se ejecutará el doble de rápido.

Al presentar el número de procesadores que realizan la fracción paralela de trabajo, la relación se puede modelar de la siguiente manera:

$$speedup = \frac{1}{\frac{P}{N}+S}$$

Donde:

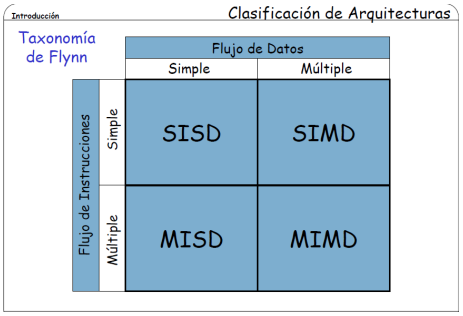
- P = fracción paralela
- N = número de procesadores
- S = fracción en serie

3. Arquitecturas Paralelas

Para poder hacer paralelismo se necesitan cierto tipo de arquitecturas, la primera clasificación fue la Taxonomía de Flynn. posteriormente se parte de esto para crear nuevas arquitecturas.

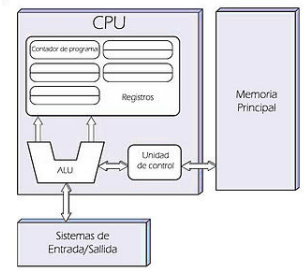
3.1. Taxonomía de Flynn

En 1966 Michael Flynn propuso un mecanismo de clasificación de las computadoras. El método de Flynn se basa en el número de instrucciones y de la secuencia de datos que la computadora utiliza para procesar información. Puede haber secuencias de instrucciones sencillas o múltiples y secuencias de datos sencillas o múltiples. Esto da lugar a 4 tipos de computadoras:

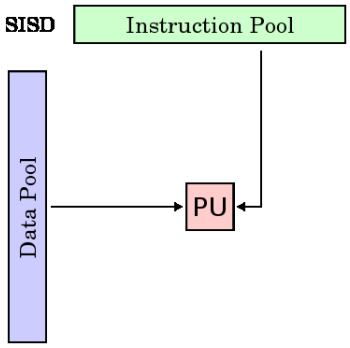


3.1.1. Arquitecturas SISD

Se refiere a las computadoras convencionales de Von Neuman. En la categoría SISD están la gran mayoría de las computadoras existentes. Son equipos con un solo procesador que trabaja sobre un solo dato a la vez. A estos equipos se les llama también computadoras secuenciales.



Todas las computadoras SISD utilizan un registro simple llamado *el contador del programa*, el cual lleva el conteo de la ejecución serial de las instrucciones. Como cada instrucción es fetcheada desde la memoria, el contador del programa es actualizado para direccionar a la siguiente instrucción que ha de ser fetcheada y ejecutada; lo que resulta ser una orden serial de ejecución.



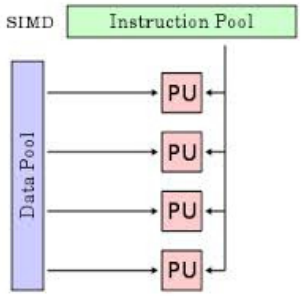
Cuenta con las siguientes características:

- Son equipos con un solo procesador, que trabaja sobre un solo dato a la vez.
- Flujo único de instrucciones.
- Flujo único de datos.
- Corresponde al modelo estructural básico, con un procesador de instrucciones y un procesador de datos
- Tiene una única vía de acceso a la memoria principal.

3.1.2. Arquitecturas SIMD

SIMD (Single instrucción múltiple data) permite efectuar varias operaciones de calculo con una sola instrucción.
Las computadoras SIMD tienen una sola unidad de control y múltiples unidades funcionales, la unidad de control se encarga de enviar la misma instrucción a todas las unidades funcionales y cada unidad funcional trabaja sobre datos diferentes. Estos equipos son de propósito específico, es decir, son apropiados para ciertas aplicaciones particulares, como por ejemplo el procesamiento de imágenes.
Esta arquitectura nace debido a la necesidad de aplicar repetidamente una misma operación en grupos datos diferentes como, Muestras contiguas de audio, matrices de vídeo, etc.
Características:

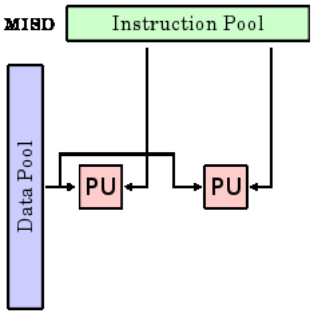
- Arquitectura empleada para conseguir paralelismo a nivel de datos.
- Un único programa controla los procesadores.
- Útil en aplicaciones uniformes
- Fueron los primeros multiprocesadores difundidos comercialmente (en la década de 1980)
- A los procesadores basados en esta arquitectura, se los conoce como procesadores matriciales.



Ejemplo de Procesador SIMD: el procesador Pentium MMX introdujo en la arquitectura IA32 un set de recursos para el tratamiento de señales.

3.1.3. Arquitecturas MISD

La arquitectura Pipeline la forma fundamental de ejecución paralela de un proceso y es una idea poderosa que puede probar de manera significativa el rendimiento de una computadora SIMD. Un procesador pipeline es un procesador MISD que trabaja de acorde al principio del funcionamiento de un Pipe.
Hay N secuencias de instrucciones(algoritmos/programas) y una secuencia de datos. El paralelismo es alcanzado dejando que los procesadores realicen diferentes cosas al mismo tiempo en el mismo dato.



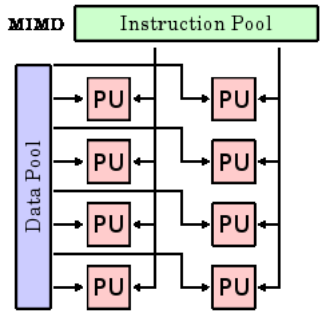
No existen muchos ejemplos de esta arquitectura, ya que MIMD y SIMD son a menudo más apropiados para técnicas comunes de datos paralelos. Específicamente, permiten un mejor escalamiento y uso de recursos computacionales que el MISD. Sin embargo, un ejemplo prominente de MISD en la informática son las computadoras de control de vuelo del transbordador espacial.

3.1.4. Arquitecturas MIMD

Un sistema MIMD es un sistema multiprocesador o una multicomputadora en donde cada procesador individual tiene su unidad de control y ejecuta su propio programa. Las computadoras MIMD tienen las siguientes características:

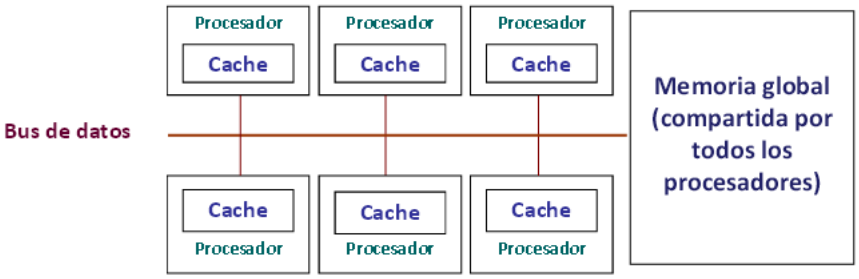
- Distribuyen el procesamiento sobre un número independiente de procesadores.
- Comparten fuentes, incluyendo el sistema de memoria principal, sobre los procesadores.

- Cada procesador opera independientemente y concurrentemente.
- Cada procesador corre su propio programa. Esto indica que los sistemas MIMD ejecutan operaciones en paralelo de manera asíncrona; los nodos activos cooperan pero operan independientemente.



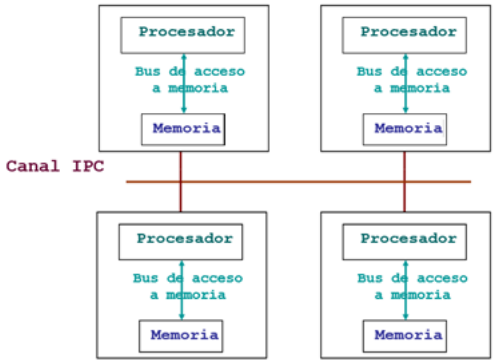
Ejemplos de computadoras MIMD incluyen la Cosmic Cube, nCUBE2, iPSC, Symmetry, FX-8, FX-2800, TC-2000, CM-5, KSR-1 y la Paragon XP/s [Roosta99].
Las computadoras MIMD se pueden categorizar en sistemas fuertemente acoplados y sistemas débilmente acoplados; dependiendo de cómo los procesadores accedan a la memoria. Las computadoras fuertemente acopladas y débilmente acopladas corresponden a los sistemas MIMD de Memoria Global (GM-MIMD) y MIMD de Memoria Local (LM-MIMD) respectivamente.
Las computadoras MIMD de paso de mensajes se refieren a multicomputadoras en donde cada procesador tiene su propia memoria, llamada memoria local o privada, y es accesible sólo por su propio procesador. Las arquitecturas MIMD de paso de mensajes están referidas tanto a arquitecturas de memoria distribuida como a arquitecturas de memoria privada.
Un sistema MIMD de memoria compartida es llamado Sistema de Acceso Uniforme a Memoria (UMA, Uniform Memory Access), ya que el tiempo de acceso a memoria es el mismo para todos los procesadores que la comparten. En este modelo, los procesadores pueden comunicarse sin restricciones y de una manera simple compartiendo datos utilizando un mismo espacio de direccionamiento. Los datos que se comparten pueden protegerse mediante el uso de métodos de ocultamiento de datos que los modernos lenguajes de programación ofrecen.
Debido a esta capacidad de soportar una gran variedad de modelos de programación eficientes, un sistema multiprocesador de memoria compartida es siempre la primera elección de los usuarios de programación paralela. Esto entra en contraste con los sistemas de paso de mensajes que son más fáciles de diseñar pero más difíciles de implementar o programar.
En general, los sistemas MIMD fuertemente acoplados proporcionan mayor rapidez en el intercambio de datos entre los procesadores que los sistemas MIMD débilmente acoplados. Ejemplos de computadoras GM-MIMD son la serie CDC 6600 y la Cray XM-P. Ejemplos de computadoras LM-MIMD son la Carnegie-Mellon Cm* y la Tandom/16 [Roosta99].
Como se pudo ver hay varios tipos de sistemas MIMD, en resumen, se clasifican en:

- Sistemas de Memoria Compartida: En este tipo de sistemas cada procesador tiene acceso a toda la memoria, es decir hay un espacio de direccionamiento compartido. Las computadoras MIMD con memoria compartida son sistemas conocidos como de multiprocesamiento simétrico (SMP) donde múltiples procesadores comparten un mismo sistema operativo y memoria.



Arquitectura MIMD con memoria compartida.

- Sistemas de Memoria Distribuida: Estos sistemas tienen su propia memoria local. Los procesadores pueden compartir información solamente enviando mensajes. Las computadoras MIMD de memoria distribuida son conocidas como sistemas de procesamiento en paralelo masivo (MPP) donde múltiples procesadores trabajan en diferentes partes de un programa, usando su propio sistema operativo y memoria.



- Sistemas de Memoria Compartida Distribuida: Es una partición de procesadores que tienen acceso a una memoria compartida común pero sin un canal compartido. Esto es, físicamente cada procesador posee su memoria local y se interconecta con otros procesadores por medio de un dispositivo de alta velocidad, y todos ven las memorias de cada uno como un espacio de direcciones globales.

4. UMA Uniform memory access

En este tipo de arquitectura, como bien dice su nombre, todos los accesos a memoria tardan el mismo tiempo. Seguramente podemos pensar que es difícil que tengamos el mismo tiempo de acceso si la memoria, aunque compartida, está dividida en módulos a los que se accede a través de una red de interconexión basada en switches. Eso es cierto y, para conseguir esta uniformidad de acceso, se tiene que aumentar el tiempo de los accesos más rápidos.

¿Por qué se busca esta uniformidad de tiempo de acceso? Porque para los programadores es más fácil deducir qué parámetros ayudarán a mejorar sus programas si el tiempo de acceso es igual para cualquier acceso. De lo contrario, como pasa con las arquitecturas NUMA, deberían ser mucho más conscientes de la arquitectura que tiene el computador y de cómo están distribuidos los datos en la arquitectura en cuestión.

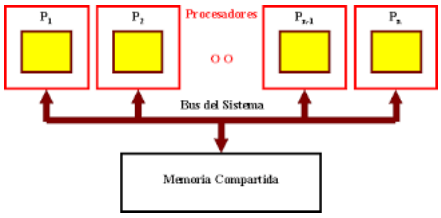
5. NUMA Non-uniform memory access

En los multiprocesadores NUMA, a diferencia de los UMA, los accesos a memoria pueden tener tiempos distintos. En estas máquinas la memoria también está compartida, pero los módulos de memoria están distribuidos entre los diferentes procesadores con el objetivo de reducir la contención de acceso a memoria.
Es de acceso a la memoria no uniforme debido a que cada procesador tiene su propia memoria local pero puede también tener acceso a la memoria de otros procesadores y un procesador puede acceder su propia memoria local más rápido que la memoria no local (memoria que está en otro procesador o compartida entre procesadores), y se divide en:

5.1. SMP Multiprocesamiento simétrico

Permite que cualquier procesador trabaje en cualquier tarea sin importar su localización en memoria; con un propicio soporte del sistema operativo. Una computadora SMP se compone de microprocesadores independientes que se comunican con la memoria a través de un bus compartido. Carácteristicas:

- Paralelismo sobre memoria compartida. Varias unidades de procesamiento comparten el acceso a la memoria, compitiendo en igualdad de condiciones por dicho acceso.
- En esta clasificación entran los modelos SIMD MIMD UMA (Uniform Memory Access).



5.2. MPP Procesamiento paralelo masivo

Se refiere a muchas CPUs separadas corriendo en paralelo para procesar un solo programa y son difíciles de programar porque las aplicaciones se deben dividir en tal manera que todos los segmentos que se ejecutan se puedan comunicar unos con otros. Además no tiene problemas de cuello de botella inherente en los sistemas SMP cuando todas las CPUs intentan acceder la misma memoria al mismo tiempo.

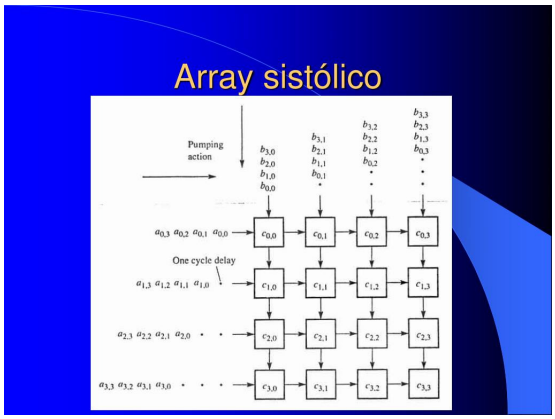
6. COMA Cache only memory access

Las arquitecturas COMA (cache only memory access) intentan solucionar este problema haciendo que la memorial local a cada procesador se convierta en parte de una memoria caché grande. En este tipo de arquitecturas, las páginas de memoria desaparecen y todos los datos se tratan como líneas de caché. Sin embargo, con esta estrategia surgen nuevos interrogantes: ¿cómo se localizan las líneas de caché? y cuando una línea se tiene que reemplazar, ¿qué sucede si esta es la última? Estas preguntas no tienen fácil solución y pueden requerir de soporte hardware adicional. Normalmente se implementan utilizando un mecanismo de directorio distribuido.
Así, aunque este tipo de máquinas prometían mejor rendimiento que las ccNUMA, se han construido pocas. Las primeras COMA que se construyeron fueron la KSR-1 y la *data diffusion machine*.

7. Otras Arquitecturas

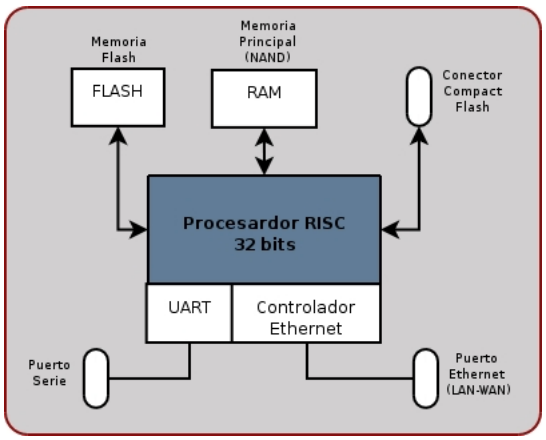
7.1. Arrays Sistólicos

Arquitectura y paradigma de la computación en paralelo. H. T. Kung y C. E. Leiserson introdujeron en 1978 el término "sistólico.[en] los sistemas de propósito específico que deben balancear el ancho de banda entre una gran cantidad de cálculos y unos altos requisitos de I/O. La metáfora del flujo de sangre ha servido para caracterizar el flujo de datos a lo largo del proceso, así, el corazón entrega y recibe una gran cantidad de sangre como resultado del bombeo rítmico e ininterrumpido de pequeñas cantidades de ese fluido a través de venas y arterias. En esta analogía el corazón corresponde a la fuente y destino de los datos, como si fuera una memoria global y la red de venas es equivalente a la matriz de procesadores y sus conexiones.
Las arquitecturas sistólicas son multiprocesadores interconectados en los cuales los datos se bombean en forma rítmica desde la memoria y a través de la red de procesadores antes de ser devueltos a la memoria. La información circula entre los procesadores como en una tubería, pero sólo los procesadores frontera mantienen comunicación con el exterior. Un reloj global conjuntamente con mecanismos explícitos de retardo sincronizan el flujo de datos a través de las conexiones.



7.2. Procesadores RISC

RISC es el acrónimo de Reduced Instruction Set Computer lo que en castellano viene a ser conjunto reducido de instrucciones de computadora. Se entiende por procesador RISC aquel que tiene un conjunto de instrucciones con unas características determinadas.
El término *reducido* puede llevar a engaño cuando nos referimos a RISC. No se trata de que sean pocas instrucciones, ya que ningún procesador actual tiene pocas, si no de que estas sean sencillas. Se acepta que un procesador sea RISC cuando la misma instrucción que carga datos de memoria no realiza operaciones sobre ellos. Es necesario esperar a que otra realice el tratamiento de esos datos.
Gracias a esto la unidad de control, que es la encargada de gestionar que los bloques funcionales como la unidad aritmética lógica o la de punto flotante realicen su función, puede ser más sencilla que con otras arquitecturas. Esto permite obtener más espacio dentro del propio chip para otros elementos. Se suelen añadir así una mayor cantidad de registros que permiten tener más datos de forma interna en la CPU lo cual lleva a trabajar en ciertas ocasiones de forma más eficiente.



7.3. Procesadores CISC

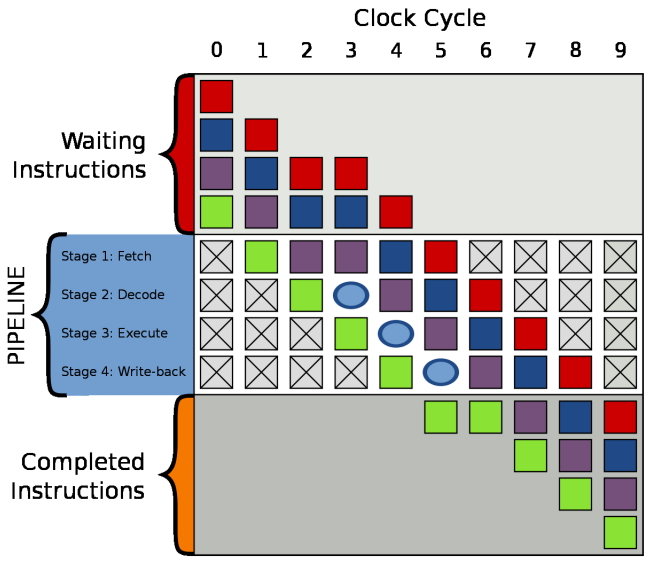
Complex Instruction Set Computer (CISC). En español (Computadora de Conjunto de Instrucciones Complejas). En ella el procesador trae cientos de registros y se necesitan muchos pasos y ciclos de reloj

para realizar una sola operación. Todos los CPUs x86 compatibles con la PC son procesadores CISC, pero en las Mac nuevas o en algunas que se hagan dibujos de ingeniería complejos, probablemente tengan un CPU RISC (Computadora de Conjunto de Instrucciones Reducido)

RISC	CISC
Instrucciones simples (una por ciclo)	Instrucciones complejas
Instrucciones de longitud fija	Instrucciones de longitud variable
Opera a altas velocidades	Opera a bajas velocidades
Usa 138 registros de 32 y 64 Bits	Usa muy pocos registros
Instrucciones implementadas por hardware	Instrucciones implementadas por software

7.4. Pipeline

La arquitectura Pipeline consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior, con un almacenamiento temporal de datos entre procesos.
Entender cómo funciona un pipeline es un paso importante para entender qué ocurre dentro de un procesador. Este sistema es común verlo en sistemas operativos multitarea ya que puede ejecutar una serie de procesos de manera simultánea, los cuales son ejecutados de manera secuencial mediante un administrador de tareas que aplica distintos tipos de prioridad y capacidad de procesamiento. Aquí se alterna entre este sistema (el de tuberías) y los demás.

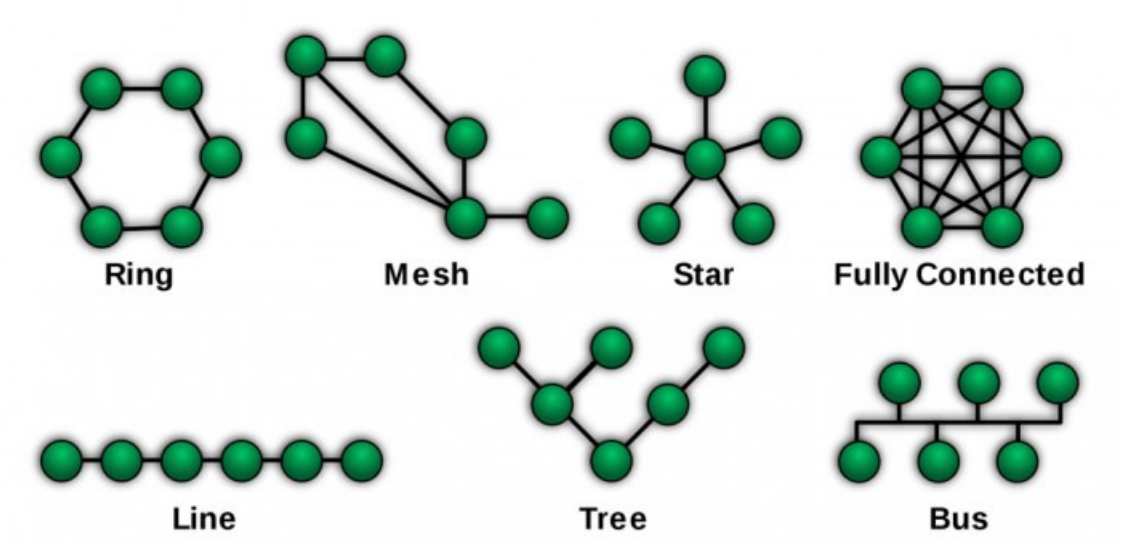


8. Conectividad entre procesadores

- Estática:
  - Usualmente canales de comunicación P2P entre procesadores.
  - Caminos prefijados entre procesadores.
- Dinámica:
  - Los caminos se determinan dinámicamente.
  - Se implementa a través de switches.
  - Simplifica la programación al evitar problemas de comunicación.
  - Garantiza igualdad de latencia para comunicaciones entre distintos procesadores a una distancia fija. Permite comunicaciones "all to all".
  - Generan topologías fácilmente escalables.

9. Topología de Red

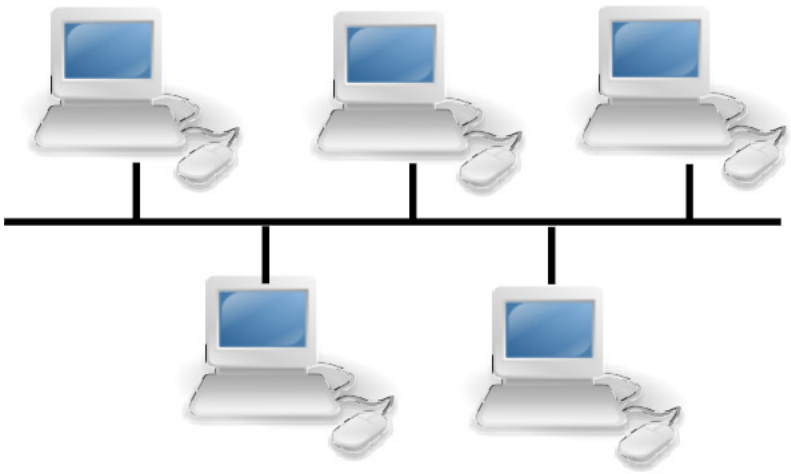
Uno de los problemas que se presenta en el paralelismo radica en el intercambio de información. Si no se cuenta con una buena conectividad entre los equipos el retardo que causa el envío de la información puede afectar el rendimiento del sistema.
La topología de red es la disposición física en la que se conectan los nodos de una red de ordenadores o servidores. Estos computadores pueden conectarse de muchas y muy variadas maneras. Las mas comunes son:



9.1. Topología de Bus

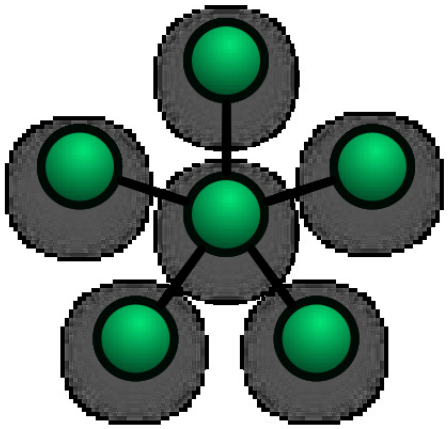
La topología de Bus se basa en un cable central, el cual lleva la información a todas las computadoras de la red, en forma de ramificaciones, de modo, que la información viaja de manera secuencial hacia

los nodos de la red. Su desventaja se basa en su distribución secuencial de datos, por lo que si se interrumpe el cable central, la red queda inutilizada. En la actualidad es muy poco utilizada.



9.2. Topología de Estrella

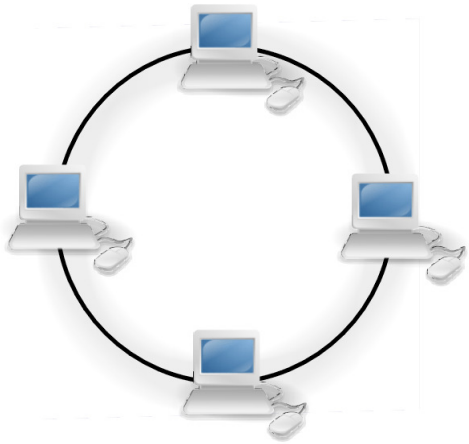
La distribución de la información va desde un punto central o Host, hacia todos los destinos o nodos de la red. En la actualidad, es muy utilizada por su eficiencia y simpleza. Se puede notar que el Host realiza todo el trabajo (una especie de servidor local que administra los servicios compartidos y la información). Por supuesto, cuenta con la ventaja que si un nodo falla, la red continuará trabajando sin inconveniente, aunque depende del funcionamiento del Host.



9.3. Topología de Anillo

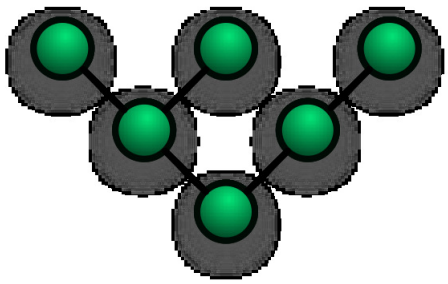
Es un tipo de topología de red simple, en donde las estaciones de trabajo o computadoras, se encuentran conectadas entre sí en forma de un anillo, es decir, forman un círculo entre ellas. La información viaja

en un solo sentido, por lo tanto, que si un nodo deja de funcionar se cae la red o deja de abastecer información a las demás computadoras que se encuentran dentro del anillo, por lo tanto, es poco eficaz.



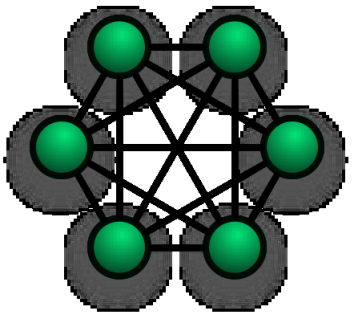
9.4. Topología de Árbol

Como su nombre lo indica, las conexiones entre los nodos (terminales o computadoras) están dispuestas en forma de árbol, con una punta y una base. Es similar a la topología de estrella y se basa directamente en la topología de bus. Si un nodo falla, no se presentan problemas entre los nodos subsiguientes. Cuenta con un cable principal llamado Backbone, que lleva la comunicación a todos los nodos de la red, compartiendo un mismo canal de comunicación.



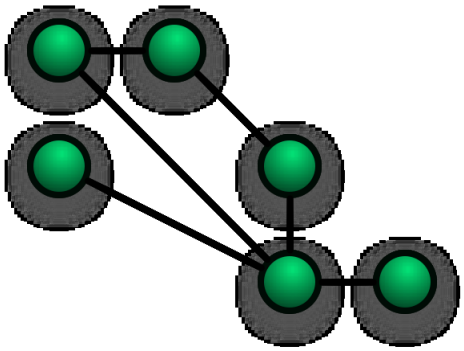
9.5. Topología de Malla

Se trata de un arreglo de interconexión de nodos (terminales) entre sí, realizando la figura de una malla o trama. Es una topología muy utilizada entre las redes WAN o de área amplia. Su importancia radica en que la información puede viajar en diferentes caminos, de manera que si llegara a fallar un nodo, se puede seguir intercambiando información sin inconveniente alguno entre los nodos.



9.6. Topología Híbrida

Es una combinación de dos o más topologías de red diferentes, para adaptar la red a las necesidades del cliente. De este modo, podemos combinar las topologías que deseemos, obteniendo diferentes variedades las cuales deben ajustarse a la estructura física del lugar en donde estará la red y los equipos que estarán conectados en dicha red.



10. Factores que determinan la eficiencia

- Ancho de banda: Número de bits capaces de transmitirse por unidad de tiempo.
- Latencia de la red: Tiempo que toma a un mensaje transmitirse a través de la red.
- Latencia de las comunicaciones: Incluye tiempos de trabajo del software y retardo de la interfaz.
- Latencia del mensaje: Tiempo que toma enviar un mensaje de longitud cero.
- Valencia de un nodo: Número de canales convergentes a un nodo.
- Diámetro de la red: Número mínimo de saltos entre los nodos más alejados. Permite calcular el peor caso de retardo de un mensaje.
- Largo máximo de un tramo de comunicación.
- Costo: Cantidad de enlaces de comunicación.

10.1. Configuración óptima

- Ancho de banda grande.
- Latencias (de red, comunicación y mensaje) bajas.
- Diámetro de la red reducido.
- Ancho de bisección grande.
- Valencia constante e independiente del tamaño de la red.
- Largo máximo de tramo reducido, constante e independiente del tamaño de la red.
- Costo mínimo.