Sistemas Operativos

Multiprocesadores - I

Sistemas Operativos

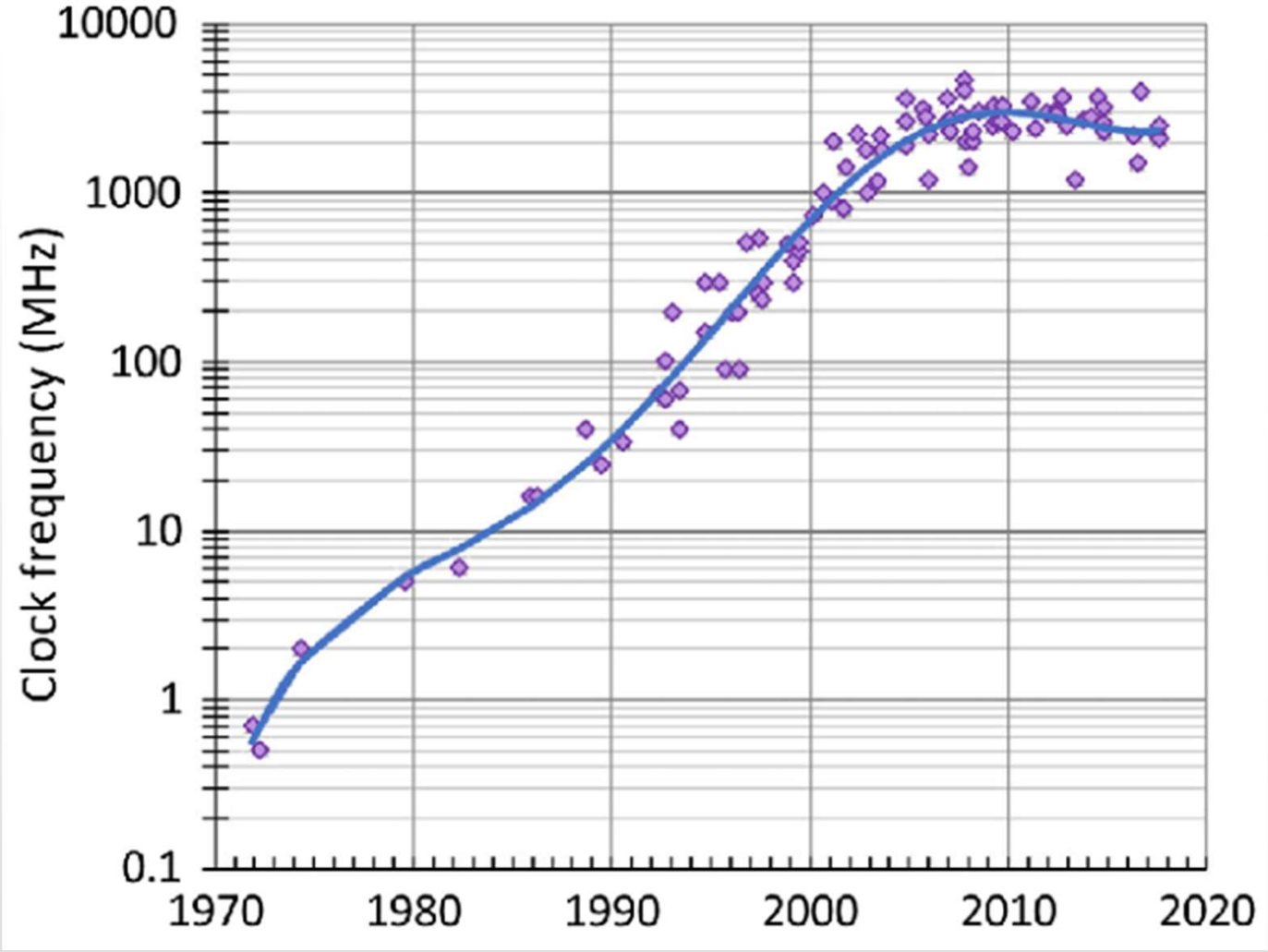
- ☑ Versión: Mayo 2025
- ☑ Palabras Claves: Multiprocesadores, SMP, Redes, Distribuidos, UMA, NUMA

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) , el de Silberschatz (Operating Systems Concepts)

Origen

- Desde su inicio, la industria de las computadoras se ha orientado fundamentalmente a buscar un poder de cómputo cada vez mayor.
- Las necesidades actuales demandan cada vez mayor poder de cómputo (física, astronomía, biología, calculo de modelos, IA)
- En el pasado, la solución era siempre hacer que el reloj operara a mayor velocidad

Origen



https://www.researchgate.net/figure/Microprocessor-clock-frequency-data-14-and-trend_fig3_338517514

Origen

- En la actualidad, lograr mayor velocidad es físicamente complejo:
  - Ninguna señal eléctrica se puede propagar más rápido que la velocidad de la luz
  - Problemas de disipación de calor (muchos transistores juntos en poco espacio)
  - Problemas de consumo eléctrico
- La solución al problema es el cómputo en paralelo y/o distribuido
- Contar con varias CPU que operen a velocidad "normal" y que en conjunto provean la potencia de cómputo necesaria

Origen – Cuestiones Físicas

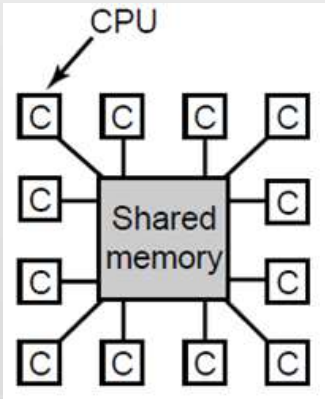
- ☑ Teoría de la relatividad de Einstein → ninguna señal eléctrica se puede propagar más rápido que la velocidad de la luz: 20 cm/nseg en cobre o FO
- ☑ En una CPU con un reloj de 10 GHz, las señales no deberían viajar más de 2 cm en total.
- ☑ En una CPU de 100 GHz la longitud total de la ruta sería cuando mucho de 2 mm.
- ☑ Una CPU de 1 THz (1000 GHz) tendría que ser más pequeña que 100 microns
- ☑ Problema fundamental: la disipación del calor.

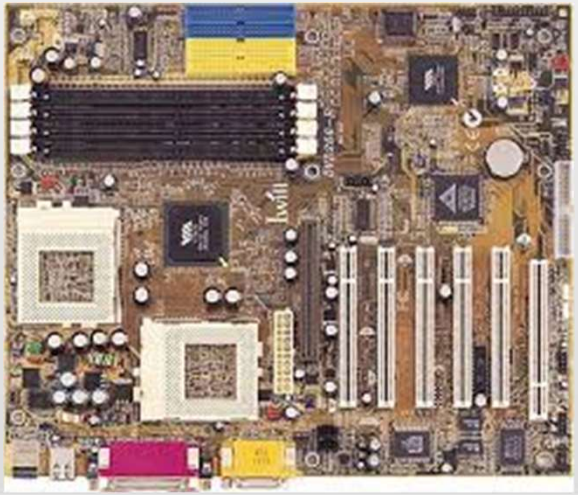
Esquemas

- Si tenemos que resolver un problema en una única CPU, el esquema de trabajo es sencillo
- Si tenemos varios problemas y varias CPU, a priori podríamos asignar estáticamente una tarea a cada una de ellas: ← no es lo mas eficiente
  - Deberá existir un coordinador que se encargue de repartir las tareas
- Al existir múltiples CPU, la complejidad aumenta en lo que refiere a distribución de tareas, pasaje de mensajes y acceso a memoria:

Esquemas

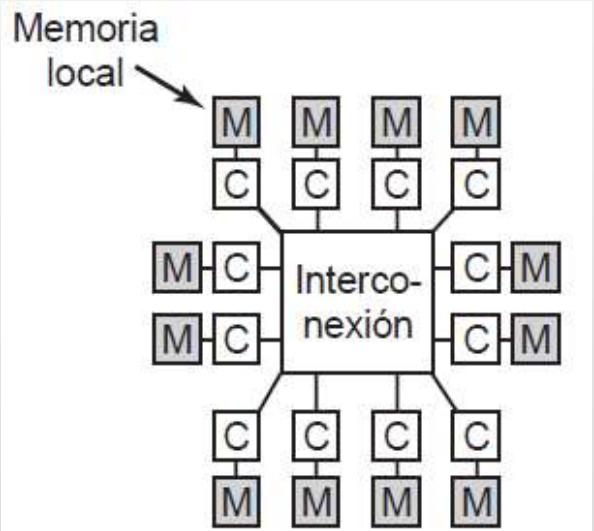
- **1. Multiprocesadores con Memoria Compartida**
  - La comunicación entre las CPU es a través de la memoria compartida
  - Cada CPU tiene el mismo acceso que otras a la memoria física a través de un único BUS físico
  - Para acceder a una palabra de memoria por lo general cada CPU requiere de 2 a 10 nseg.
  - Existe un único espacio lógico de direcciones para todos los procesos





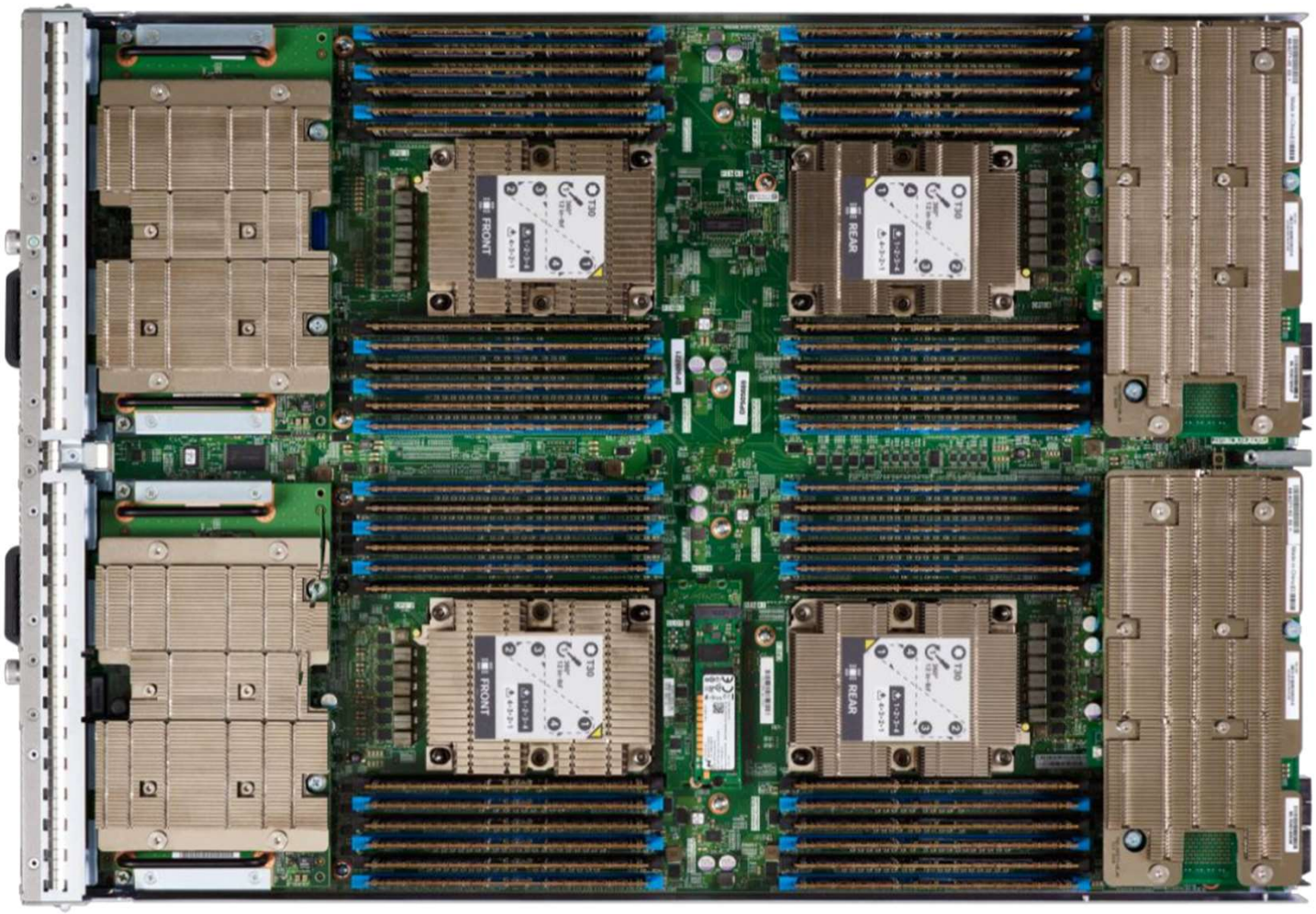
Esquemas

- **2. Multicomputadora con memoria independiente / pasaje de mensajes:**
  - Varios pares de CPU-memoria se conectan a una interconexión de alta velocidad pasando mensajes
  - Cada memoria es local para una sola CPU y puede ser utilizada sólo por esa CPU
  - El retardo del paso de mensajes es de entre 10 a 50 µseg



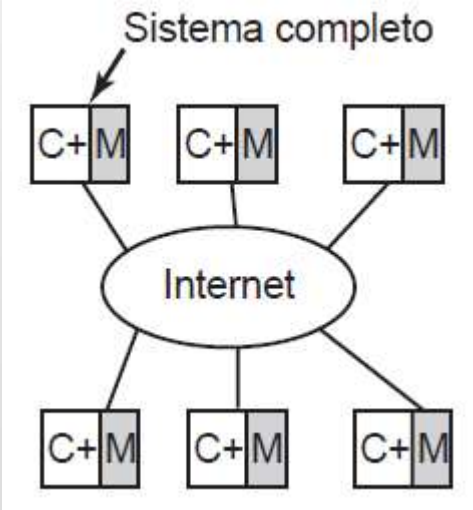
Esquemas

- ☑ 2 [illegible] de [illegible]



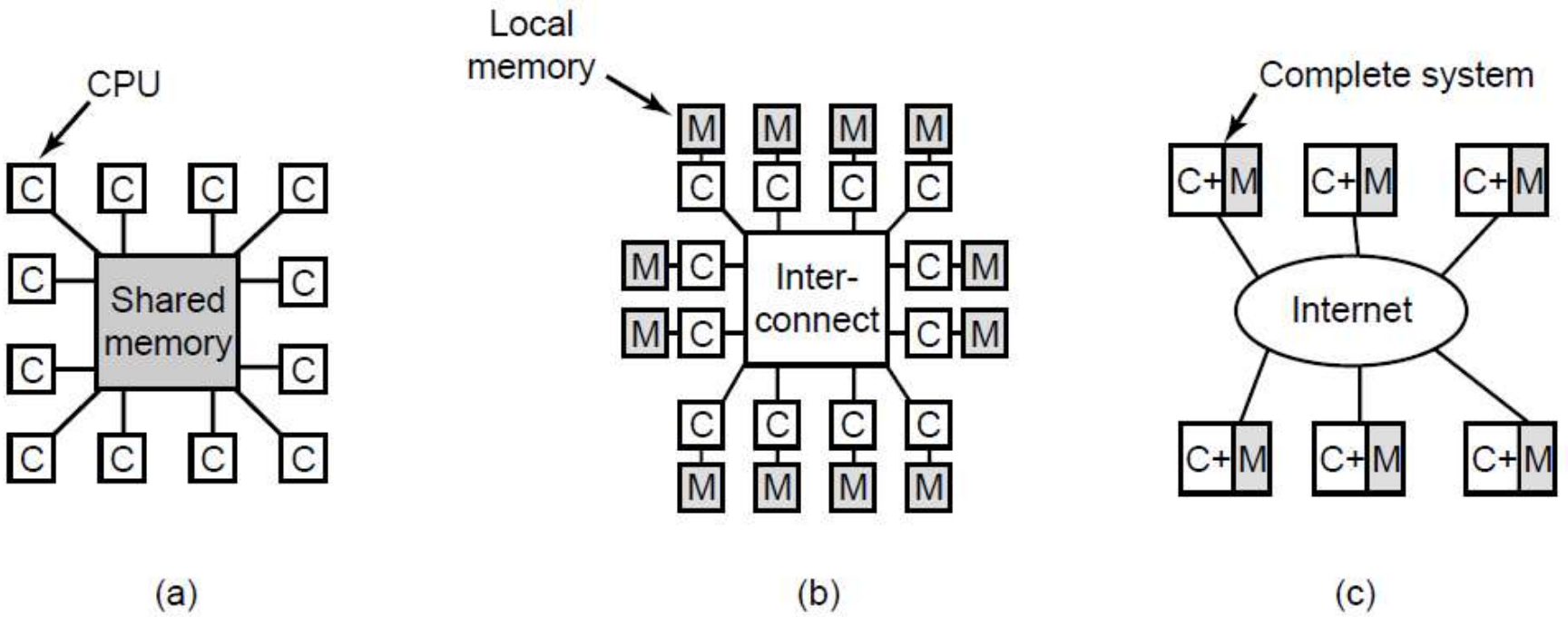
Esquemas

- **3. Sistemas Distribuidos:**
  - Conecta sistemas de cómputo completos a través de una red
  - Cada sistema tiene su propia memoria, y se comunican mediante el paso de mensajes
  - El retardo del paso de mensajes es de entre 10 a 100 mseg
  - Cada nodo de cómputo es una computadora completa
  - Heterogeneidad de sistemas y hardware



Esquemas (cont.)

2 a 10 nseg *10000 a 50000 nseg* *10 a 100 mseg*



Diferencias en las velocidades de comunicación

Chips Multinúcleo

- A medida que la tecnología avanza, los transistores se hacen mas pequeños y aparece la posibilidad de agregar más de uno a un chip.
- Al tener mas transistores se puede:
  - Agregar más memoria cache → Está demostrado que la tasa de aciertos no se incrementa demasiado
  - Agregar mas velocidad de clock a una CPU → Sigue existiendo un único hilo de ejecución
  - Agregar más CPU al mismo chip (núcleos), los que podrían compartir la cache y la memoria principal → Se logra paralelismo
- Es importante que el software se diseñe teniendo en cuenta los aspectos del hardware para aprovecharlo al máximo

Tipos de SO Multiprocesador

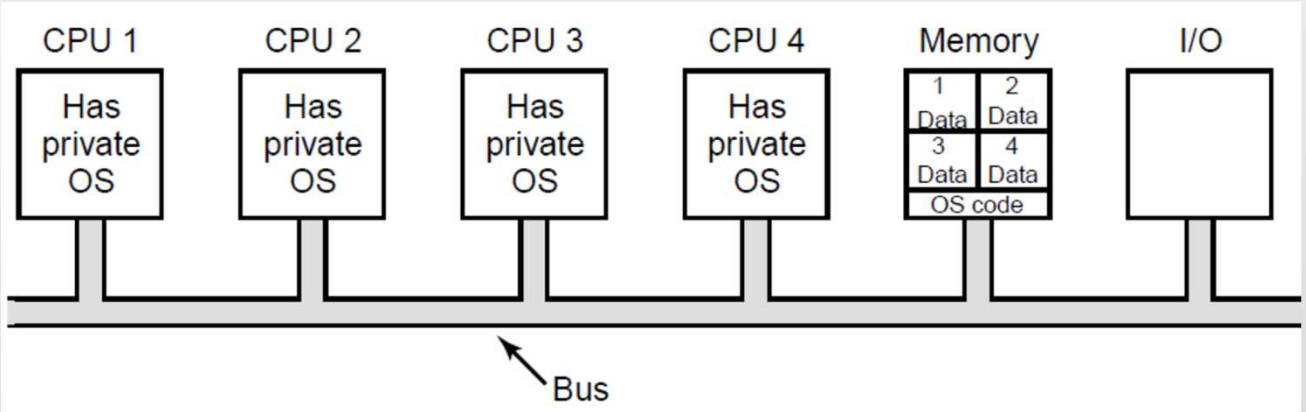
- Hasta el momento hicimos un análisis del Hardware Multiprocesador
- Analicemos como el hardware es manejado por los distintos Sistemas Operativos (**software**)
- Existen diversas metodologías posibles para la administración en esquemas multiprocesadores desde el lado del **software**:

Responsabilidades del Sistemas Operativo

- ☑ Por lo general, los sistemas operativos realizan tareas regulares como lo son:
  - ☑ Manejo de System Calls
  - ☑ Administración de Memoria
  - ☑ Administración de E/S (sist. de archivos y dispositivos)
- ☑ Algunas características Particulares en sistemas Multiprocesadores:
  - ☑ Sincronización de procesos
  - ☑ Administración de Recursos
  - ☑ Planificación de CPU
- ☑ Todas estas características se ven afectadas en sistemas Multiprocesadores. Hay que atacar las problemáticas que pueden ocurrir

Tipos de SO Multiprocesador

1. **Cada CPU con su SO (modelo poco utilizado):**
   - Se divide estáticamente la memoria para cada CPU con su copia privada (las CPUs operan independientes)
   - Se comparte el código de SO
   - Cada CPU cuenta con su propio conjunto de procesos
     - Desbalance en la carga de trabajo
   - Hay planificación independiente por CPU



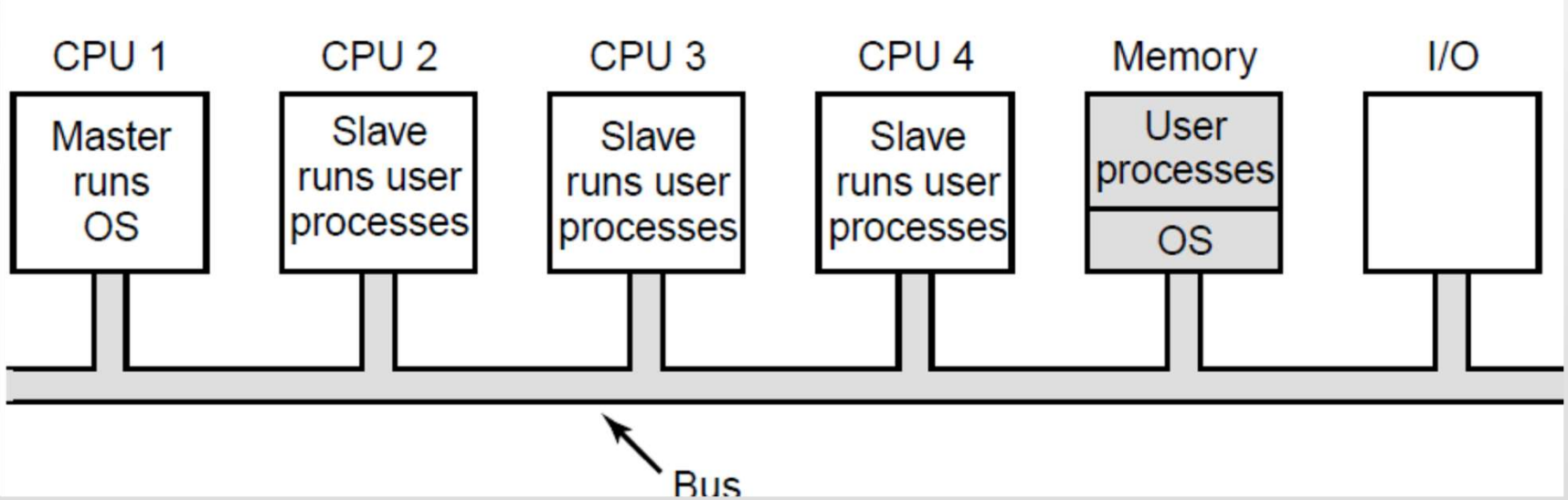
Tipos de SO Multiprocesador

- Existen diversas metodologías posibles para la administración en esquemas multiprocesadores desde el lado del software:

1. **Cada CPU con su SO:**
   - Cada CPU atrapa y maneja las SysCalls de sus procesos
   - Los procesos quedan "atados" a una única CPU
   - No se pueden compartir paginas
     - Pasaje de mensajes
     - Memoria desperdiciada
   - Cache de disco, cada CPU tiene su propia copia
     - Inconsistencia de la información

2. Maestro – Esclavo

- Única copia del SO y de su información
- Todas las SysCalls se redirigen a una CPU
- Está CPU puede ejecutar procesos si "le sobra tiempo"

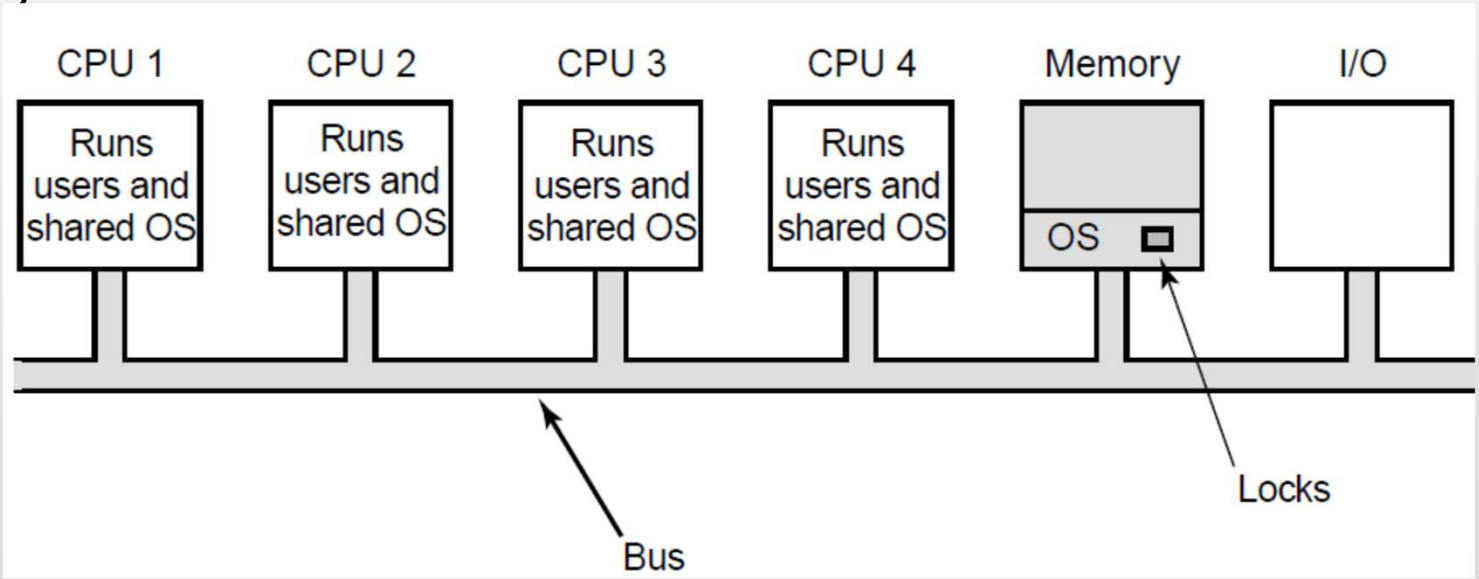


2. Maestro – Esclavo

- Resuelve problemas del modelo anterior
  - Se mantiene una única cola de planificación
  - Cuando una CPU está libre, pide un proceso al master
  - Se pueden asignar las páginas entre todos los procesos de manera dinámica.
- Problema, con muchas CPUs hay un cuello de botella en el Maestro
  - Ej: Si el 10% del tiempo se atienden SysCalls, con 10 CPU el master se saturaría, con 11 estaría sobrecargado

3. SMP – Multiprocesadores Simétricos

- Soluciona el inconveniente de saturación de una única CPU
- Única copia del SO en memoria y cualquier CPU puede ejecutarlo
- Cuando se invoca una System Call, es ejecutada por la CPU que la invocó



3. SMP – Multiprocesadores Simétricos (cont.)

- Provee equilibro entre procesos y memoria, ya que solo hay un único conjunto de tablas del SO
- No hay cuello de botella, ya que no hay una CPU master
- Problemas
  - **Dos o mas CPUs ejecutando código del SO en un mismo instante de tiempo**
  - **Dos CPUs seleccionando el mismo proceso para ejecutar, o seleccionan la misma página de la memoria libre!**

3. SMP – Multiprocesadores Simétricos (cont.)

- Posibles soluciones a los problemas planteados:
  1. **Utilizar "locks" para las estructuras del SO:**
     - Considerar a todo el SO como una gran sección critica. Cualquier CPU puede ejecutar el SO, pero solo una a la vez.
     - Se comportaría como el modelo maestro-esclavo
     - Es un modelo poco utilizado, debido a la mala performance que provee

3. **SMP – Multiprocesadores Simétricos (cont.)**

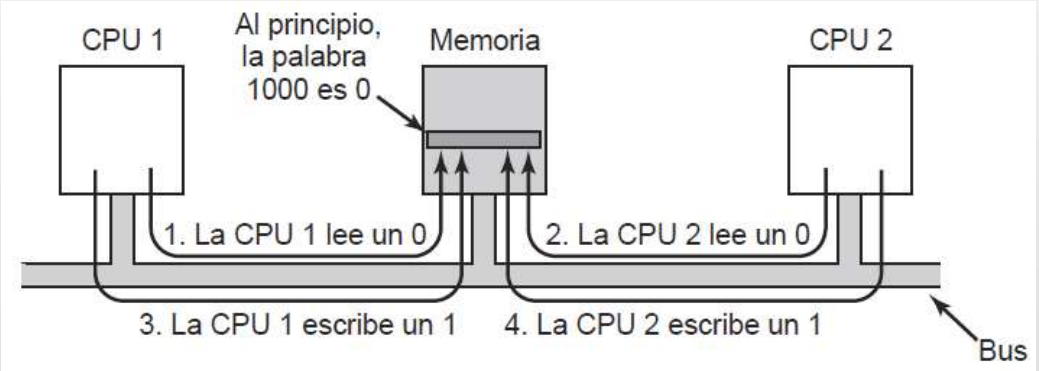
- Posibles soluciones a los problemas planteados:
  2. **Lock por estructura(s):**
     - Existen varias secciones criticas independientes, cada una protegida por su propio mutex
     - Mejora el rendimiento
     - Dificultad para determinar cada sección critica
     - **Ciertas estructuras pueden pertenecer a más de una sección crítica**, lo cual ante bloqueos podría generar **Deadlocks**
     - Es el esquema que generalmente se utiliza

Sincronización de Multiprocesadores

- ☑ Es necesario que las CPU de un multiprocesador se encuentren sincronizadas (acceso a regiones críticas, estructuras, etc.).
- ☑ En entornos uniprocesador si un proceso realiza una llamada al sistema que requiera acceder a cierta tabla crítica del kernel, el código del kernel sólo tiene que deshabilitar las interrupciones antes de tocar la tabla.
- ☑ En sistemas multiprocesadores, se deshabilitan las interrupciones de una CPU, pero otra podría generarlas…
- ☑ Surge la necesidad de contar con un protocolo de mutex apropiado para garantizar la exclusión mutua.

Sincronización de Multiprocesadores

- ✓ Una posibilidad para garantizar la exclusión mutua es el uso de TSL (Probar y establecer bloqueo):
  - ✓ Lee la palabra de memoria y la almacena en un registro. Al mismo tiempo escribe un 1 en la memoria para hacer el lock (2 accesos al BUS). Cuando termina libera (escribe 0). En uniprocesadores esta implementación es correcta.
  - ✓ El problema surge en entornos multiprocesadores: (la operación no es indivisible



  - ✓ Ambas CPU obtuvieron un 0 de la instrucción TSL, por lo que ambas tienen acceso a la sección crítica

Sincronización de Multiprocesadores

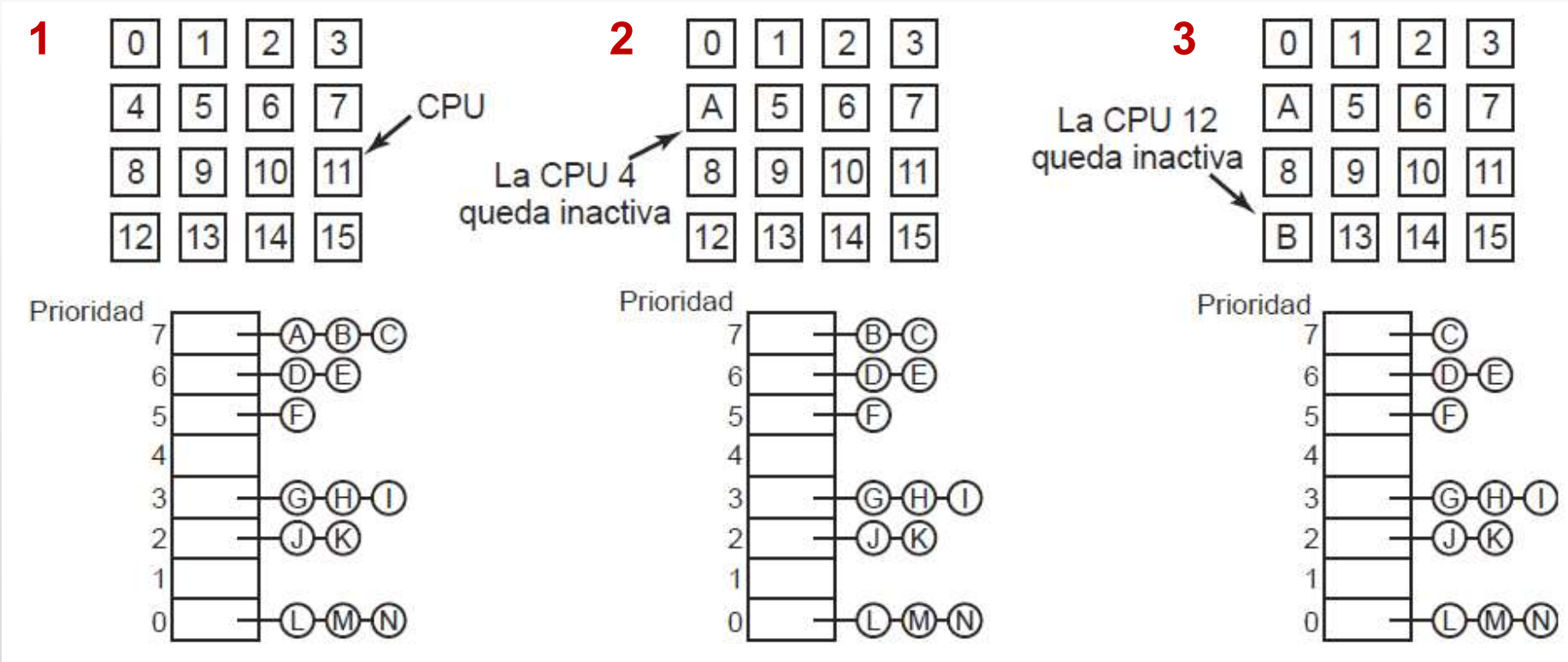
- La solución al problema anterior, es que en multiprocesadores la instrucción TSL bloquee el acceso al BUS
  - Se necesita soporte de hardware para poder implementarlo
  - Genera Carga en la memoria y el BUS, ya que la CPU que solicita debe mantener el bloqueo y las otras CPU deben esperar liberación del bloqueo
  - No es lo más eficiente…
- Existen otras soluciones más eficientes, pero se debe tener soporte del Hardware

Planificación de Multiprocesadores

- Antes de analizar como se va a hacer la planificación, es necesario determinar que se va a planificar:
  - Uniprocesador : ¿Qué hilo (o proceso) se seleccionara?
  - Multiprocesador: Se suma ¿En que CPU se va a ejecutar?
  - Complejidad con hilos que trabajan "en conjunto":
    - Si son ULT, el planificador los planifica a nivel de proceso
    - Si son KLT, es posible tomar decisiones sobre su planificación ← **Este tipo de hilos vamos a analizar**

- **Planificación de hilos independientes:**
  - Esta situación se da generalmente en ambientes de tiempo compartido
  - Muchos usuarios ejecutando tareas que generalmente no tienen relación entre sí.
  - El esquema más sencillo para planificarlos es tener una única cola de listos para todos los hilos
  - Podríamos tener varias… Una para cada prioridad

- Planificación de hilos independientes:
  - Las 16 CPU están ocupadas y hay 14 hilos en la cola de listos ordenados por prioridad



Multiprocesadores con M. Compartida (cont.)

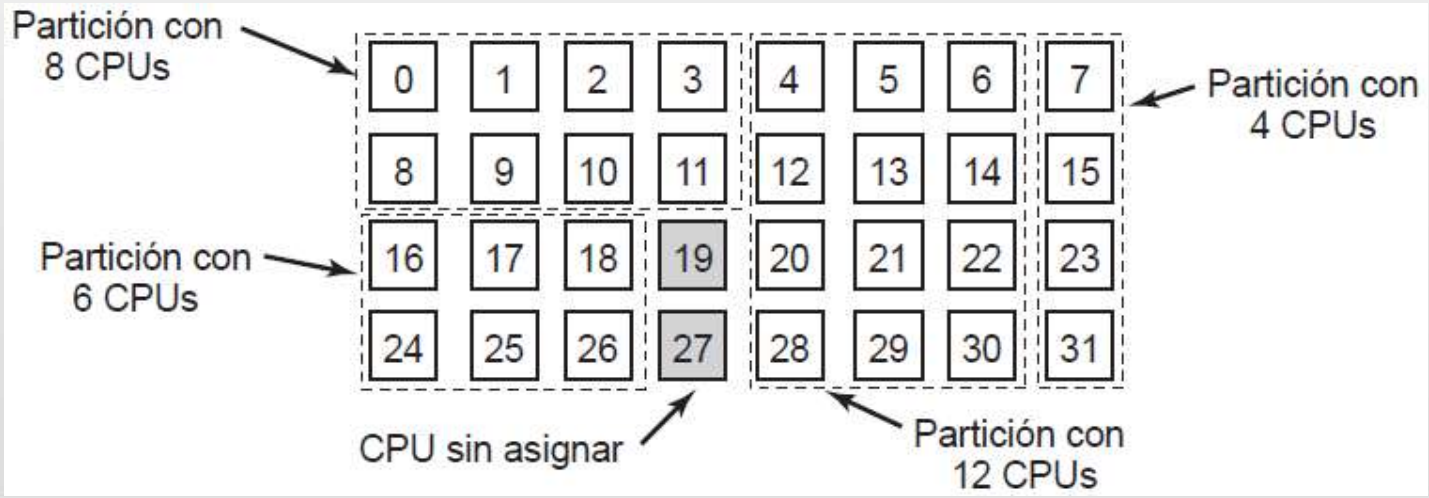
☑ Planificación de hilos independientes:

✓ Aprovechamiento de la Cache

- Un hilo que se ejecuta en un CPU donde ya se ha ejecutado, tendrá mayor posibilidad de que sus datos aún sigan en la cache de dicha CPU. Planificación por afinidad (Vaswani y Zahorjan, 1991).
- Se utiliza el algoritmo de Planificación de 2 niveles:
  - Cuando se crea un hilo, se asigna a una CPU
  - Cada CPU tiene su propia colección de hilos y los planifica por separado
  - Si queda una CPU ociosa, se reparten los hilos
  - Minimiza la contención de las estructuras de datos asociadas para la planificación, ya que no hay solo una (una o mas cola de listos **por cpu**).

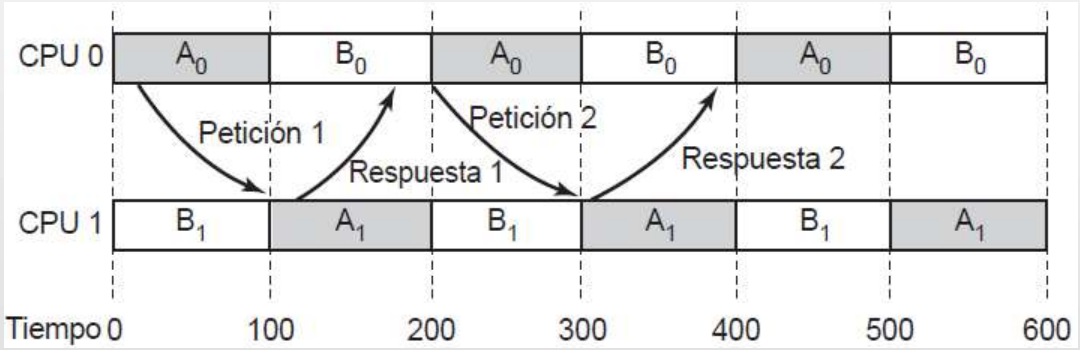
Multiprocesadores con M. Compartida (cont.)

- **Planificación de hilos que trabajan en conjunto:**
  - Se planifican en conjunto (en varias CPUs)
  - Mejora en trabajos en paralelo
  - El grupo se planifica si hay CPUs libres para cada hilo del grupo. Si no las hay, el grupo espera
  - No hay multiprogramación por CPU (cada CPU ejecuta solo 1 hilo) , baja la productividad



☑ Planificación de hilos que trabajan en conjunto:

- Se podrían multiprogramar las CPU, pero podría ocurrir que los hilos no se ejecuten sincrónicamente, ya que se planifican independientemente.
- Supongamos una situación de 2 hilos A_0 y A_1 que se ejecutan intercambiando mensajes compartiendo CPU con los hilos de un proceso B:



- Como A_0 se ejecuta en un intervalo distinto que A_1 ocurre que el lapso de ejecución de A es de 200 mseg, cuando podría haber sido de 100 si se hubieran ejecutado en el mismo intervalo.

Multiprocesadores con M. Compartida (cont.)

- **Planificación de hilos que trabajan en conjunto:**
  - **Planificación por pandillas (una solución al problema anterior):**
    - Hilos relacionados se toman como una Pandilla
    - Todos los miembros de una pandilla se ejecutan simultáneamente en distintas CPUs multiprogramadas
    - Todos los miembros de la pandilla inician y terminan sus intervalos en conjunto.
    - Ejemplo:
      - Supongamos un multiprocesador con 6 CPU utilizadas por 5 procesos (A..E) y un total de 24 hilos

Multiprocesadores con M. Compartida (cont.)

- Ejemplo (cont.)
  - En el instante de tiempo 0, se programan los hilos A_0 .. A_5. Luego se ejecuta B_0, B_1, B_2, C_0, C_1, C_2, etc…
  - Se continúa la ejecución cíclicamente

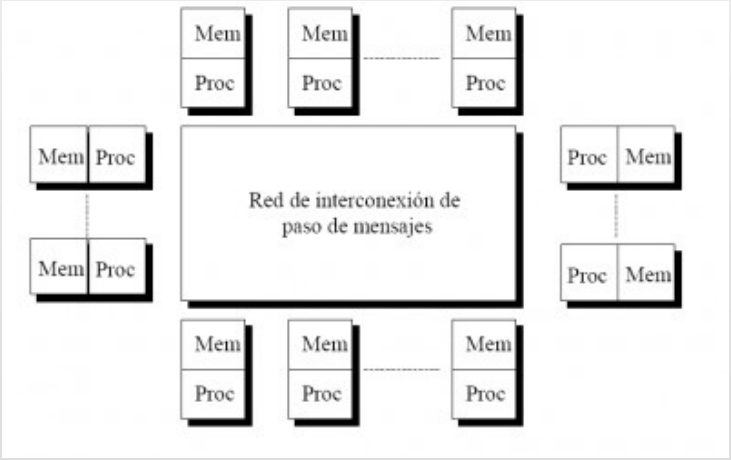
Time slot	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5
0	A_0	A_1	A_2	A_3	A_4	A_5
1	B_0	B_1	B_2	C_0	C_1	C_2
2	D_0	D_1	D_2	D_3	D_4	E_0
3	E_1	E_2	E_3	E_4	E_5	E_6
4	A_0	A_1	A_2	A_3	A_4	A_5
5	B_0	B_1	B_2	C_0	C_1	C_2
6	D_0	D_1	D_2	D_3	D_4	E_0
7	E_1	E_2	E_3	E_4	E_5	E_6

  - La idea de ejecutar todos los hilos de un proceso juntos, permite que el pasaje de mensajes se realice de manera más rápida y eficiente

Multicomputadoras

Multicomputadoras

- ☑ También conocidos como cluster de computadoras
- ☑ CPUs con acoplamiento fuerte
- ☑ No se comparte memoria, cada CPU tiene la suya
- ☑ Multicomputadora:
  - ☑ PCs con una interfaz de red de alto rendimiento
  - ☑ Generalmente carecen de Placa de Video, Sonido y en algunos casos disco



Multicomputadoras

- ☑ También conocidos como cluster de computadoras
- ☑ CPUs con acoplamiento fuerte
- ☑ No se comparte memoria, cada CPU tiene la suya
- ☑ Multicomputadora:
  - ☑ PCs con una interfaz de red de alto rendimiento
  - ☑ Generalmente carecen de Placa de Video, Sonido y en algunos casos disco

Multicomputadoras

- ☑ También conocidos como cluster de computadoras
- ☑ CPUs con acoplamiento fuerte
- ☑ No se comparte memoria, cada CPU tiene la suya
- ☑ Multicomputadora:
  - ☑ PCs con una interfaz de red de alto rendimiento
  - ☑ Generalmente carecen de Placa de Video, Sonido y en algunos casos disco

Multicomputadoras

- Multicomputadora:
  - Poseen mucha CPU, memoria y placas de interconexión redundantes:
    - Red: Ethernet (10, 40, 100, 400 Gbps), Infiniband (120 Gbps)
    - HBA: Host Bus Adapter para conexión Storage. Generalmente Fibra óptica y usa el protocolo FibreChannel
  - Es necesario diseñar correctamente la red

Multicomputadoras (cont.)

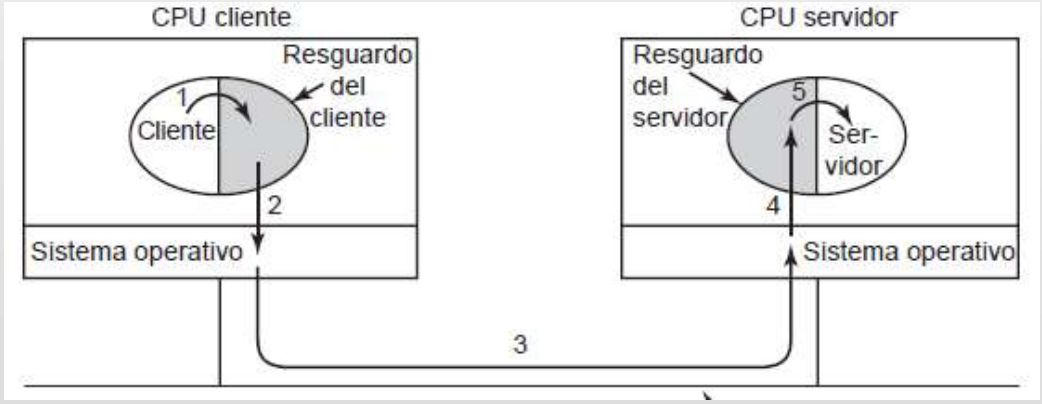
- **Software de Comunicación a nivel de usuario:**
  - Para comunicarse, los procesos en distintas CPUs en una multicomputadora se envían mensajes entre **sí.**
  - **Paso de mensajes:**

```
send(dest, &mptr);
receive(direc, &mptr);
```

    - Send y Receive con y sin bloqueo:
      - El SO Provee interfaces a los procesos de usuario para realizar la comunicación.
      - Envío con bloqueo (síncronas) (la CPU queda inactiva durante la transmisión del mensaje).
      - Envío sin bloqueo con copia (asincrónicas) (se desperdicia el tiempo de la CPU por la copia adicional).
      - Envío sin bloqueo con interrupción (dificulta la programación).

Multicomputadoras (cont.)

- **Software de Comunicación a nivel de usuario:**
  - Otra alternativa: RPC (Remote Procesure Call) para obtener un mayor nivel de abstracción
    - Permite invocar a procedimientos que se ejecutan en otra CPU.
    - El proceso en una máquina invoca al procedimiento remoto y se bloquea hasta que llegue la respuesta
    - La comunicación es transparente al programador



Multicomputadoras (cont.)

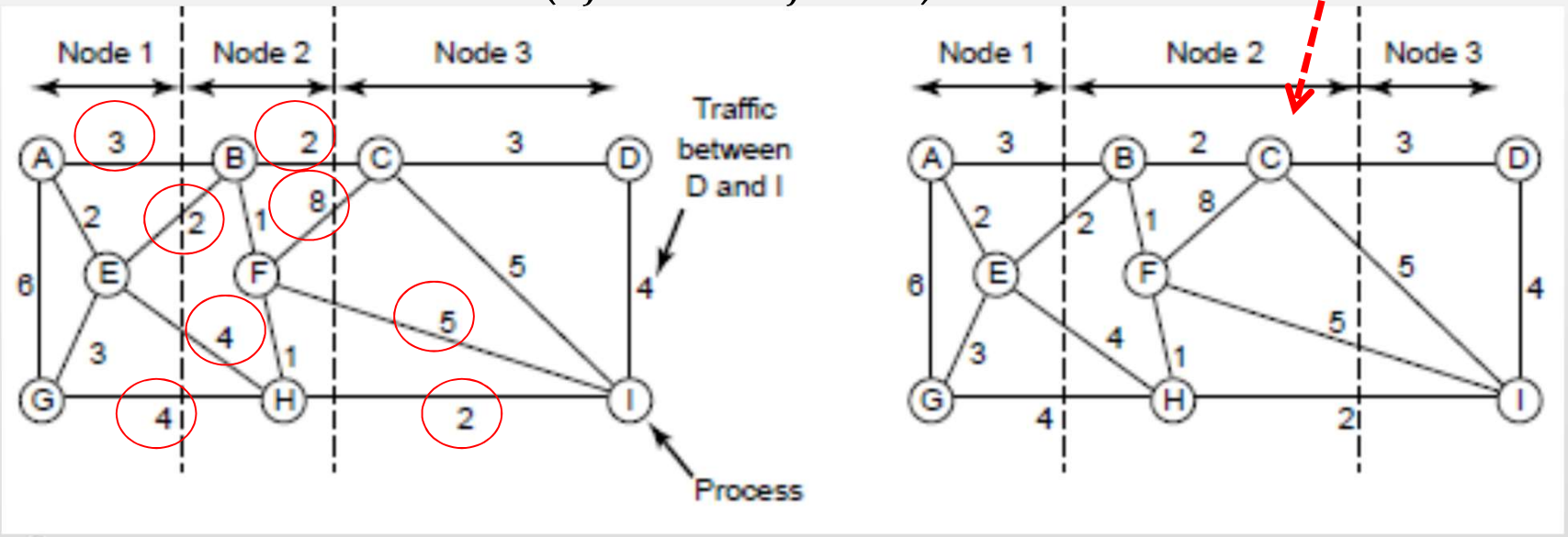
- Planificación
  - Cada nodo tiene su propio conjunto de procesos
  - Un nodo no toma procesos de otros para ejecutarlos, ya que seria bastante costoso
  - Es importante la asignación de procesos a los nodos
    - Balanceo de la carga
  - Se puede aplicar el concepto de planificación por pandillas
    - Sincronización entre los nodos en cada inicio de una ranura de tiempo

Multicomputadoras (cont.)

- ☑ Planificación – Balanceo de la carga (cont.)
  - ✓ Se representa al sistema como un grafo
  - ✓ Cada nodo es un proceso, y la comunicación entre ellos se representa a través de una arista
  - ✓ Se debe particionar el grafo en tantos subgrafos como nodos se tengan, teniendo en cuenta:
    - ✓ Requerimientos totales de CPU
    - ✓ Requerimientos totales de memoria
    - ✓ Minimizar la cantidad de aristas entre nodos de distintos subgrafos (tráfico de red)
    - ✓ Buscar clusters con acoplamiento fuerte (veamos el ejemplo)

Multicomputadoras (cont.)

- Planificación – Balanceo de la carga (cont.)
  - Tenemos 9 procesos, sabiendo el costo de comunicación entre estos y tenemos 3 nodos
  - Trafico de la red, suma de los pesos entre enlaces de 2 nodos diferentes (ej1: 30 , ej2: 28)

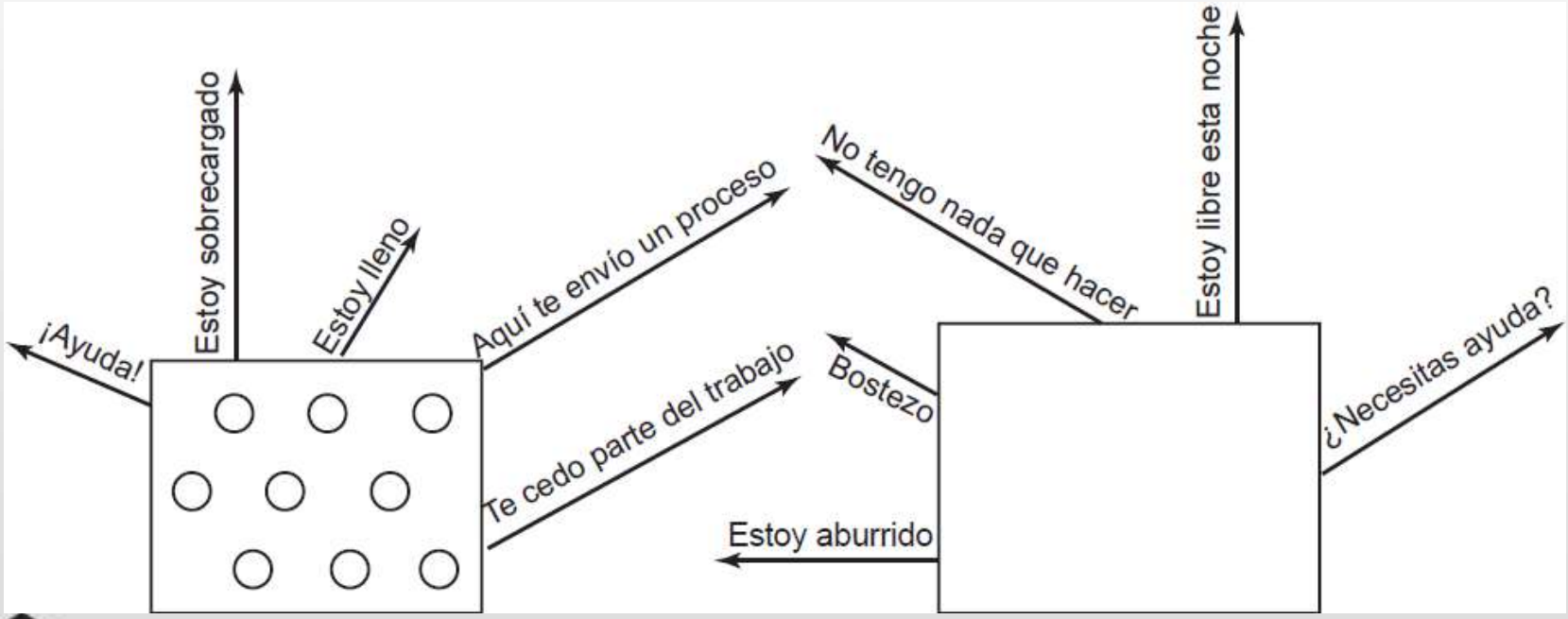


Multicomputadoras (cont.)

- Planificación – Balanceo de la carga (cont.)
  - Otra forma
    - Usar Algoritmos distribuidos
    - El proceso se ejecuta en el nodo que lo creo al menos que el mismo este sobrecargado
      - Muchos procesos, trashing, etc.
    - El nodo sale a buscar un nodo no sobrecargado para "pasarle" el proceso
    - Problemas con sobrecarga del enlace si todos los nodos se encuentran sobrecargados. Mucho pasaje de mensajes!

Multicomputadoras (cont.)

- ☑ Planificación – Balanceo de la carga (cont.)
  - ✓ Otra forma (cont.)
    - Es posible también que un nodo con poca carga informe la situación



Sistemas Distribuidos

Definición

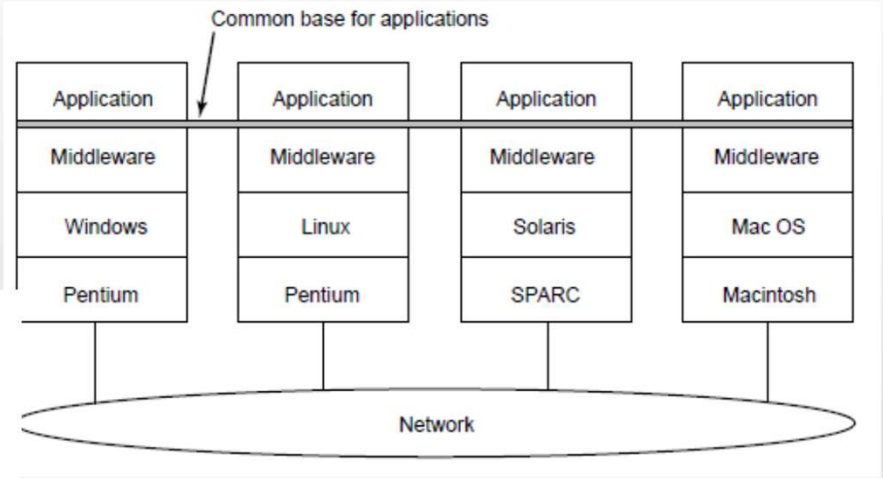
- A distributed system is a collection of independent computers that appears to its users as a single coherent system (Tanembaum)
- A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages (Colouris)

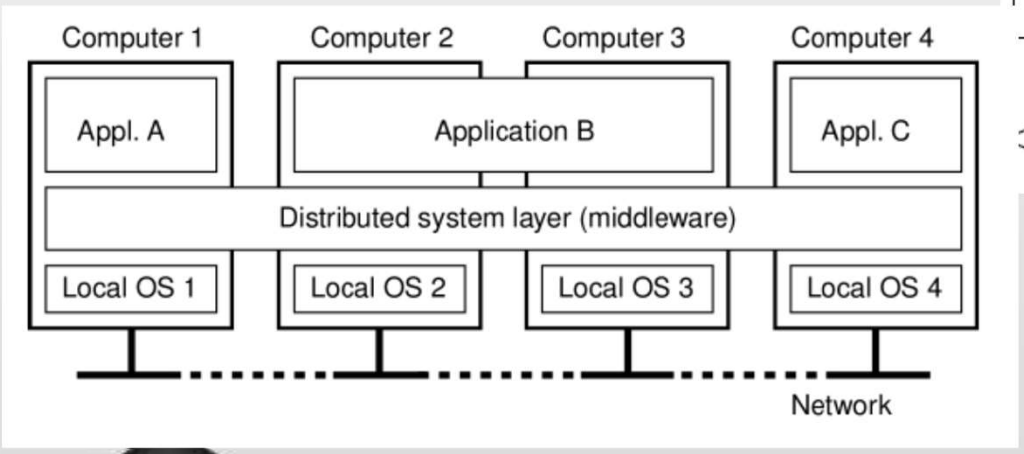
Sistemas Distribuidos

- ☑ Similar a las multicomputadoras
  - ✓ Cada nodo tiene su propia memoria privada
- ☑ Menor acoplamiento
  - ✓ Se pueden encontrar esparcidos por todo el mundo
- ☑ Cada nodo es una PC completa
  - ✓ Incluyendo dispositivos
- ☑ Cada nodo puede ejecutar un SO y Hardware diferente
  - ✓ Incluyendo su propio sistemas de archivos

Sistemas Distribuidos (cont.)

- Middleware
  - Capa de software por encima del SO que permite una uniformidad entre los distintos SOs.





Sistemas Distribuidos (cont.)

- El **middleware** es una capa de Software fundamental en un Sistema distribuido:
  - Provee una interfaz común a todos los procesos
  - Soluciona los problemas de heterogeneidad,
  - Provee servicios a las capas superiores
  - Provee estructuras de datos y operaciones que permiten a los procesos y usuarios inter-operar, de manera consistente, entre máquinas remotas