Sistemas Operativos

Multiprocesadores - I

Sistemas Operativos

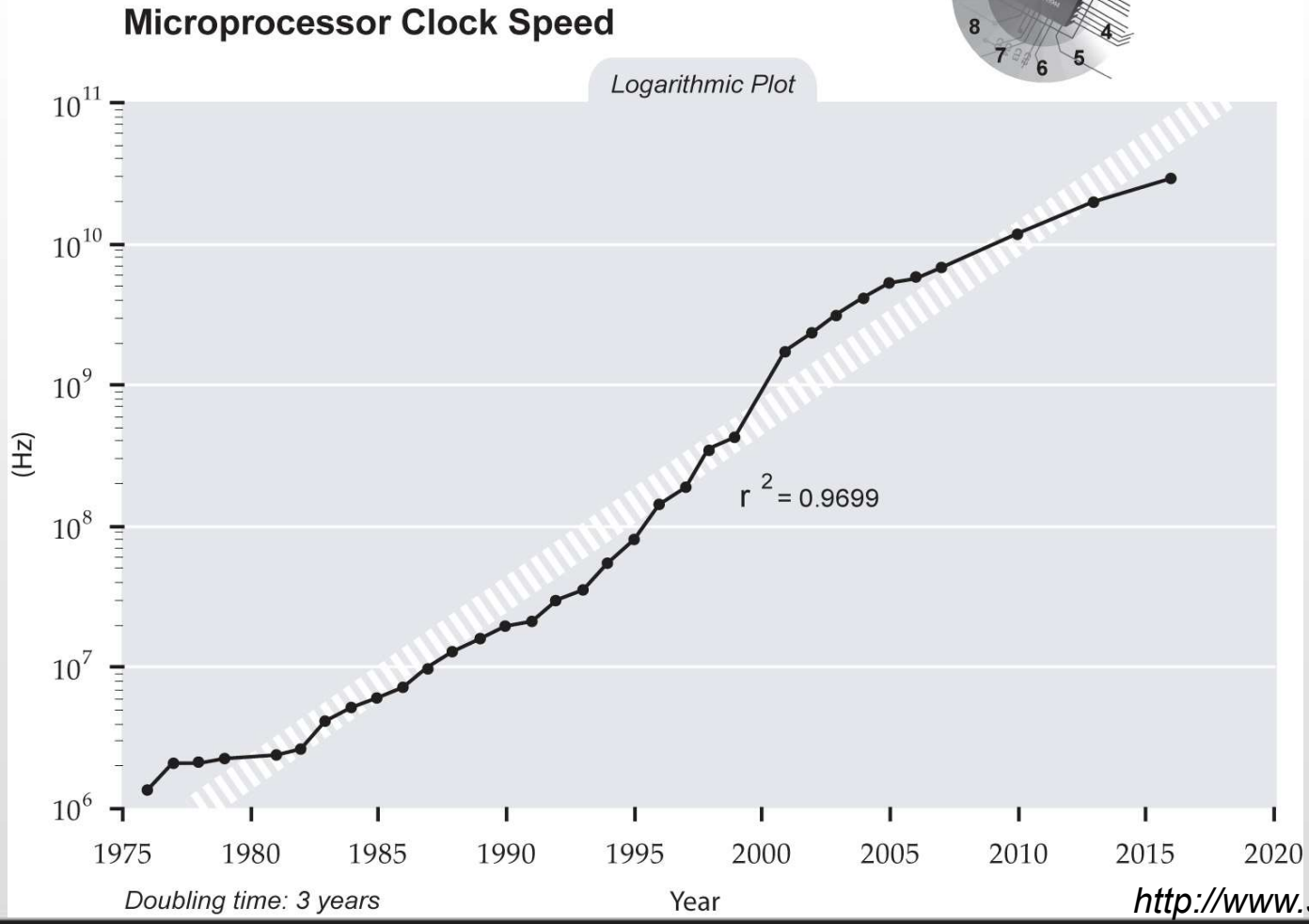
- ☑ Versión: Marzo 2020
- ☑ Palabras Claves: Multiprocesadores, SMP, Redes, Distribuidos, UMA, NUMA

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)
- En el pasado, la solución era siempre hacer que el reloj operara a mayor velocidad

Origen



http://www.singularity.com

Origen

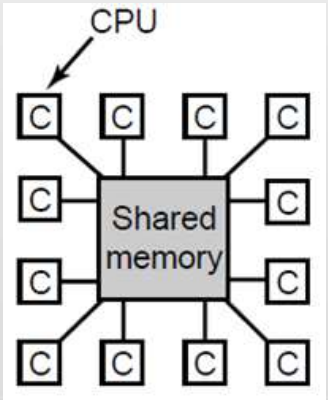
- En la actualidad, lograr mayor velocidad es físicamente complejo:
  - Ninguna señal eléctrica se puede propagar más rápido que la velocidad de la luz
  - Problemas de disipación de calor (muchos transistores juntos en poco espacio)
  - Problemas de consumo eléctrico
- La solución al problema es el 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

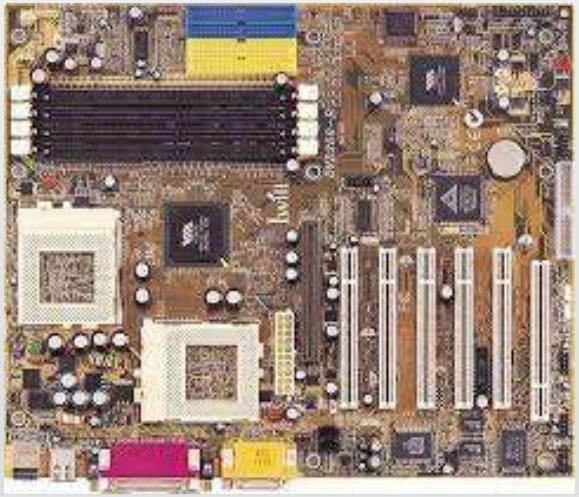
Esquemas

- 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

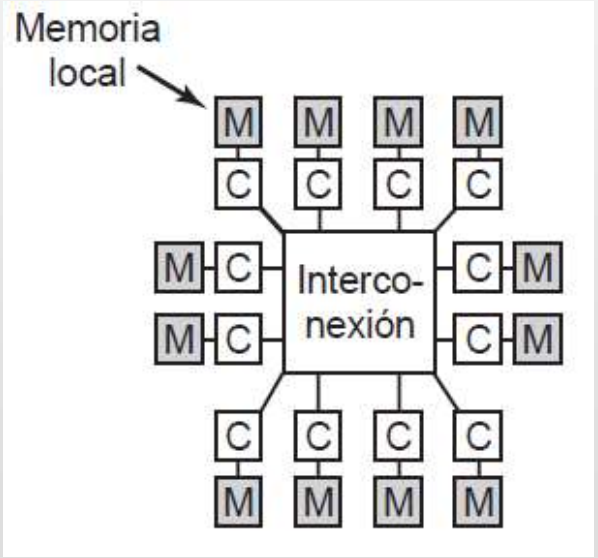
- **1. Multiprocesadores con Memoria Compartida**
  - La comunicación entre las CPU es a través de la memoria compartida
  - Cada CPU tiene el mismo acceso que otras a la memoria física a través de un único BUS físico
  - Para acceder a una palabra de memoria por lo general cada CPU requiere de 2 a 10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





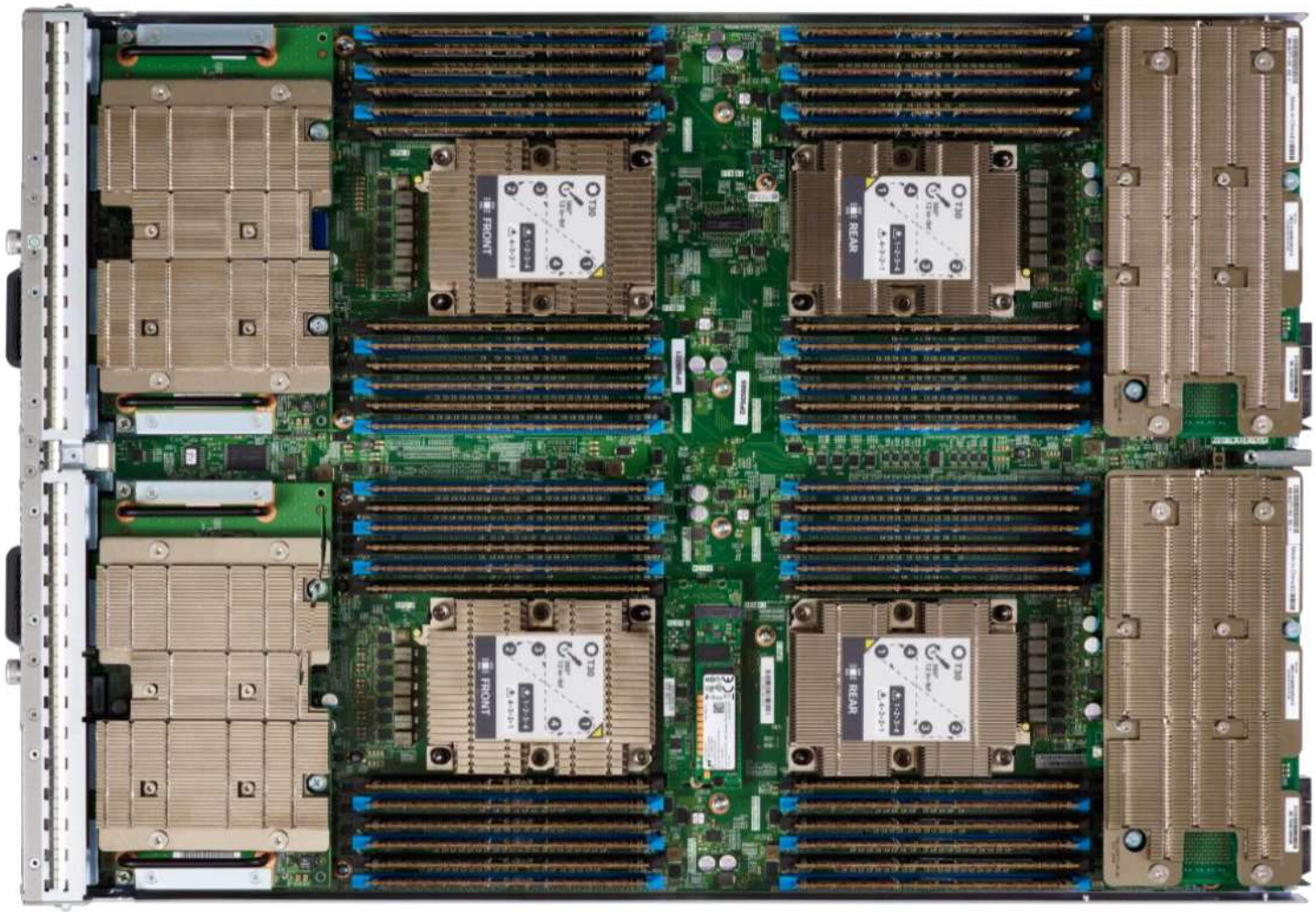
Esquemas

- **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

- ☑ M

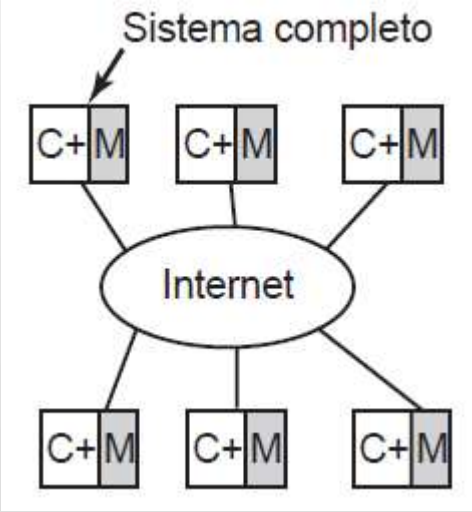


s

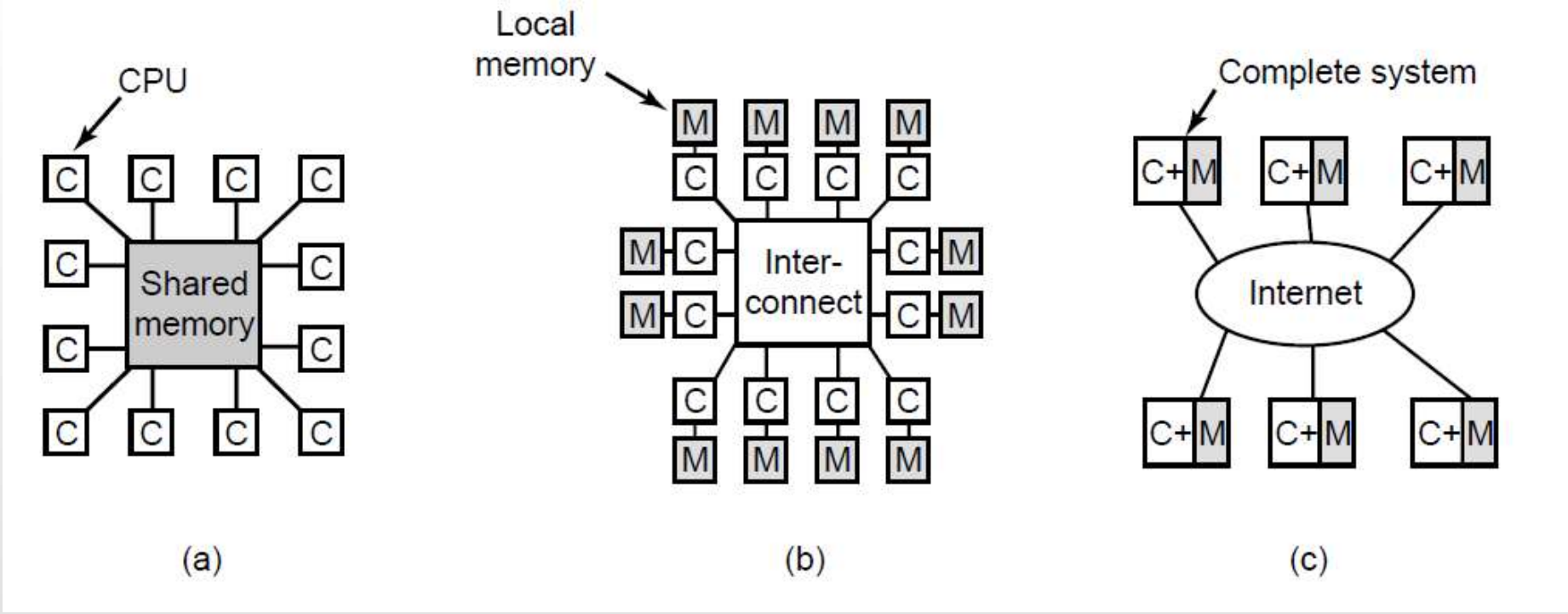
lo,

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
  - También conocidos como sistemas débilmente acoplados



Esquemas (cont.)



Diferencias en las velocidades de comunicación

1. Multiprocesadores con M. Compartida

- Dos o mas CPU comparten el acceso a la RAM
- Los programas se ejecutan en cualquier CPU
- Cada proceso ve un espacio normal de direcciones virtuales
- La CPU puede escribir cierto valor en una palabra de memoria y después puede volver a leer esa palabra y obtener un valor distinto (tal vez porque otra CPU lo cambió).
- Si se sincronizan correctamente, una CPU escribe y la otra Lee.

Multiprocesadores con M. Compartida (cont.)

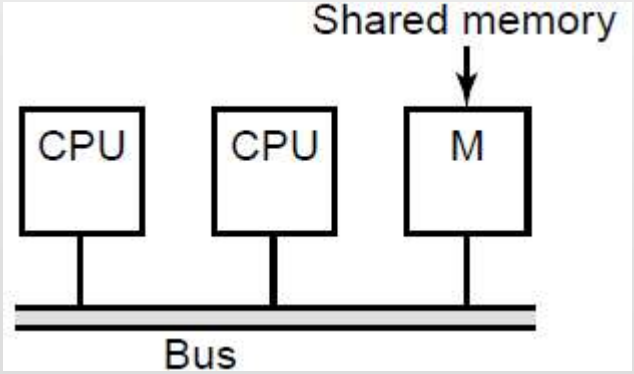
- Por lo general, los sistemas operativos realizan tareas regulares como lo son:
  - Manejo de System Calls
  - Administración de Memoria
  - Administración de E/S (sist. de archivos y dispositivos)
- Algunas características Particulares:
  - Sincronización de procesos
  - Administración de Recursos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

Multiprocesadores con M. Compartida (cont.)

- Hardware
  - Cada procesador puede direccionar toda la memoria
  - Dependiendo del procesador y la velocidad de acceso a la memoria, los podemos clasificar en:
    - UMA (Uniform Memory Access)
    - NUMA (Non-uniform memory Access)

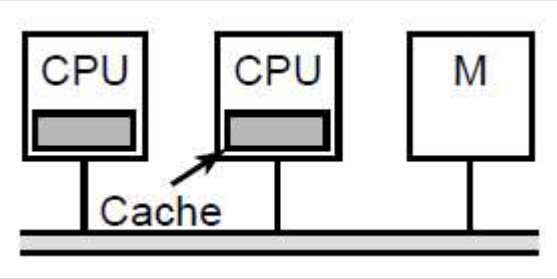
Multiprocesadores con M. Compartida (cont.)

- Hardware – UMA
  - Los multiprocesadores más simples se basan en un solo bus para comunicarse con la memoria
  - Antes de acceder al BUS, se debe comprobar que el mismo no se encuentre ocupado
  - A medida que aumenta la cantidad de CPUs, el acceso a memoria se torna mas ineficiente
  - La principal limitación es el ancho de banda del BUS
  - Se tiene mucho tiempo de CPU ocioso



Multiprocesadores con M. Compartida (cont.)

- Hardware – UMA
  - Una evolución, es agregar una cache a cada CPU
  - Al permitir almacenar datos en la cache, el acceso al BUS compartido se reduce.
  - Si el bloque se almacena en la cache en modo RO (read only), puede estar en varias cache
  - Si el bloque se almacena en modo RW, solo puede residir en una cache
  - Aparecen mecanismos de protección para evitar datos "sucios"
  - Copias "limpias" vs. Copias "sucias"

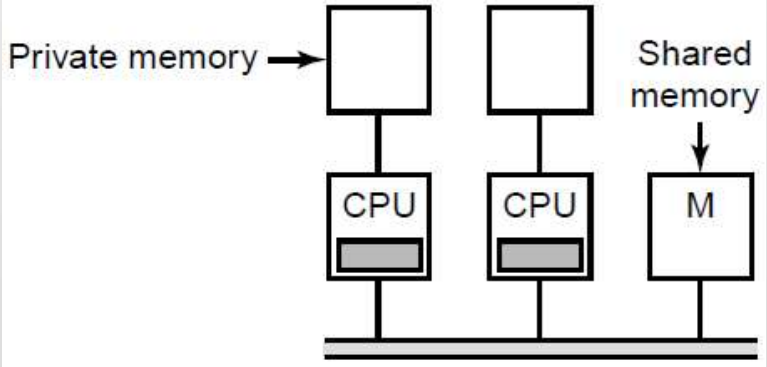


Multiprocesadores con M. Compartida (cont.)

- Hardware – UMA
  - Aparecen mecanismos de protección para evitar datos "sucios":
    - Cuando una CPU intenta escribir una palabra que esta en una o más caches se manda un mensaje al bus para informar:
      - Si las otras CPU tienen copia limpia (la misma), la descartan para que solo quede la que se va a modificar
      - Si otra CPU tiene copia sucia, la escribe a memoria e informa a la CPU para mantener coherencia
    - A este conjunto de reglas se le conoce como protocolo de coherencia de cachés y es uno de muchos

Multiprocesadores con M. Compartida (cont.)

- Hardware – UMA
  - Otra alternativa es asignar a cada CPU un área de memoria local que es accedida por un BUS dedicado. → Caché + Memoria Privada
  - La memoria compartida solo se utiliza para escribir variables compartidas
  - Reduce el uso del BUS, pero requiere de una asistencia activa por parte del compilador

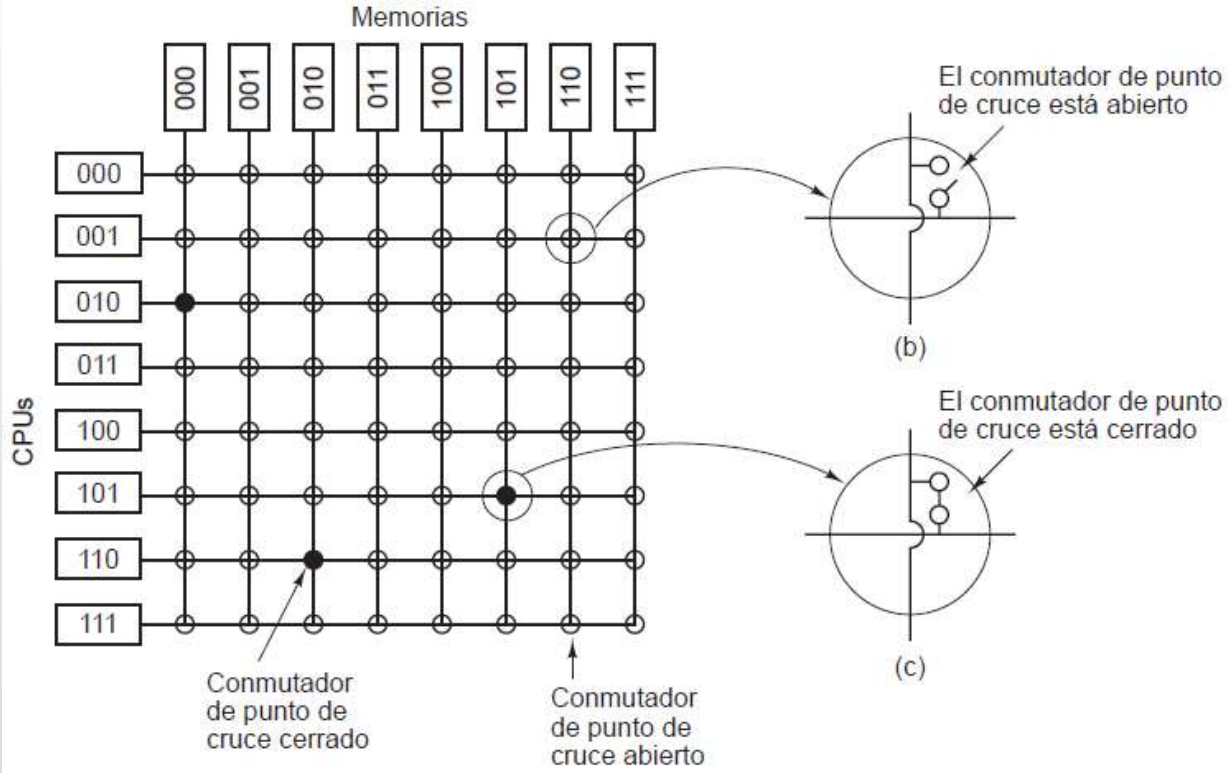


☑ Hardware – UMA – Otras arquitecturas

- ✓ Inclusive con el agregado de cache, un único BUS limita a 16 o 32 CPU aproximadamente
- ✓ Para lograr mejor performance se necesita otro esquema de interconexión:

Interruptores de barras cruzadas:

n CPU and n memories require n^2 switches

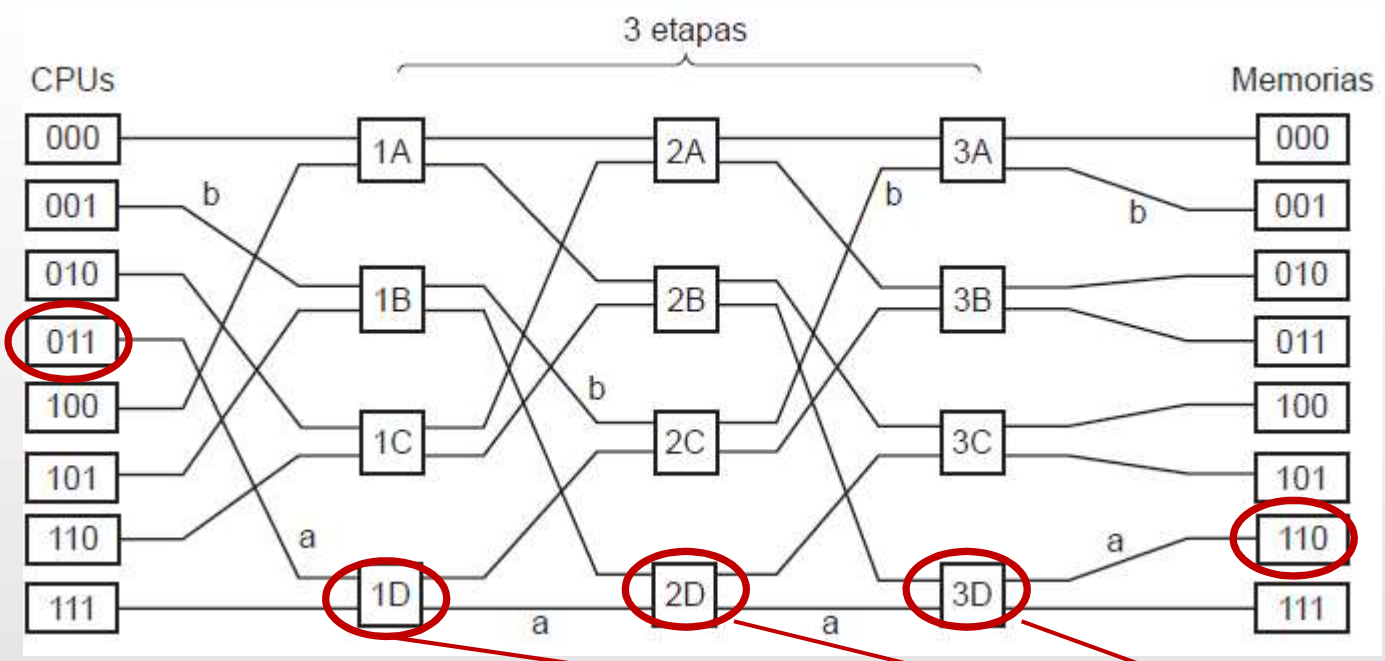


☑ Hardware – UMA – Otras arquitecturas

- ✓ Inclusive con el agregado de cache, un único BUS limita a 16 o 32 CPU aproximadamente
- ✓ Para lograr mejor performance se necesita otro esquema de interconexión:
  - ✓ **Redes de conmutación multi-etapa:**
    - ✓ Se pueden ver como una especie de grafo dirigido de caminos que permiten comunicar cada CPU con un módulo de memoria por un camino independiente.
    - ✓ Cada nodo es un conmutador con 2 entradas y 2 salidas

Multiprocesadores con M. Compartida (cont.)

- ✓ Redes de conmutación multi-etapa



- ❑ La CPU 011 solicita leer un ítem del modulo de memoria 110
- ❑ La CPU envía un mensaje READ al switch 1D con Modulo 110 -- 011.
- ❑ EL switch analiza el bit más significante a izquierda y lo utiliza para rutear: 0 rutear hacia arriba, 1 hacia abajo. En este caso se rutea a 2D.

Multiprocesadores con M. Compartida (cont.)

- ✓ Redes de conmutación multi-etapa
  - ✓ Los mensajes entre CPU y memoria están compuestos por 4 partes:
    - ✓ Module: which memory block is requested – which CPU is requestor
    - ✓ Address: address within memory block;
    - ✓ Opcode: operation to carry out (READ or WRITE);
    - ✓ Value (optional): value to be written (for a WRITE)
- ❑ La ventaja frente al esquema de barras cruzadas es que solo requiere 12 switches para realizar la misma tarea
- ❑ Tanto los interruptores de barras cruzadas, como las redes multietapas tienen como finalidad presentar circuitos (2 o mas) que permitan la comunicación de varias CPUs con varios módulos de memoria

Multiprocesadores con M. Compartida (cont.)

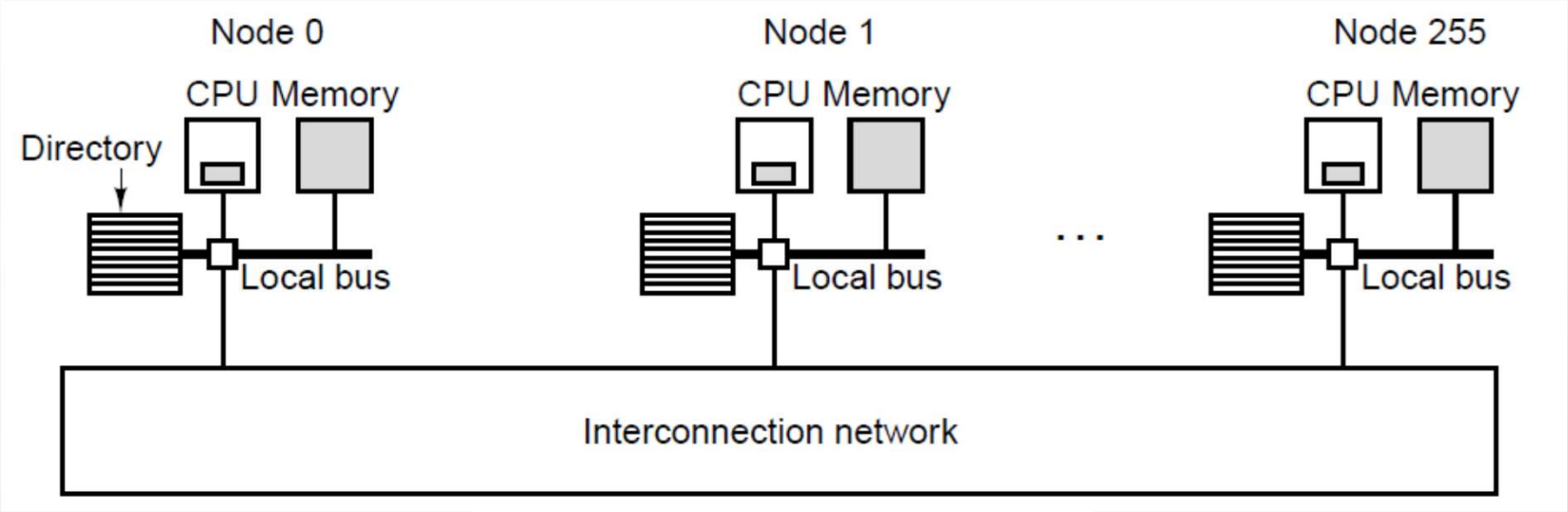
- Los procesadores y técnicas anteriores tienen la característica de ser poco escalables y muy costosos.
- **Hardware – NUMA (Non-uniform memory Access)**
  - Permiten escalar en número de CPUs
  - Se posee un único espacio de memoria visible por todas las CPUs
  - El acceso a la memoria remota es más lento que el acceso a la memoria local, ya que se requiere acceso a un bus compartido
  - El rendimiento es menor que en máquinas UMA, pero es menos costoso

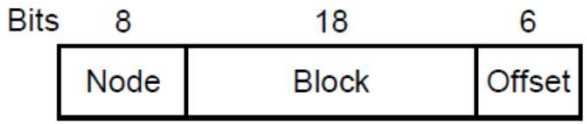
Multiprocesadores con M. Compartida (cont.)

- **Hardware – NUMA (Non-uniform memory Access)**
  - Se pueden implementar mecanismos de cache para acelerar los tiempos de acceso a la memoria
  - Cuando no hay cache, al sistema se lo llama NC-NUMA (No Cache NUMA), cuando hay cache se lo llama CC-NUMA (Cache Coherent NUMA)
  - En el uso de CC-NUMA, es importante mantener una coherencia en las caches
  - El método mas común para construir grandes procesadores basados en CC-NUMA es el multiprocesador basado en directorios:
    - Mantiene una BD que indica donde está cada línea y su estado (limpia o sucia (modificada))

Multiprocesadores con M. Compartida (cont.)

- Hardware – NUMA (cont.)
  - Una dirección de memoria debe traducirse en: nodo+línea+desplazamiento





Multiprocesadores con M. Compartida (cont.)

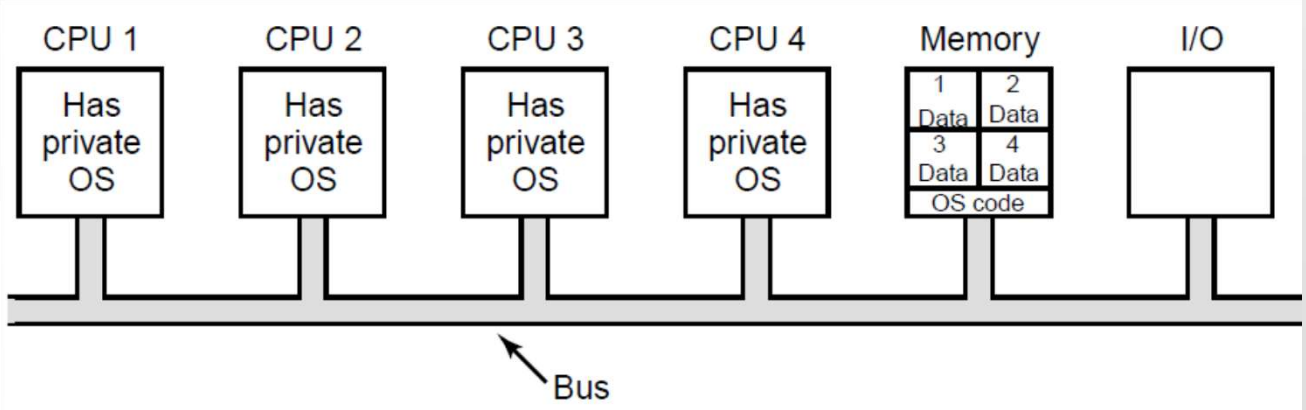
- **Chips Multinúcleo**
  - A medida que la tecnología avanza, los transistores se hacen mas pequeños y aparece la posibilidad de agregar más de uno a un chip.
  - Al tener mas transistores se puede:
    - Agregar más memoria cache → Está demostrado que la tasa de aciertos no se incrementa demasiado
    - Agregar mas velocidad de clock a una CPU → Sigue existiendo un unico 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:

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

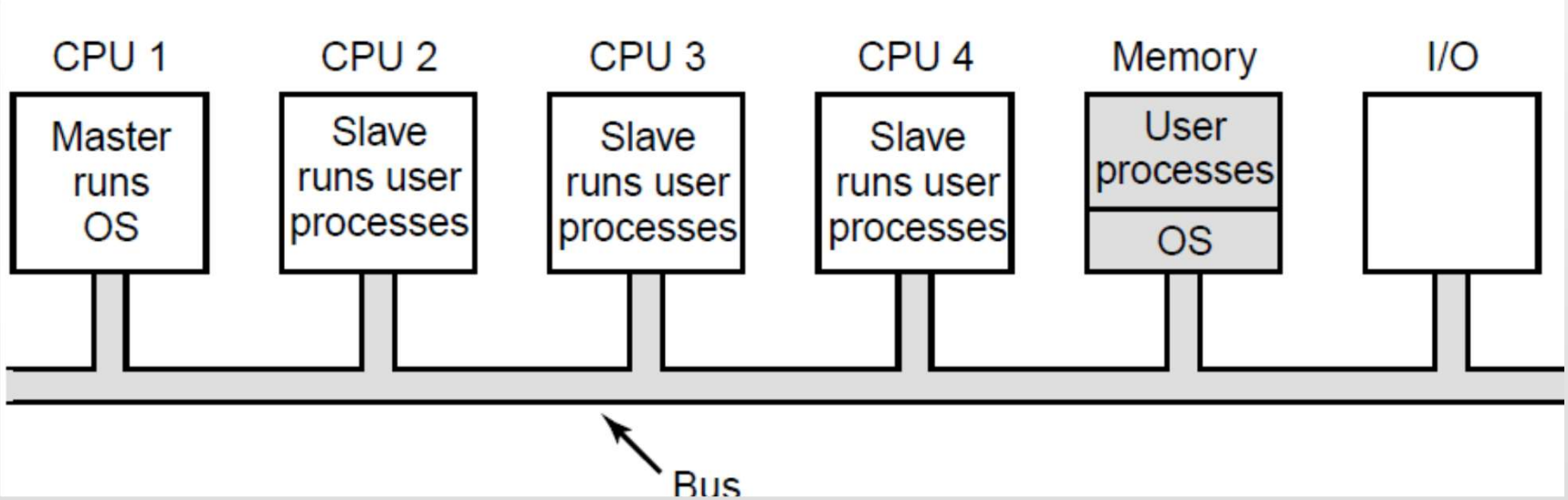


Tipos de SO Multiprocesador

- 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
   - No se pueden compartir paginas
     - Pasaje de mensajes
     - Memoria desperdiciada
   - Cache de disco, cada CPU tiene su propia copia
     - Inconsistencia de la información

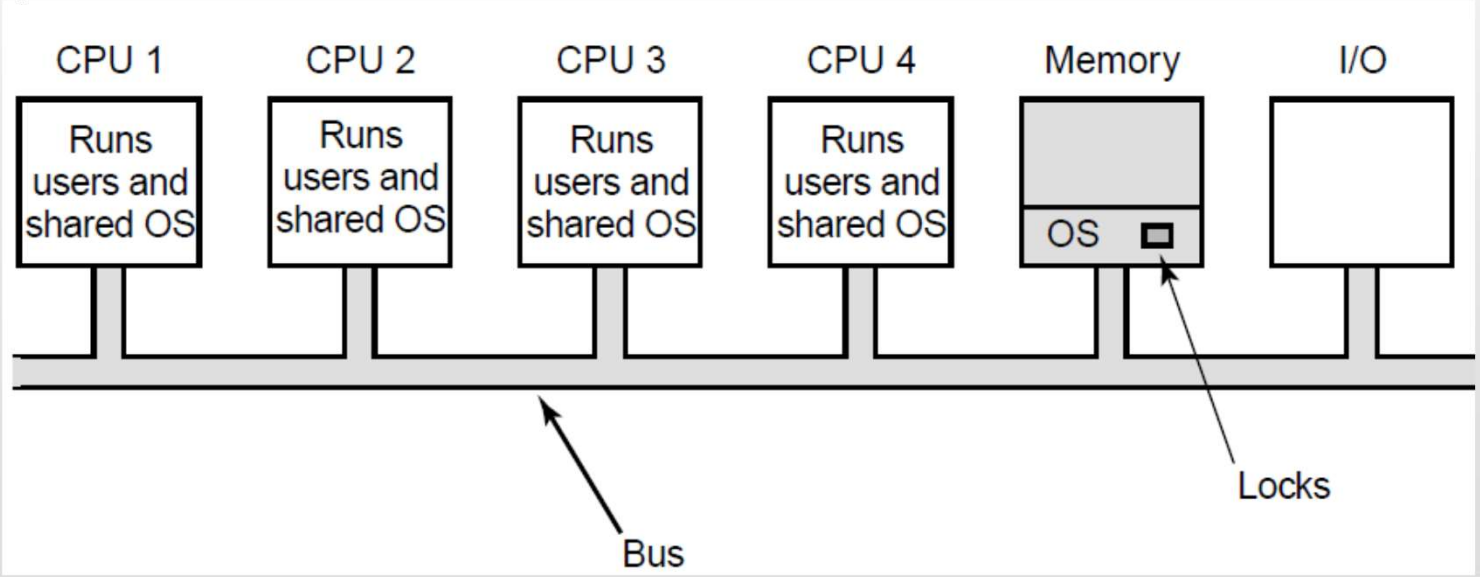
2. Maestro – Esclavo
   - Única copia del SO y de su información
   - Todas las SysCalls se redirigen a una CPU
   - Está CPU puede ejecutar procesos si "le sobra tiempo"



2. Maestro – Esclavo
   - Resuelve problemas del modelo anterior
     - Se mantiene una única cola de listos
     - 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.)

- 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.)**

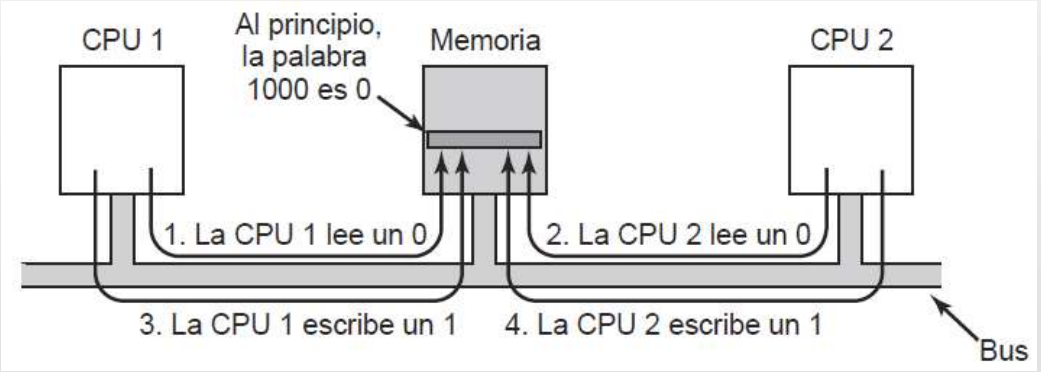
- Posibles soluciones a los problemas planteados:
  2. **Lock por estructura(s):**
     - Existen varias secciones criticas independientes cada una protegida por su propio mutex
     - Mejora el rendimiento
     - Dificultad para determinar cada sección critica
     - **Ciertas estructuras pueden pertenecer 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
  - Problema con el Spin-lock (bloqueo de giro)
    - Se desperdicia tiempo de la CPU que hace la petición, ya que se debe elevar el nivel de procesador
    - Genera Carga en la memoria y el BUS
  - No es lo más eficiente…

Sincronización de Multiprocesadores

- ✓ Surge una nueva solución que es el uso de cache (para evitar el bus), pero también genera problemas:
  - Al leer la palabra en la cache, generalmente se realizan modificaciones
  - Como se modifica, se deben invalidar todas las copias de las otras caches (trashing), lo que causa que se deba escribir el valor a la memoria y las otras CPUs lo relean.
  - Esto genera mayor uso del BUS
  - Además al tener un lockeo establecido por una CPU, las otras están continuamente consultando por la liberación

Sincronización de Multiprocesadores

- ✓ Otras soluciones
  - Cada CPU tiene su propia variable de lockeo en cache
  - La CPU que no puede obtener el bloqueo se agrega a una lista y espera en su propio lock
  - Agregar "delays" entre cada intento de TSL