Tema 4. Sistemas operativos distribuídos

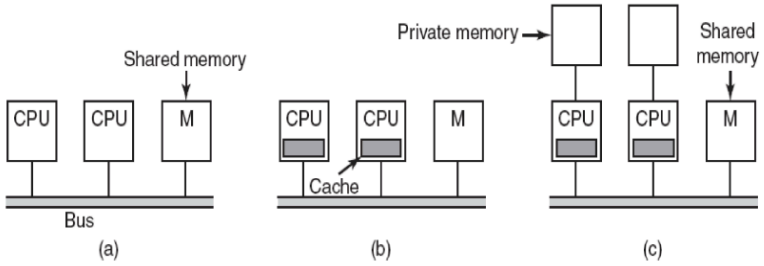
Tipos de multiprocesadores

Hardware de multiprocesadores UMA

Aínda que *todos os multiprocesadores teñen a propiedade de que cada CPU pode direccionar toda a memoria*, algúns teñen a característica de que **cada palabra de memoria se pode ler coa mesma velocidade que calquera outra palabra**. Estas máquinas coñécense como multiprocesadores **UMA** (*Uniform Memory Access*, Acceso uniforme á memoria).

UMAs con arquitectura baseada en bus

Os multiprocesadores máis simples baséanse **nun só bus**. Se está **inactivo**, a CPU *coloca a dirección da palabra que desexa no bus*, declara unhas cantas sinais de control *e espera* ata que a memoria coloque a palabra desexada no bus. Se o bus está **ocupado** cando unha CPU desexa ler ou escribir na memoria, a CPU *agarda ata que o bus estea inactivo*.



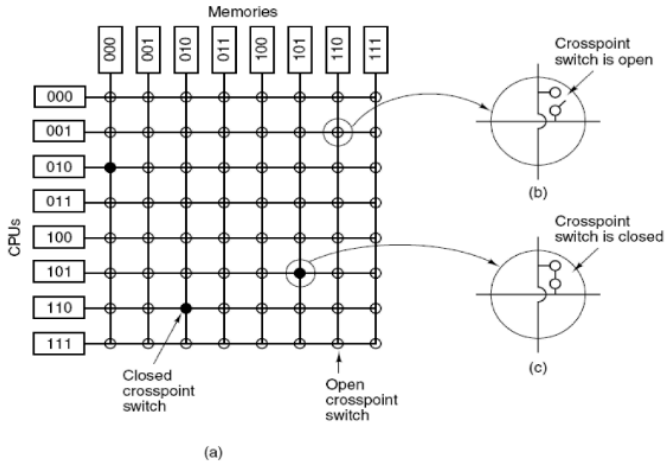
Problema: con un *número elevado de CPUs*, a maioría delas estarán unha gran cantidade de tempo *inactivas*.

Solución 1: Engadir unha *caché a cada CPU*. Cando se fai referencia a unha palabra, obténse o seu bloque completo e gárdase na caché privada en modo lectura (permitindo que o mesmo bloque estea en varias cachés) ou lectura-escritura → *protocolo de coherencia*

Solución 2: Cada CPU ten *unha caché* e unha *memoria privada local* que utiliza mediante un *bus privado*. Deste xeito, *a memoria compartida úsase só para escribir variables compartidas*. O *compilador* debe colocar todo o texto do programa, as cadeas, constantes e demais datos de só lectura, as pilas e as variables locais nas memorias privadas.

UMAs basados en interruptores de barras cruzadas

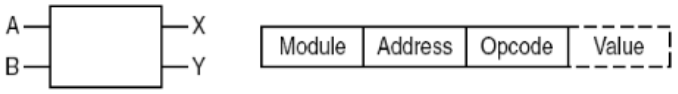
O uso **dun único bus limita o número de CPUs** que se poden empregar, polo que se propoñen os **interruptores de barras cruzadas**, que son os circuítos máis simples para *conectar n CPUs a k memorias*. En *cada intersección* dunha liña horizontal (entrante) e unha vertical (saínte) hai un **punto de cruce**, o cal é un interruptor que pode estar aberto ou pechado, dependendo de se as liñas se van conectar ou non.



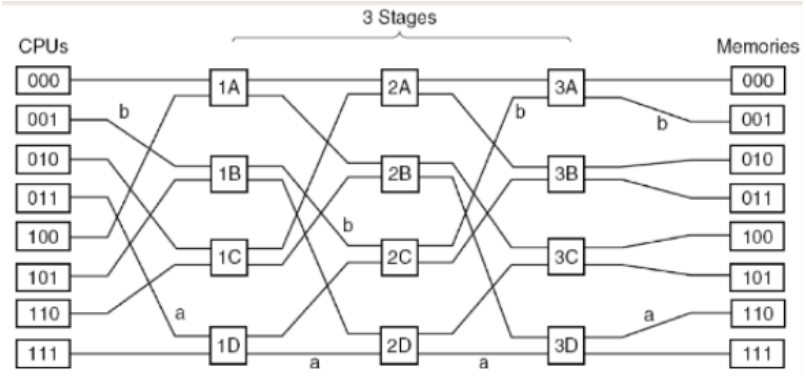
É unha **rede sen bloqueos** en moitos casos e **non precisa planificación**, aínda que se poden chegar a construír *redes moi grandes*.

UMAs que utilizan redes de conmutación multietapa

Partimos dun **conmutador simple con 2 entradas e 2 saídas**, recibindo como entrada mensaxes da forma que se ve no debuxo.



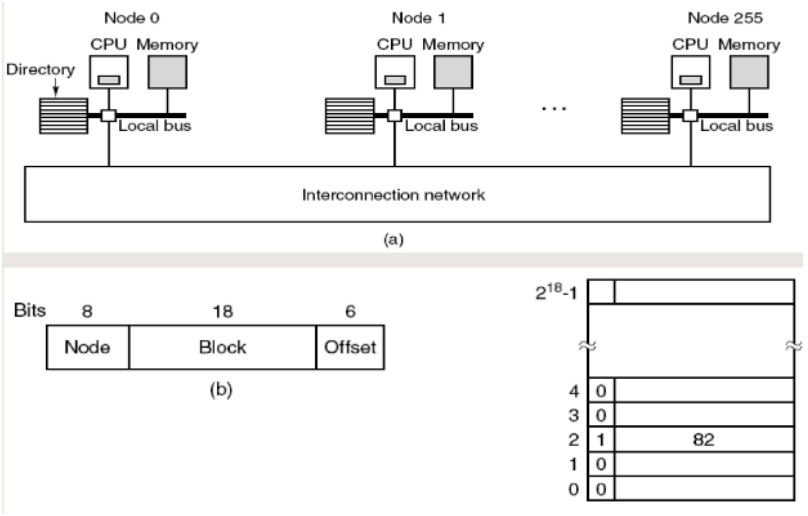
A partir deste tipo de conmutadores pódense crear *redes de conmutación multietapa*. Un exemplo é a **rede omega**.



Se unha CPU quere acceder a unha *memoria coa que, en principio, non está conectada*, por exemplo a 011 coa memoria 111, compáranse bit a bit e, se son iguais, esa conexión será en paralelo e, se non, cruzada. Isto causa **contención**, xa que deberemos *sacrificar a conexión dalgúns módulos*.

Hardware de multiprocesadores NUMA

O **espazo de memoria é compartido** entre todos os procesadores, polo que o **tempo de acceso á memoria non é uniforme**. Necesita **protocolos de coherencia**, que poden estan baseados en *directorio*.



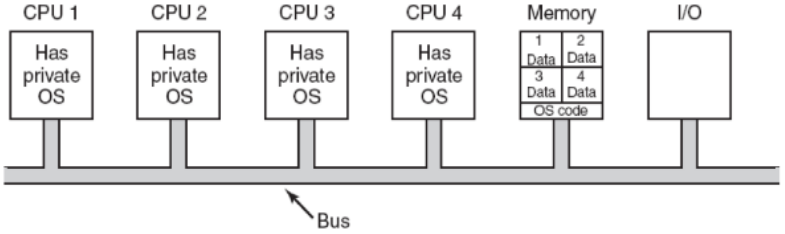
Tipos de sistemas operativos multiprocesador

Pasamos do hardware de multiprocesador ao **software**. Hai varias metodoloxías posibles.

Cada CPU ten o seu propio sistema operativo

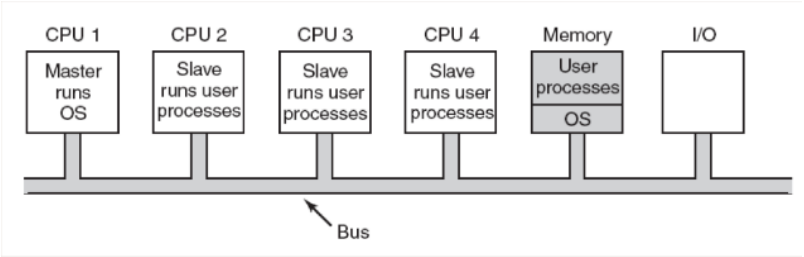
A primeira opción baséase en que cada CPU teña a súa propia **copia privada do sistema operativo**. Aínda que ao executar un programa o *código tamén sexa o mesmo para todas, cada unha ten datos distintos*, procesos e páxinas que non se comparten. Isto *inclúe tamén as táboas de procesos e páxinas*.

Unha **optimización** obvia é *permitir que todas as CPUs compartan o código do sistema operativo* e obteñan copias privadas só das estruturas de datos.



Multiprocesadores mestre-escravo

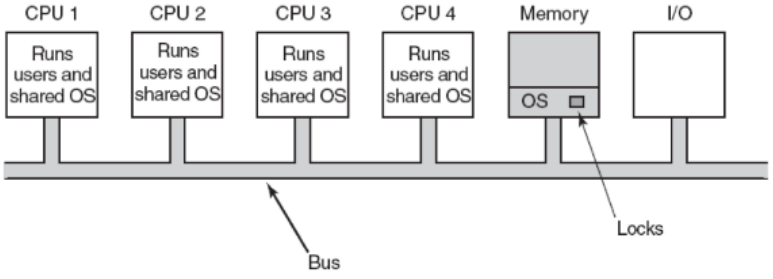
A segunda opción emprega a estratexia **mestre-escravo**. Todas as *chamadas ao sistema redirícense á CPU 1* para ser procesadas nela, onde hai unha *copia do sistema operativo e as súas táboas*. A CPU 1 tamén *pode executar procesos de usuario se ten tempo libre*. As demais encárganse de executar o código, pero *en ningún momento poden falar coa memoria xestionada pola CPU 1*.



Problema: o *rendemento é baixo* porque todos os movementos deben pasar pola CPU mestre.

Multiprocesadores simétricos (SMP)

A terceira opción son os **multiprocesadores simétricos** ou **SMP**. É un modelo moi versátil e amplamente utilizado.



Para todas as CPUs hai *unha única copia do sistema operativo na memoria*. Como todas queren acceder a ela ao mesmo tempo, poden producirse condicións de carreira. Para evitalo, prográmase con **semáforos** ou **mutexes**, así poden *acceder ao núcleo* de forma exclusiva (usando *locks*, por exemplo). Pola outra banda, hai *táboas únicas e compartidas*.