ÍNDEX DE CONTINGUTS

1.- MODELS BÀSICS DE MEMÒRIA COMPARTIDA 2

1.1.- EL MODEL UMA 2
1.2.- EL MODEL NUMA 3
1.3.- EL MODEL CC-NUMA 5
1.4.- EL MODEL COMA 6
1.4.1.- EL MODEL DVSM 7
1.4.2.- SIMPLE COMA 7

1.- MODELS BÀSICS DE MEMÒRIA COMPARTIDA

Podem fer una classificació de les arquitectures dels sistemes multiprocessadors amb memòria compartida en els següents tipus bàsics:

- **UMA**: uniform memory access
- **NUMA**: non-unifrom memory access
- **CC-NUMA**: caché only memory architecture
- **COMA**: cache-only memory access

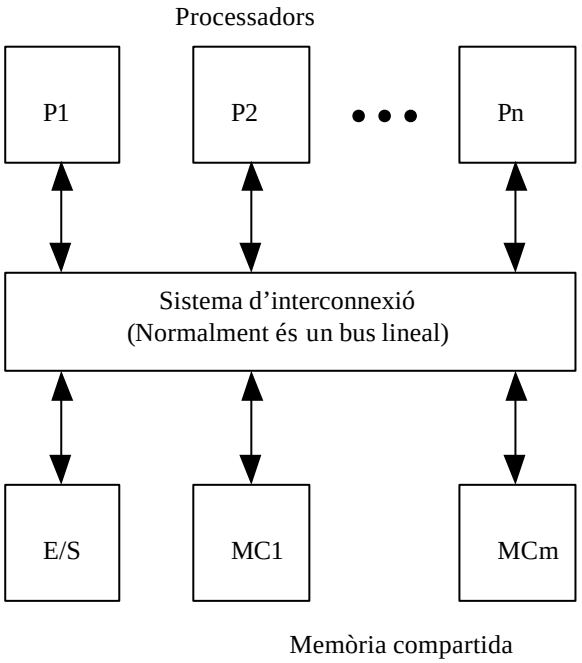
1.1.- EL MODEL UMA

En aquesta arquitectura la memòria física es comparteix de forma uniforme per tots els processadors. No hi ha memòria local a cada processador.

El temps d'accés per cada processador a memòria sempre és el mateix, independentment d'on estigui situada la paraula que s'accedeix. En aquest model cada processador ha d'utilitzar una caché pròpia.

Els multiprocessadors que segueixen aquest model estan fortament acoblats, degut a l'alt grau de compartició de recursos.

En el següent esquema podem veure com està compartida la memòria amb UMA:



Com es pot veure, el bus també s'utilitza per a compartir perifèrics.

Aquest model de microprocessador és adequat per aplicacions de propòsit general amb múltiples usuaris. Pot ser utilitzat per accelerar l'execució de programes grossos en aplicacions on el temps és crític.

Per a coordinar events paral·lels, la sincronització i comunicació entre processadors es fa utilitzant variables compartides en la memòria comú.

Quan tots els processadors tenen accés als dispostius perifèrics el sistema s'anomena ***multiprocessador simètric***. En aquest cas tots els processadors són capaços d'executar tan rutines de kernel del S.O. i rutines de servei a E/S com programa d'usuari.

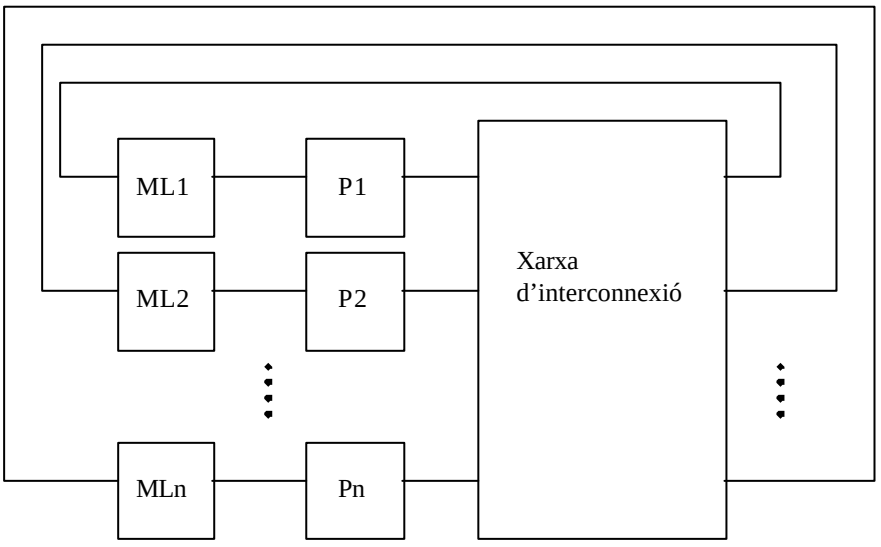
Els ***multiprocessadors asimètrics*** són aquells on només un subconjunt de processadors són capaços d'executar el S.O. i les rutines d'E/S. Tots els processadors que pertanyen a aquest grup són els *mestres*. La resta de microprocessadors només poden executar codi d'usuari sota la supervisió d'un mestre.

1.2.- EL MODEL NUMA

NUMA és un sistema de memòria compartida on el temps d'accés varia segons la localització de la paraula de memòria.

La memòria compartida està distribuïda físicament a tots els processadors en forma de memòries locals. La col·lecció de totes les memòries locals forma un espai d'adreces global accessible per tots els processadors.

En el següent esquema tenim la configuració bàsica del model NUMA:



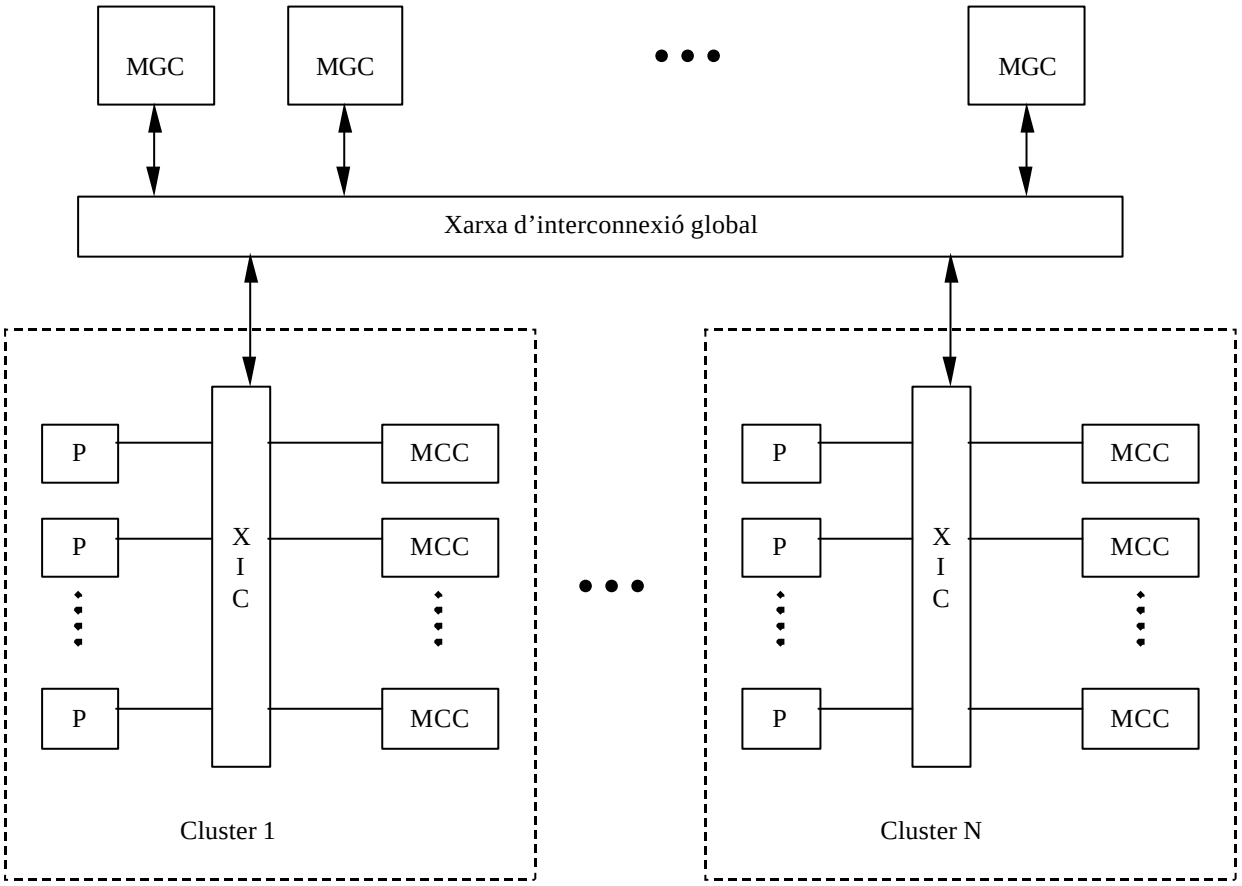
Els blocs MLi són les memòries locals de cada processadors (representats pels blocs Pi).

Com es pot preveure, l'accés per part d'un processador a la seva memòria local és més ràpid que l'accés a la memòria remota (memòries locals d'altres processadors), degut al retard afegit d'accedir a través la xarxa d'interconnexió.

Existeix un model considerat com a NUMA més complex que l'exposat fins ara. Consisteix en afegir memòria compartida globalment.

En aquest cas hi ha tres nivells de memòria principal amb els seus corresponents temps d'accés. La memòria més ràpidament accessible és la local a cada processador. La següent és la memòria global i la més lenta és la memòria remota.

Aquest model sol agrupar processadors en diferents ***clusters*** de manera que cada *cluster* és per sí mateix un sistema multiprocessador UMA o NUMA. Els clusters estan connectats a mòduls de memòria compartida globals. El sistema resultant es considera un model NUMA. Vegem un esquema:



MGC: memòria global compartida
XIC: xarxa d'interconnexió del cluster
MCC: memòria compartida del cluster

Tots els clusters tenen accés a la memòria compartida global. El temps d'accés per part d'un processador a un mòdul de memòria compartida del seu cluster és menor

que el temps d'accedir un mòdul de memòria global. I el temps d'accedir un mòdul de memòria d'un altre cluster és el més elevat.

Aquesta arquitectura permet definir drets d'accés entre clusters.

1.3.- EL MODEL CC-NUMA

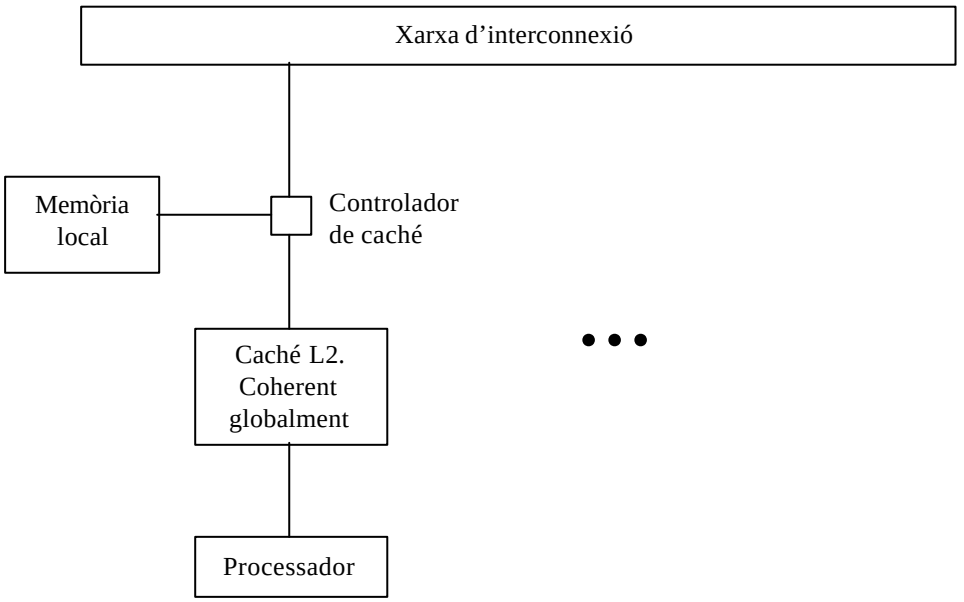
Com hem vist en el model NUMA, la memòria física del sistema es distribueix a tots els processadors de manera que una petita porció de la memòria total és local a cadascun d'ells.

L'accés a la memòria local requereix un temps més petit que accedir a la memòria d'un altre processador. Per tal d'intentar reduir el temps d'accés a la memòria remota es va definir CC-NUMA, que consisteix en utilitzar cachés de segon nivell.

Això, nobstant, introdueix el problema de mantenir coherència en els continguts de les cachés a nivell global del sistema. Aquesta tasca es realitzarà mitjançant hardware.

Un processador només pot tenir emmagatzemades dades remotes en la seva caché, mai en el seu mòdul de memòria local. Això restringeix la quantitat de dades de nodes remots que pot tenir replicades un determinat node en un instant de temps.

Per a les cachés, es sol utilitzar memòria SRAM (estàtica).



1.4.- EL MODEL COMA

Model utilitzat quan només s'utilitza memòria de tipus caché. El podem considerar com un cas especial de NUMA on les memòries principals distribuïdes es converteixen en cachés. Per tant, la memòria de cada processador no està jerarquitzada. El conjunt de totes les memòries cachés formen un espai global d'adreces.

L'accés a cachés d'altres processadors està controlada per els ***directoris distribuïts de caché***.

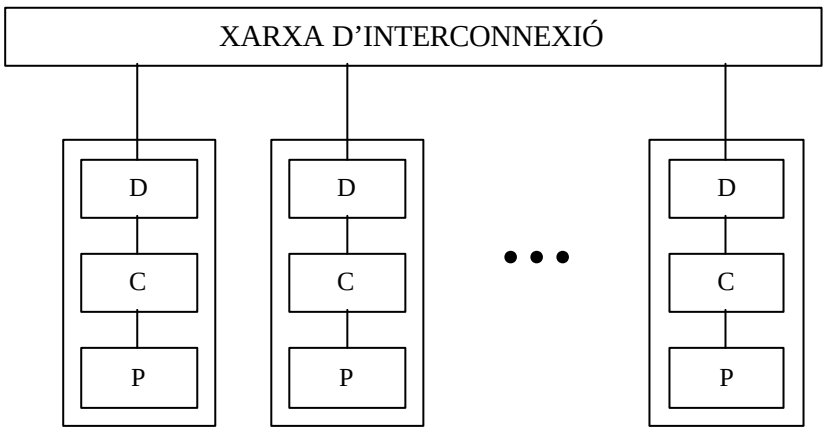
En el model COMA, la memòria local de cada processador es converteix en una gran memòria caché.

Les adreces físiques llançades per un processador són transformades mitjançant una funció de hashing cap a una adreça física real que apunta a un conjunt de línies de caché que són les que poden contenir la dada buscada. En cada caché hi ha incloses tags i comparadors per determinar quina línia de caché és la que conté la dada.

L'adreça que produeix un processador només és un identificador de la dada buscada, però no diu res de la posició física on està ubicada. Una mateixa dada pot estar ubicada en vàries cachés de diferents nodes i pot ser replicada quan faci falta. D'aquesta manera, una dada és "atreta" cap a la memòria local del node en què el processador la necessita. Com que una dada no té una localització per defecte s'ha d'anar en compte a l'hora de fer un reemplaçament en una de les cachés, ja que si es vol eliminar una dada s'ha de comprovar que no sigui l'última còpia que en queda en tot el conjunt global de memòries cachés.

El directori de caché inclòs en cada node s'encarrega d'implementar el control que permet que les cachés permaneixin coherents, sense la necessitat de la intervenció del sistema operatiu. Cada directori ens diu quins processadors tenen una còpia o no de cadascuna de les línies de la caché.

Esquema del model COMA:



D: directori de caché
C: caché
P: processador

1.4.1.- EL MODEL DVSM

El model de *memòria compartida virtual distribuïda* es basa en mantenir la coherència de les dades entre processadors purament per software.

DVSM utilitza la MMU (Memory Management Unit) dels processadors per a detectar i iniciar accions per mantenir la coherència. S'utilitzen missatges a través la xarxa d'interconnexió (com poden ser paquets en una Ethernet) per a aconseguir el seu propòsit.

La major part dels sistemes basats en DVSM utilitzen l'arquitectura COMA, ja que la memòria local de cada processador es tracta com si fós una caché, en què les unitats de dades (pàgines) s'assignen, s'invaliden, es mouen i es repliquen de node a node.

Aquest sistema orientat a control per software té una sèrie d'avantatges sobre el model orientat a hardware que és COMA. La MMU del processador permet que una pàgina de memòria virtual es pugui mapejar a qualsevol pàgina física de memòria en el node local. Això permet que es pugui construir un sistema amb plena assosciativitat, mentre que COMA només pot aconseguir o bé un mapeig directe o una assosciativitat limitada a conjunts.

Com que DVSM s'implementa amb software es poden utilitzar algoritmes de reemplaçament i assignació molt més complexes que els que es poden implementar per hardware.

Existeixen, però, tres problemes bàsics en el sistema DVMS. El primer és el gran tamany de les pàgines (de 8kbytes o més, per exemple), ja que això provoca que el grau de memòria realment compartida disminueixi. El segon problema és la llarga latència de temps assosciada amb una fallada de pàgina, ja que el temps per a generar i transmetre els missatges necessaris per ser enviats a través d'una Ethernet, per exemple, són força grans. El tercer problema, i el més important, és que un processador també ha d'atendre el tràfic de manteniment de coherència emès pels altres processadors.

1.4.2.- SIMPLE COMA

Aquest model és un híbrid entre COMA i el DVSM ja que es basa en utilitzar la MMU dels processadors a més d'una mínima part hardware per a mantenir la coherència i emplaçament de les dades en la memòria. Això permet reduir l'alta complexitat del hardware utilitzat en COMA o CC-NUMA.

Control de l'espai en la memòria local

La MMU s'utilitza per a situar una dada en la memòria local quan una aplicació presenta una adreça virtual. Si la dada buscada no està en la memòria local, la MMU activa una excepció de tal forma que passa a ser responsabilitat del software de trobar un lloc lliure en la memòria local (obligant a fer un reemplaçament en cas que estigui plena) per a poder acollir-la. Gràcies a aquesta solució software (com amb DVSM) s'aconsegueix una assosciativitat de la caché local total (qualsevol dada pot estar

guardada en qualsevol posició de la memòria independentment de l'adreça virtual que té assosciada).

La MMU també permet eliminar la funció de hashing i la verificació de tags per hardware que s'utilitza amb COMA, ja que la taula de pàgines de la MMU fa aquestes funcions.

Manteniment de la coherència

La manera amb què les dades es mantenen coherents amb *Simple COMA* difereix del model DVSM en què no es fa ni a nivell de pàgina ni per software. Aquí es fa totalment per hardware, amb un grau de granularitat molt fi i sense la intervenció del processador.

Per a aconseguir una granularitat fina, cada pàgina de cada memòria local es divideix en subparts del mateix tamany que cada línia de caché.

Quan es va a buscar una dada en la memòria local, es llegeix una línia de la caché, però en paral·lel, es comproven les unitats de coherència (incloses per cada línia i que ens permeten saber si les dades que contenen són vàlides o no) per saber si realment aquest accés s'ha de finalitzar o no cal ja que la dada buscada ha estat modificada o no hi és. Tot això es realitza per hardware i no provoca un increment en el temps d'accés.

En cas que la dada buscada no es pugui llegir de la memòria local (perquè és invàlida) el hardware s'encarrega d'aconseguir la còpia de la dada vàlida que estarà en ulgun altre node.