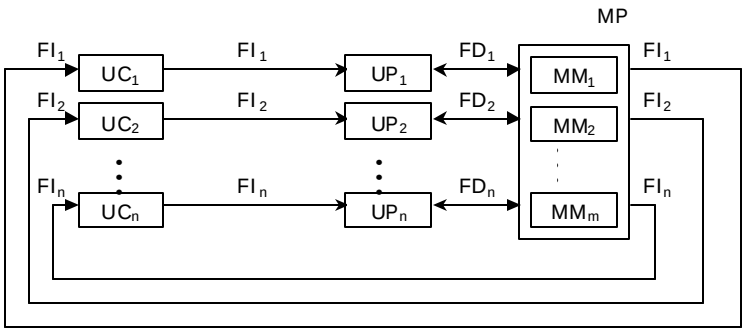
Arhitecturi pentru paralelismul la nivel de proces

Asist. Felician ALECU
Catedra de Informatica Economica, A.S.E. Bucuresti

Most multiprocessor systems can be classified as MIMD computers according to the multiplicity of instruction and data streams. A MIMD computer is tightly coupled if the degree of interactions amoung the processors is high. Otherwise, we consider the paralel system as being loosely coupled. Almost all commercial MIMD computers are loosely coupled systems. The MIMD computers can be categorized in terms of two aspects: wheather the memory is physically centralized or distributed and whether or not the address space is shared.
Keywords: *parallel computer architectures, MIMD computers, tightly and loosely coupled parallel systems, interconnection networks, message passing.*

Arhitectura MIMD (Multiple Instruction stream, Multiple Data stream) reprezinta cel mai general model de sistem paralel, majoritatea sistemelor multiprocesor fiind construite pe principiile acestei arhitecturi (figura 1). Calculatoarele MIMD implementeaza paralelismul la nivel de proces. Pentru ducerea la bun sfârsit a unei sarcini de calcul, procesoarele sistemului paralel comunica si se sincronizeaza între ele.



unde

- UC - unitate de control
- UP – unitate de procesare
- MM – modul de memorie
- MP – memorie partajata;
- FI – flux de instructiuni;
- FD – flux de date.

Fig. 1. Calculator MIMD

Calculatoarele MIMD reprezinta o clasa foarte vasta de calculatoare paralele care se pot clasifica în functie de doua criterii importante:

- tipul memoriei utilizate: memorie partajata (centralizata) sau distribuita;
- spatiul de adresare folosit: comun (partajat) sau individual.

Din combinarea acestor criterii rezulta patru clase arhitecturale pentru paralelismul la nivel de proces din care doar trei au aplicabilitate practica (Tabelul 1). În momentul de fata nu exista nici un sistem de calcul paralel care sa poata fi inclus în clasa multicalculatoarelor cu spatiu de adresa unic.

Multiprocesoarele cu memorie partajata centralizata mai sunt denumite si multiprocesoare simetrice (*SMP – Symmetric Multiprocessor*). Multicalculatoarele cu memorie distribuita utilizeaza pentru comunicatia dintre procesoare mecanismul numit transfer de mesaje. Aceste sisteme mai sunt cunoscute sub denumirea de MPP (*Massively Parallel Processors*).

Indiferent de modul de organizare a spatiului de adresa, un rol foarte important în obtinerea unor performante deosebite îl are reteaua

de interconectare care asigura suportul fizic necesar pentru realizarea comunicatiei între procesoare. Retele de interconectare pot fi statice sau dinamice si ele asigura transferul de date si de mesaje între procesoarele sistemului.

Schimbul de date între doua procesoare ale sistemului este o operatiune mare consumatoare de timp care poate genera inactivitate la nivelul elementelor de procesare. Din acest motiv, algoritmii paraleli ar trebui sa fie proiectati astfel încât majoritatea acceselor la memorie sa nu necesite transportul datelor solicitate prin reteaua de interconectare.

Tabelul 1 – Clasificarea calculatoarelor MIMD

	Memorie partajata (multiprocesoare)	***Memorie distribuita (multicalculatoare)***
Spatiu comun de adresare	UMA (SMP)	–
Spatiu individual de adresare	NUMA COMA CC-NUMA	MPP

1. Arhitecturi cu memorie partajata

Calculatoarele cu memorie partajata dispun de un spatiu comun de memorie care poate fi accesat de toate procesoarele din sistem. Comunicatia între procesoare se realizeaza prin intermediul datelor stocate în spatiul unic de memorie partajata, toate procesoarele având acces la întreaga memorie.

Utilizarea unui spatiu unic de memorie poate conduce la conflicte de acces la memorie atunci când mai multe procesoare încearca sa utilizeze concomitent aceeasi zona de memorie sau când doresc sa utilizeze o variabila partajata la care un alt procesor are acces exclusiv. Prevenirea si solutionarea conflictelor generate de accesul concurent al mai multor procesoare la memoria partajata se realizeaza cu ajutorul mecanismelor de excludere mutuala si a celor de sincronizare între procese

Calculatoarele MIMD cu spatiu unic de memorie au un cost relativ scazut si din acest motiv se bucura de o popularitate ridicata. Aceste calculatoare mai sunt cunoscute sub numele de *multiprocesoare* si sunt considerate a fi sisteme puternic cuplate datorita gradului ridicat de partajare a resurselor. Exista mai multe modele de calculatoarele MIMD cu memorie partajata:

- UMA (*Uniform Memory Access*);
- NUMA (*Non-Uniform Memory Access*);
- COMA (*Cache-Only Memory Access*);
- CC-NUMA (*Cache-Coherent Non-Uniform Access Memory*).

Functionarea întregului sistem paralel este coordonata de catre un singur sistem de operare care se ocupa de planificarea în executie a proceselor pe procesoarele disponibile în sistem în conditiile în care se doreste o încarcare cât mai echilibrata a acestora.

1.1. Modelul UMA

Memoria fizica a calculatoarelor de tip UMA este compusa din mai multe blocuri care formeaza un spatiu unic de adresare. Procesoarele acceseaza blocurile de memorie prin intermediul retelei de interconectare (figura 2).

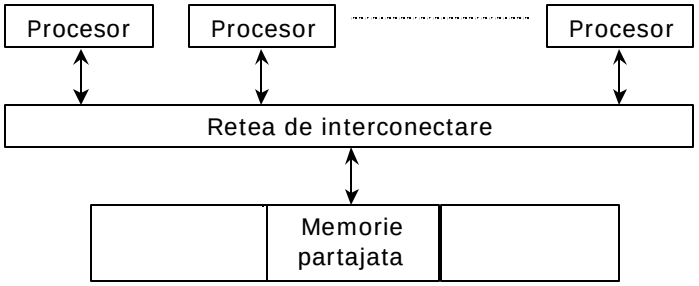


Fig. 2. Arhitectura UMA

Principala caracteristica a calculatoarelor UMA este ca toate procesoarele au timp de acces egal la memoria partajata. Însa, tocmai datorita acestui timp uniform de acces, numarul procesoarelor utilizate în cadrul sistemului paralel nu poate fi foarte mare deoarece cresterea numarului de procesoare din sistem atrage dupa sine o probabilitate mai mare de conflict în ceea ce priveste accesarea concurenta a aceleiasi zone de memorie. Într-un astfel de caz se vor înregistra timpi de asteptare care vor conduce la o utilizare ineficienta a procesoarelor si la o încarcare neechilibrata a acestora.

În plus, pentru obtinerea unor performante deosebite, reteaua de comunicatie va trebui sa ofere o latime de banda suficient de mare deoarece fiecare procesor poate necesita accesul la memoria partajata ceea ce presupune transportul datelor solicitate din memoria comuna catre procesorul care a formulat cererea.

O îmbunatatire adusa modelului UMA o reprezinta utilizarea memoriilor cache la nivelul elementelor de procesare (figura 3). Fiecare procesor va avea asociata o astfel de memorie. Atunci când un procesor solicita date, mai întâi se verifica daca aceste date se gasesc deja în memoria cache locala. În caz afirmativ, acestea se transfera foarte rapid procesorului fara a accesa memoria partajata. Astfel, problema conflictelor de acces concurent la spatiul unic de memorie se rezolva cel putin partial. Mult mai putine cereri vor ajunge sa acceseze memoria partajata si din acest motiv probabilitatea de aparitie a conflictelor devine mult mai mica.

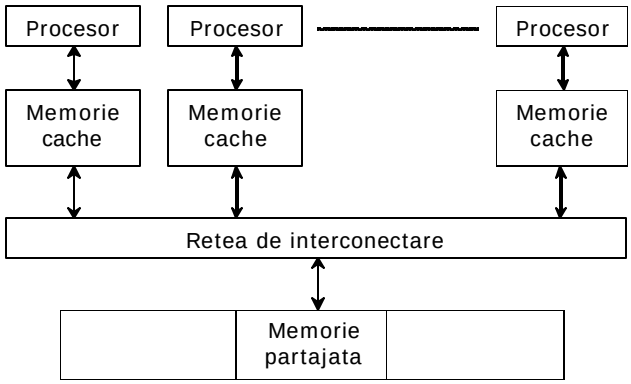


Fig. 3. Arhitectura UMA cu memorii cache locale

Utilizarea memoriilor cache presupune însa si asigurarea consistentei acestora. În momentul în care anumite date stocate în memoria globala se modifica, aceste modificari trebuie sa fie semnalate tuturor memoriile cache care stocheaza copii ale datelor respective. Mai mult, atunci când se schimba continutul datelor de la nivelul unei memorii cache este nevoie ca modificarile sa fie efectuate în toate memoriile cache care contin copii ale datelor actualizate. Mentinerea coerentei memoriei cache se poate realiza atât prin mecanisme software cât si prin mecanisme hardware.

Modelele cu acces ne uniform la memorie încearca sa evite dezavantajele întâlnite la calculatoarele de tip UMA. Arhitecturile DSM (*Distributed Shared Memory*) reprezinta sisteme cu memorie partajata dar care este distribuita din punct de vedere fizic. Astfel, procesoarele au asociate memorii locale care formeaza împreuna o memorie globala ce poate fi accesata direct de catre toate elementele de procesare. Timpul de acces la locatiile de memorie nu mai este uniform, el depinzând de localizarea fizica a datelor ce se doresc a fi transferate.

1.2. Modelul NUMA

În cazul acestui model, memoria partajata este distribuita fizic procesoarelor din sistem (figura 4). Astfel, fiecare element de procesare al unui calculator de tip NUMA are asociata o memorie locala. Toate aceste memorii locale formeaza un spatiu unic de adresa. Din acest motiv, timpul de acces la memorie este

neuniform deoarece accesul la datele aflate în memoria locala este foarte rapid în timp ce accesul la date aflate în memoria locala a altor elemente de procesare va fi mult mai lent datorita faptului ca datele respective vor fi transportate prin reteaua de comunicatie.

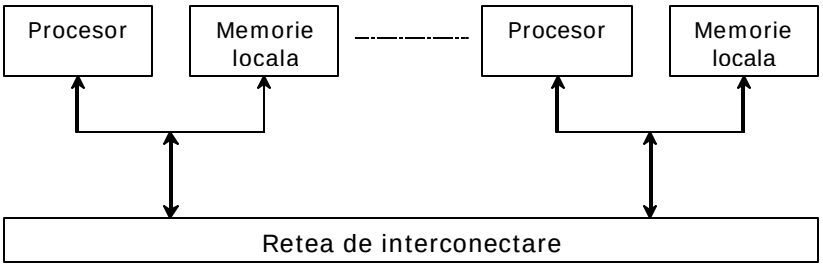


Fig. 4. Arhitectura NUMA

Se observa ca sistemele NUMA sunt foarte asemanatoare cu sistemele distribuite din moment ce fiecare procesor dispune de o memorie locala iar toate aceste module locale formeaza un spatiu unic de adresare.

1.3. Modelul COMA

În cazul modelului COMA (*Cache-Only Memory Access*), spatiul global de adrese este format din memorii cache locale fiecarui procesor. Atunci când un procesor are nevoie de anumite date, acestea sunt mai întâi aduse în memoria cache de catre controllerul de memorie dupa care acestea vor fi foarte rapid accesate (figura 5).

Se poate remarca cu usurinta faptul ca nu mai exista conceptul de memorie globala. Controler-ul de memorie are responsabilitatea de a încarca în memoriile cache locale datele ce urmeaza a fi accesate de catre procesoare. Încarcarea datelor presupune localizarea acestora si transportul prin intermediul retelei de interconectare. Prin folosirea memoriile cache locale, probabilitatea ca un procesor sa acceseze majoritar date numai din memoria locala creste foarte mult.

Totusi, folosirea memoriilor cache pe post de memorii locale atrage dupa sine si o serie de probleme. Atunci când se încearca aducerea unui bloc de date într-o memorie cache integral ocupata apare necesitatea mutarii unui alt bloc de date pentru a se crea spatiul necesar memorarii noului bloc care va fi accesat de catre procesor. Însa, datorita faptului ca nu mai exista o memorie globala în care sa se scrie acest bloc, va fi nevoie ca datele continute sa fie mutate în memoria cache a unui alt procesor. Aceste date însa nu vor fi niciodata accesate de procesorul care foloseste memoria respectiva ceea ce atrage dupa sine o scadere a eficientei de utilizare a sistemului. În plus, operatia de transfer a blocului de date dintr-o memorie cache în alta este si ea consumatoare de timpi procesor din moment ce datele trebuie sa circule prin reteaua de interconectare. Acesti timpi morti atrag dupa sine o utilizare ineficienta si o încarcare neechilibrata a procesoarelor.

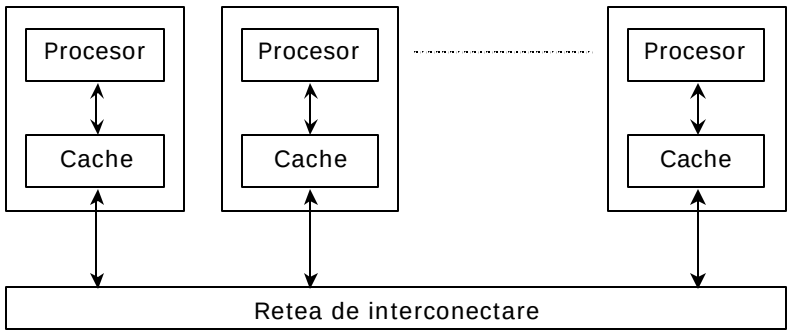


Fig. 5. Arhitectura COMA

O alta problema o reprezinta asigurarea coerentei memoriilor cache în contextul în care operatiile de mutare a blocurilor devin din ce în ce mai complicate atunci când numarul de

procesoare din sistem este din ce în ce mai mare. Controler-ul de memorie trebuie sa asigure gestiunea nu numai a datelor ce vor fi aduse în memoriile cache ci si a blocurilor care au fost mutate dintr-o memorie locala în alta.

1.4. Modelul CC-NUMA

Modelul CC-NUMA combina modul în care este construit spatiul unic de adresa la calculatoarele de tip NUMA si la cele de tip COMA. Fiecare element de procesare al unui calculator de tip CC-NUMA dispune atât de o memorie locala cât si de o memorie cache asociata (figura 6).

Ca si în cazul modelului NUMA, accesul la datele stocate în memoria locala proprie este foarte rapid. Accesarea unor date aflate în memoria locala a unui alt procesor implica aducerea acestora în memoria locala proprie. Transferul de date folosind reteaua de comunicatie atrage dupa sine întârzieri deoarece nu este la fel de rapid precum accesarea memoriilor locale .

Existenta memoriilor cache are ca efect reducerea traficului prin reteaua de comunicatie. Datele solicitate de catre un procesor sunt mai întâi cautate în memoriile locale dupa care sunt aduse în memoria cache în cazul în care acestea se gasesc în memoria locala a unui alt procesor. La accesarile ulterioare se va folosi copia datelor respective stocata în memoria cache, deci nu va mai fi nevoie ca datele sa fie din nou transferate prin reteaua de interconectare.

Utilizarea memoriile cache atrage dupa sine necesitatea implementarii unor mecanisme software sau hardware care sa asigure consistenta memoriilor cache. La modificarea unor date aflate în memoria locala a unui procesor este nevoie sa se actualizeze toate copiile acestor date aflate în memoriile cache. Aceste mecanisme care asigura coerenta memoriilor cache contribuie la cresterea traficului efectuat în cadrul retelei de interconectare.

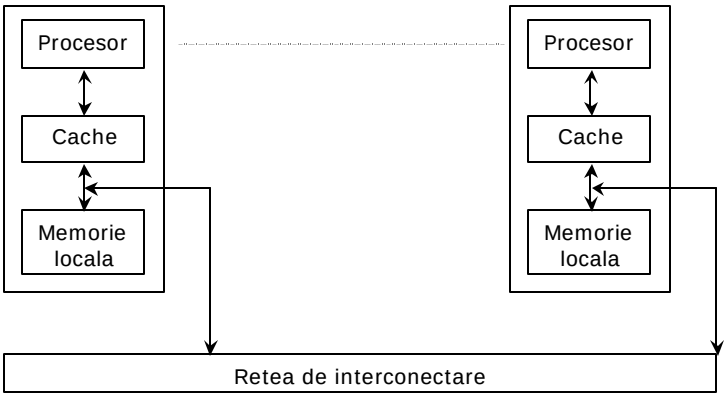


Fig. 6. Arhitectura CC-NUMA

Principalul dezavantaj al arhitecturilor MIMD cu memorie distribuita si cu spatiu individual de adresare (NUMA, COMA, CC-NUMA) îl reprezinta întârzierile care pot sa apara atunci când un procesor doreste sa acceseze date care se afla în memoria locala a unui alt element de procesare. Aceste întârzieri se datoreaza faptului ca procesorul respectiv va astepta ca datele solicitate sa fie transportate în memoria sa locala. Pentru minimizarea timpilor de inactivitate, procesorul în cauza poate efectua o comutare de procese atunci când procesul activ este blocat în asteptarea datelor solicitate. În general comutarea proceselor este o operatiune complexa care implica costuri importante care pot influenta performantele sistemului.

Eficienta operatiei de comutare poate fi substantial îmbunatatita prin utilizarea firelor de executie (thread-uri). Un proces poate fi împartit în mai multe fire de executie care comunica între ele prin accesul la spatiul de adresa alocat procesului, pe care îl partajeaza. Din acest motiv, firele de executie reprezinta un mecanism eficient de exploatare a concurentei programelor. Un fir de executie reprezinta cea mai mica entitate de planificare în sistemele bazate pe thread-uri. Comuta-

rea între doua fire de executie care apartin aceluiasi proces este o operatie foarte rapida care nu atrage dupa sine costuri suplimentare semnificative. Astfel, procesul activ este suspendat pâna în momentul în care datele solicitate au fost aduse în memoria locala. În tot acest timp, un alt fir de executie al aceluiasi proces este executat. Aceasta comutare între fire de executie apartinând aceluiasi proces conduce la scaderea timpilor de inactivitate si deci implicit la cresterea eficientei cu care este utilizat sistemul paralel.

2. Arhitecturi cu transfer de mesaje

În cazul calculatoarelor cu memorie distribuita, fiecare procesor are asociata o memorie locala pe care o poate accesa în mod direct (figura 7). Pentru a avea acces la memoria locala a unui alt procesor se foloseste mecanismul numit schimb de mesaje (*message-passing*). Procesoarele pot interactiona între ele doar prin schimb de mesaje din moment ce nu exista locatii comune de memorie care sa fie folosite la schimbul de date dintre doua procesoare. .

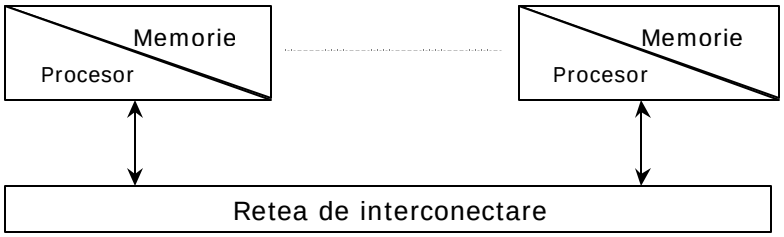


Fig. 7. Arhitectura MIMD cu transfer de mesaje

Calculatoarele MIMD cu memorie distribuita mai sunt denumite si *multicalculatoare* si ele reprezinta sisteme slab cuplate datorita gradului redus de partajare a resurselor. Principalul avantaj al acestor sisteme îl reprezinta scalabilitatea, adica posibilitatea extinderii prin conectarea unor noi procesoare. Cresterea numarului de procesoare nu atrage dupa sine probleme legate de accesul concurent la memorie deoarece majoritatea apelurilor sunt catre memoria locala. Însa, transferul de mesaje presupune schimburi de date care pot aglomera reteaua. Transferul de date între doua procesoare este o operatie mare consumatoare de timp. Din acest motiv, performantele retelei de interconectare pot influenta în mod hotarâtor eficienta cu care este utilizat sistemul paralel. În plus, este foarte greu ca sarcina de calcul distribuita procesoarelor sa fie echilibrata. Sistemul de operare ruleaza în fiecare nod si din acest motiv putem vedea întregul sistem ca fiind format din statii de lucru care comunica prin transfer de mesaje. Complexitatea acestor sisteme este foarte mare datorita faptului ca într-un nod de procesare individual se poate gasi un simplu procesor sau un întreg sistem de tip UMA.

Bibliografie

[Chi95] I. Chiorean, *Calculul parallel - fundamente*, Editura Albastra, Cluj-Napoca, 1995

[Dod02] Gh. Dodescu, B. Oancea, M. Raceanu, *Procesare paralela*, Editura Economica, Bucuresti, 2002

[Tan98] A. S. Tanenbaum, *Retele de calculatoare*, Computer Press Agora, Bucuresti, 1998

[Tan99] A. S. Tanenbaum, *Organizarea Structurata a Calculatoarelor*, Computer Press Agora, Bucuresti, 1999