UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

POLITEHNICA DIN BUCUREȘTI
UNIVERSITATEA
1818

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Arhitectura Sistemelor de Calcul

6. Fluxul de date

Exemplu de Utilizare – SISD

- Problema: A[n, n], B[n, n], C = A x B
- Pe structuri de calcul SISD – 3 for-uri:

```
for i = 0 to n – 1
   for k = 0 to n – 1
        c_ik = 0
        for j = 0 to n – 1
              c_ik = c_ik + a_ij * b_jk
        end j loop
   end k loop
end i loop
```

- Complexitatea acestui algoritm este… $O(n^3)$! → nesatisfacatoare! (mai ales daca n e mare…)

Exemplu de Utilizare – SIMD

- Aceeasi problema: A[n, n], B[n, n], C = A x B
- Avem n procesoare & toate executa aceeasi instructiune odata → in fiecare calcul se calculeaza cate o linie si nu doar un element
- Considerand ($0 \leq k \leq n - 1$) → se opereaza pentru toti indicii k simultan, adica se calculeaza pe linii

```
for i = 0 to n – 1
   c_ik = 0 (0 ≤ k ≤ n – 1)
   for j = 0 to n – 1
       c_ik = c_ik + a_ij * b_jk (0 ≤ k ≤ n – 1)
   end j loop
end i loop
```

- Complexitatea acestui algoritm este… $O(n^2)$! → considerabil mai bine ca in cazul SISD

Comentarii & Observatii – SIMD

- Fiecare element al matricei produs C, este o suma ce se efectueaza secvential
- Cele n sume se calculeaza apoi in paralel !?
- a_{ij} NU depinde de k → accesul la aceasta memorie se face aproximativ secvential → NU e chiar “sumele se calculeaza in paralel”!
- Solutia: structurile SIMD trebuie sa dispuna de o instructiune de Broadcast & o retea de comutare (RC) ce sa asigure acest Broadcast
- P_j citeste a_{ij} prin RC (constanta a_{ij} e difuzata catre toate procesoarele)
- Concluzie SIMD: probleme mari la comunicatiile inter-procesoare & accesul si organizarea datelor!

Exemplu de Utilizare – MIMD

- Aceeasi problema: A[n, n], B[n, n], C = A x B
- O UCmd/P trebuie sa preia functia de master: organizare si control + partajarea calculelor pe procesoare individuale
- Conway a propus o metoda cu 2 primitive: FORK & JOIN
  - FORK: desface un FI in n FI executabile simultan pe proc indep
  - JOIN: reuneste n FI intr-unul singur cand cele n FI s-au terminat

```
for k = 0 to n – 2 (nu si pt el insusi)
   FORK Adr
end k loop
Adr:
   for i = 0 to n – 1
       c_ik = 0 (0 ≤ k ≤ n – 1) – pe coloane k fix
       for j = 0 to n – 1
              c_ik = c_ik + a_ij * b_jk (0 ≤ k ≤ n – 1)
       end j loop
   end i loop
   JOIN
```

Complexitatea este… $O(n^2)$!

→ la fel ca SIMD

Comentarii & Observatii – MIMD

- In mod deliberat programul pt SIMD a fost scris a.i. actiunile P-urilor sa simuleze actiunile din structura MIMD
- Fiecare P_j calculeaza un C_{ik} in paralel
- Diferente SIMD/MIMD:
  - In SIMD procesoarele se sincronizau instructiune (It) cu It
  - In MIMD nu exista (neaparat) sincronizari intre FI ale P-urilor din structura
  - La SIMD se calculeaza elementele lui C pe linie si coloana (FI unic)
  - La MIMD orice procesoare pot calcula orice elemente din C (fiecare P are un FI propriu!)
- Castigul e acelasi; se pot folosi mai multe perechi FORK/JOIN
- Concluzie MIMD: mai usor de programat & utilizat decat SIMD – dar, mai scump! (Totul se plateste…)