

Cristina POPÎRLAN

UNIVERSITATEA DIN CRAIOVA



Centrul de Cercetare în Inteligență Aplicată – „Nicolae
Țândăreanu”

Research Center for Applied Intelligence – „Nicolae
Țândăreanu”

www.rcai.eu

Colecția « Computer Science »

Coordonatori colecție:

Daniela Dănciulescu – Director RCAI – Universitatea din Craiova
Gabriel Stoian – Universitatea din Craiova

Comitetul științific:

Gheorghe Grigoraș, Universitatea Alexandru Ioan Cuza din Iași
Viorel Negru, Universitatea de Vest din Timișoara
Petrică Pop Sitar, Universitatea Tehnică din Cluj-Napoca
Mihaela Păun, Academia de Studii Economice din București
Cristian Kevorchian, Universitatea din București
Claudiu-Ionuț Popîrlan, Universitatea din Craiova

Inițiată în 2001, sub egida Centrului de Cercetare în Inteligență Aplicată – „Nicolae Țândăreanu”, colecția „Computer Science” reunește contribuții valoroase – publicații științifice de înaltă ținută, studii, teze de doctorat, etc. – continuând astfel tradiția publicării în volum separat a seriei 100 (lucrările conferinței anuale AIDC – Artificial Intelligence and Digital Communications), a seriei 200 (Computer Science Fundamentals) și a seriei 300 (Research Reports in Artificial Intelligence).

Rămânând fidelă misiunii sale de dezvoltare și promovare a cunoașterii în domeniul științei calculatoarelor și a tehnologiilor digitale, colecția „Computer Science” reprezintă o sursă reală de informare și își propune lărgirea spectrului publicistic prin dezvoltarea de noi serii tematice.

Propunerile pentru publicare se vor adresa comitetului științific la adresa: office@rcai.eu

Cristina POPÎRLAN

Algoritmi paraleli și distribuiți



Editura UNIVERSITARIA
Craiova, 2019

Referenți științifici:
Prof. univ. dr. Daniela Dănciulescu
Lect. univ. dr. Gabriel Stoian

Copyright © 2019 Editura Universitaria
Toate drepturile sunt rezervate Editurii Universitaria

Descrierea CIP a Bibliotecii Naționale a României
POPÎRLAN, CRISTINA

Algoritmi paraleli și distribuiți / Cristina Popîrlan. - Craiova:
Universitaria, 2019
Conține bibliografie
ISBN 978-606-14-1566-3

51

© 2019 by Editura Universitaria

Această carte este protejată prin copyright. Reproducerea integrală sau parțială, multiplicarea prin orice mijloace și sub orice formă, cum ar fi xeroxarea, scanarea, transpunerea în format electronic sau audio, punerea la dispoziția publică, inclusiv prin internet sau prin rețelele de calculatoare, stocarea permanentă sau temporară pe dispozitive sau sisteme cu posibilitatea recuperării informațiilor, cu scop comercial sau gratuit, precum și alte fapte similare săvârșite fără permisiunea scrisă a deținătorului copyrightului reprezintă o încălcare a legislației cu privire la protecția proprietății intelectuale și se pedepsesc penal și/sau civil în conformitate cu legile în vigoare.

Lista de figuri

2.1	Calculator SISD	21
2.2	Calculator MISD	21
2.3	Calculator SIMD	22
2.4	Calculator MIMD	24
2.5	Exemplu de sisteme de conectare	27
2.6	Sistem de conectare	27
2.7	Rețea de comunicare	28
2.8	Mecanismele de comunicație între procese: (a) Comunicația prin variabile partajate (b) Comunicația prin transfer de mesaje	33
2.9	Modelul PCAM de proiectare a algoritmilor paraleli	39
2.10	Exemplu de diagramă de activitate UML	40
2.11	Modelul PRAM cu p procesoare și memorie partajată	60
2.12	Exemplu de graf direct orientat pentru $(x_1 + x_2)(x_2 + x_3)$	65
2.13	Graf direct orientat pentru $(x_1 + x_2)(x_2 + x_3)$	66
2.14	Calcul paralel pentru adunarea a $\frac{16}{15}$ scalari	68
2.15	Algoritm alternativ pentru adunarea paralelă a 16 scalari	69

2.16	Calcul paralel al puterilor unei matrici	70
2.17	Graf de dependență	72
2.18	Graf direct orientat asociat grafului de dependență	73
2.19	Paralelizarea iterației Seidel-Gauss	74
2.20	Paralelizarea iterației Seidel-Gauss - altă ordine de actualizare a componentelor	75
2.21	Graf de dependență pentru un algoritm iterativ	77
2.22	Nodurile colorate ale grafului de dependență	78
2.23	Graf de dependență - exercițiu	80
2.24	Implementarea sincronizării globale	84
2.25	Sincronizarea globală folosind un arbore	85
2.26	Exemplu de dificultate de comunicare	89
2.27	Exemplu de comunicare cu întârziere	90
2.28	Exemplu de comunicare lentă (canal de comunicare lent/ procesor lent)	90
2.29	Diagrama de timp pentru algoritmi sincroni/asincroni	92
3.1	Implementarea metodei substituției inverse	100
3.2	Implementare alternativă pentru substituția inversă	101
3.3	Reducția pară-impară	102
3.4	Convergența metodei Jacobi	110
3.5	Convergența metodei Seidel-Gauss	111
3.6	Graf direct orientat asociat unei matrici stocastice	127
4.1	Exemplu de contracție și pseudocontractie	131

4.2	Teorema de punct fix	133
4.3	Convergența iterației Seidel-Gauss pentru contracții bloc	135
4.4	Funcția Q	137
4.5	Metoda Seidel-Gauss bazată pe funcția Q pentru rezolvarea sistemului $x = T(x)$	138
4.6	Direcții descendente în algorimii gradientului și gradientului scalat	141
4.7	Iterația algoritmului Jacobi neliniar	144
4.8	Iterația algoritmului Seidel-Gauss neliniar	144
4.9	Condiția satisfăcută de proiecția $[x]^+$	146
4.10	Iterația algoritmului gradientului bazat pe proiecții	147
4.11	Iterația algoritmului gradientului scalat bazat pe proiecții	149
4.12	Problema drumului de cost minim în cazul în care avem un ciclu de lungime 0	154
4.13	Relația dintre numărul de iterații și lungimile arcelor pentru algoritmul Bellman-Ford	155
4.14	Numărul de iterații - algoritmul Bellman-Ford	157

Capitolul 1

Introducere

Calculul paralel și distribuit reprezintă un domeniu larg de interes, existând o activitate de cercetare intensă în acest sens motivată de numeroși factori. Întotdeauna a fost nevoie de o soluție de rezolvare a problemelor cu calcule foarte mari, dar, doar recent, tehnologiile avansate au luat în calcul folosirea calculului paralel în rezolvarea acestor probleme ținând cont și de calculatoarele paralele din ce în ce mai puternice.

Sisteme de calcul paralele și distribuite construite și exploatate până în prezent acoperă un spectru de arhitecturi și modele de programare extrem de variate. De fapt, fiecare calculator paralel sau sistem distribuit reprezintă o soluție arhitecturală deosebită de toate celelalte existente, în încercarea de se a obține performanțe cât mai ridicate.

Datorită numărului mare de sisteme paralele și distribuite construite și a numeroaselor diferențieri de arhitectură, este destul de dificil și confuz să fie studiate toate împreună și, ca în multe alte situații, o clasificare în generații, bazată pe

perioada de construcție și pe diferite decizii constructive, poate fi utilă.

Se pot considera mai multe etape în dezvoltarea sistemelor paralele și distribuite.

Primele calculatoare paralele se încadrează în generația de prototipuri de cercetare, care cuprinde mai multe încercări de construire a calculatoarelor paralele, proiectate în special pentru studierea problemelor de bază teoretice și de implementare ale calculului paralel. Cele mai multe astfel de sisteme, construite înainte de 1980, nu au reprezentat un succes comercial propriu-zis, dar au pregătit trecerea la sisteme paralele actuale, performante și disponibile comercial.

Primele generații de calculatoare paralele cuprind calculatoare dezvoltate în ultimele două decenii ale secolului trecut de marile companii producătoare de calculatoare precum Intel, DEC, Sun, HP, Cray, IBM, sau de companii specializate, precum NCUBE. În aceste generații se întâlnesc atât calculatoare SIMD cât și MIMD, dar se observă clar tendința de părăsire a arhitecturii SIMD, și de adoptare aproape unanimă a arhitecturilor MIMD (multiprocesoare și multiculatoare). Dintre aceste calculatoare, merită amintite calculatoarele produse de firma Thinking Machine Corporation (TMC), și anume CM-2 (Connection Machine-2, produs în anul 1985) și CM-5 (produs în anul 1995).

Cei mai mari producători de supercalculatoare sunt marii producători de calculatoare comerciale: IBM (37% - 35%), Hewlet Packard (42% - 23%), Cray (4% - 16%), SGI (3.8% - 6.6%), Dell (3.2% - 2.2%), Sun etc. atât ca număr de supercalculatoare, cât și ca performanțe însumate ale sistemelor instalate.

Dezvoltarea algoritmilor paraleli și distribuți este determinată pe de o parte de legătura dintre nevoile de calcul noi și cele vechi, iar pe de altă parte de

progresul tehnologic.

Aplicațiile care privesc inteligența artificială, calculul simbolic și calculul numeric sunt cele care au jucat un rol important în dezvoltarea arhitecturilor paralele și distribuite.

Nevoia de determinare rapidă a unui număr mare de calcule numerice din cadrul aplicațiilor ce privesc rezolvarea ecuațiilor cu derivate parțiale și a procesării imaginilor a fost ceea ce a dus la dezvoltarea rapidității și a paralelizării mașinilor de calcul.

Recent a crescut interesul pentru aplicarea calculului paralel și distribuit în aplicații precum cele ce privesc analiza, simularea și optimizarea pe scară largă a sistemelor interconectate. Alte exemple de aplicare a calculului paralel sunt: rezolvarea sistemelor de ecuații, programarea matematică și problemele de optimizare. O proprietate comună a acestor probleme este aceea de a putea fi descompuse în subtask-uri.

O importanta ramură în știința calculatoarelor, și anume calculul paralel, s-a dezvoltat din necesitatea existenței unor calculatoare cu performanțe ridicate, capabile să rezolve aplicații cu cerințe de calcul intensiv, iar aceste performanțe nu se pot obține folosind calculatoarele secvențiale. Exemple de astfel de domenii și aplicații sunt:

- **Domeniul științific:** fizica nucleară, simularea fenomenelor naturale (meteorologice, fizica pământului), biochimie (proiectarea de noi medicamente);
- **Domeniul ingineresc:** proiectare în mecanică, electronică, arhitectură;
- **Domeniul economic:** previziuni financiar-bancare, modelări macro-economice.

Calculatoarele secvențiale tradiționale se bazează pe modelul introdus de John von Neumann în anul 1945. Acest model constă dintr-o unitate centrală de procesare (procesorul) și o unitate de memorie, conectate între ele printr-un canal de comunicație pentru transferul datelor. Într-un astfel de calculator, procesorul extrage instrucțiuni și date din memorie, le prelucrează și depune rezultatul înapoi în memorie, oferind posibilități limitate de prelucrare.

Viteza de calcul a unui calculator secvențial este limitată de doi factori:

- viteza de transfer a datelor între procesor și memorie;
- viteza de operare a procesorului.

Proiectanții procesoarelor folosesc diferite tehnici pentru a depăși aceste limitări, prin introducerea paralelismului (concurenței) în interiorul procesoarelor, folosind capacitatea (număr de componente/chip) din ce în ce mai mare a circuitelor VLSI.

Viteza de transfer între procesor și memorie poate fi mărită prin diferite tehnici de organizare a unității de memorie, cum sunt:

- întreteserea memoriei, care conduce la creșterea benzii folosite la comunicația între procesor și memorie;
- memoriile cache, care produc scăderea latenței de comunicație între procesor și memorie.

Întreteserea memoriei se obține prin divizarea unității de memorie într-un număr de blocuri, fiecare conectat la procesor printr-un canal independent. În

acest mod, viteza de transfer a datelor crește prin creșterea numărului de canale între procesor și memorie, deci a benzii de comunicare procesor-memorie.

Cealaltă modalitate de creștere a vitezei de transfer constă în utilizarea unui bloc de memorie de dimensiune mai mică, dar de viteză ridicată, cu rolul de buffer între procesor și memoria principală, de dimensiune mare, dar de viteză mai scăzută. Un astfel de bloc de memorie se numește memorie cache. Este posibil ca un bloc de date să fie extras din memoria principală și încărcat în memoria cache, de unde datele sunt transferate către procesor cu viteză foarte mare, ceea ce produce scăderea latentei de comunicare între procesor și memorie.

Memoriile cache utilizează principiul localității datelor, care stipulează faptul că, dacă o instrucțiune accesează o anumită locație de memorie, atunci instrucțiunile următoare vor accesa, cel mai probabil, locații vecine de memorie.

Creșterea vitezei de operare a procesoarelor se poate obține prin mai multe modalități, printre care cele mai importante sunt:

- Creșterea frecvenței (clock rate) de lucru a procesoarelor.
- Introducerea paralelismului (concurenței) în interiorul procesoarelor prin:
 - execuția pipeline a instrucțiunilor;
 - prelucrarea vectorială pipeline;
 - acceleratoare hardware;
 - alte tehnici de accelerare a execuției instrucțiunilor (hyperthread-ing, multi-core etc).

Viteza de operare a procesoarelor este dată de ciclul de ceas (clock cycle) al procesorului, care este timpul necesar execuției unei operații de bază.

Dar, este binecunoscut faptul că frecvența de ceas nu poate crește infinit, oricâte progrese tehnologice se vor face în continuare, datorită limitării fizice fundamentale, prin care viteza de propagare a semnalelor este limitată la valoarea vitezei luminii în vid (3×10^8 m/s).

Astfel încât pentru creșterea mai puternică a performanțelor procesoarelor se adoptă, pe lângă creșterea frecvenței de ceas, diferite soluții arhitecturale din ce în ce mai complexe, în principal prin introducerea paralelismului în interiorul procesoarelor.

Introducerea paralelismului (concurenței) în interiorul procesoarelor (paralelism implicit) permite creșterea vitezei de execuție prin creșterea complexității circuitelor VLSI ale procesoarelor, în care se pot prevedea mai multe căi de date, mai multe etaje de prelucrare etc.