CONTROLUL DINAMIC AL GRANULARITĂȚII PROCESELOR PARALELE DE SINTEZĂ DE IMAGINE

Dr. Ing. Felicia Ionescu
SIMULTEC S.A., Bucureşti

Rezumat: Lucrarea evidenţiază necesitatea şi modalităţile de asigurare ale controlului dinamic al granularităţii proceselor paralele de sinteză de imagine. Este prezentat un model de distribuire dinamică a sarcinilor de calcul între procesoarele componente ale calculatorului paralel, care reprezintă sistemul de sinteză de imagine în timp real. Acest model include un atribut de definire a complexităţii de calcul a elementelor componente ale bazelor de date grafice, necesar controlului granularităţii programului şi anume timpul de execuţie estimat pentru generarea imaginii fiecărui obiect sau asociere de obiecte. Pentru mascarea timpului de comunicaţie şi de distribuire a sarcinilor de calcul se foloseşte tehnica de execuţie cu contexte multiple, care permite procesoarelor să comute de la un context la altul, atunci când au loc operaţii de comunicaţie. Această tehnică permite identificarea regiunii de execuţie optimă, prin condiţii impuse granularităţii programului. După deducerea experimentală a valorii granularităţii optime, acestă valoare este utilizată pentru partiţionarea dinamică a sarcinilor de calcul distribuite procesoarelor, ca sarcini de calcul cu o granularitate cât mai apropiată de granularitatea optimă.

1. Introducere

Într-un calculator paralel, cu memorie distribuită şi transfer de mesaje, cele mai importante surse ale costului suplimentar de calcul paralel sunt costul comunicaţiilor între procesoare şi încărcarea neechilibrată a acestora. Ambele surse de cost suplimentar depind de granularitatea programului şi impun condiţii divergente asupra acesteia.

Granularitatea unui proces paralel este o măsură a cantităţii de calcul a acestuia şi se defineşte ca fiind timpul mediu de execuţie a procesului, între două puncte de sincronizare [1],[4]. Fiecare proces se execută independent, deci asincron faţă de celelalte procese, iar punctele de sincronizare între acestea sunt punctele în care procesele comunică între ele pentru transmitere de date sau control al secvenţelor de calcul. Granularitatea unui program este valoarea medie a granularităţii proceselor componente.

Sistemul de sinteză de imagine pe care se analizează este realizat sub forma unui calculator paralel, cu memorie distribuită şi transfer de mesaje, iar programul care implementează operaţiile grafice este compus dintr-o mulţime de procese care se execută în paralel, în diferite procesoare ale reţelei.

Sarcina de coordonare eficientă a proceselor paralele este una din cele mai dificile probleme, care necesită o partiţionare a programului, adaptată la caracteristicile constructive ale calculatorul paralel, în ceea ce priveşte puterea de calcul a procesoarelor, topologia de interconectare, caracteristicile comunicaţiei interprocesoare şi dimensiunea reţelei [5]. Pentru obţinerea adaptării între arhitectura paralelă a calculatorului şi programul paralel executat trebuie să fie îndeplinite mai multe cerinţe, printre care şi controlul granularităţii programului, care asigură execuţia eficientă, cu granularitate optimă şi costuri suplimentare minime.

2. Distribuirea proceselor paralele de sintezÍ de imagine

Operaţiile grafice se aplică modelelor obiectelor şi fenomenelor lumii reale, memorate structurat sub forma bazelor de date grafice. Pentru proiectarea sistemului şi analiza performanţelor acestuia, este necesară definirea unui model de reprezentare a bazei de date grafice, precum şi a unui model de distribuire a proceselor paralele în reţea.

A. Modelul de reprezentare a bazei de date grafice

Pentru analiza sintezei de imagine în timp real, se consideră o bază de date grafică orientată pe obiecte, structurată ierarhic, care se poate reprezenta sub forma unui graf aciclic direcţionat B = (O,D), [3],unde:

$$\mathbf{O} = \{ o_i \mid 0 \le i < o, o = |O| \};$$

$$\mathbf{D} = \{ d_{i,j} = (o_i, o_j) \mid o_i, o_j \in O \}.$$

O este mulţimea nodurilor grafului, fiecare nod reprezentând o clasă de obiecte grafice, iar **D** este mulţimea arcelor grafului, care descriu dependenţa între clasele de obiecte grafice. Un nod frunză al grafului reprezintă modelarea prin suprafaţa de frontieră a unui obiect tridimensional elementa; un nod intermediar o_i reprezintă un obiect tridimensional, compus prin asocierea tuturor acelor obiecte componente, care sunt noduri terminale ale arcelor incidente spre exterior din nodul o_i .

Pentru controlul dinamic al granularităţii programului de sinteză de imagine, la crearea structurii bazei de date se estimează şi se memorează, ca atribut ale fiecărei clase de obiecte, timpul de execuţie al fiecărui nod al grafului. La

încărcarea datelor de intrare ale bazei de date se poate estima timpul de execuţie al fiecărui nod frunză, în funcţie de numărul elementelor sale geometrice (poligoane, laturi, vârfuri) şi de atributele de redare ale poligoanelor componente (iluminare, antialiasing, texturare). Pentru celelalte noduri, timpul de execuţie este suma timpilor de execuţie a tuturor nodurilor succesoare ale acestora: $t_i = \Sigma\, t_{i,j}$ şi se obţine printr-o parcurgere în lărgime (breadth-first-search) a grafului bazei de date.

Toate procesele de calcul pentru generarea imaginii unui obiect tridimensional elementar, deci a unui obiect descris de un nod frunză în graful bazei de date, sunt grupate într-un cluster de procese şi reprezintă o sarcină de calcul nedivizibilă, care va fi executată întotdeauna pe un singur procesor din reţea [2].

Considerând că, din fiecare punct de observare a scenei modelată de baza de date, sunt vizibile un număr k de obiecte tridimensionale elementare, cu un timp de execuţie mediu (timpul necesar pentru generarea imaginii fiecărui obiect) t_k, atunci timpul de execuţie secvenţială pentru generarea unei imagini este $T_S = k * t_k$.

Timpul de execuţie paralelă T_P necesar pentru generarea unei imagini într-un sistem paralel, constituit din p procesoare, este compus din timpul de generare a imaginii tuturor obiectelor, deci T_S, la care se adaugă un cost suplimentar, T_O, divizate cu numărul p de procesoare:

$$T_P = (T_S + T_O) / p.$$

Costul suplimentar de execuţie paralelă T_O are mai multe surse, dintre care cele mai importante sunt costul comunicaţiilor între procesoare şi încărcarea neechilibrată a acestora. Ambele surse de cost suplimentar depind de granularitatea programului şi impun condiţii divergente asupra acesteia. Pentru obţinerea eficienţei maxime a sistemului, se deduce, teoretic şi experimental, o valoare optimă a granularităţii, care se foloseşte apoi, în mod dinamic, pentru controlul sarcinilor de calcul distribuite în reţea.

B. Modelul de distribuire a proceselor paralele

Pentru generarea unei imagini pentru un punct de observare dat se execută o operaţie de traversare a grafului bazei de date pentru extragerea obiectelor vizibile din acel punct de observare. Operaţia de traversare a grafului bazei de date, care este o operaţie de căutare în adâncime în graf (depth-first-search) a tuturor obiectelor tridimensionale cuprinse, total sau parţial, în regiunea de vizibilitate a observatorului, creează, în mod dinamic, un arbore de traversare. În operaţia de traversare a bazei de date se retează orice nod, împreună cu toţi succesorii lui, dacă obiectul reprezentat de acel nod este invizibil (se află în afara regiunii de vizibilitate a observatorului). Din această cauză, arborele de traversare, generat în mod dinamic, pentru fiecare imagine sintetizată, depinde de punctul de observare considerat şi este necesară partiţionarea dinamică a acestuia.

Algoritmul de traversare a grafului bazei de date TDB începe prin expandarea nodului iniţial al grafului şi generarea succesorilor lui, care sunt memoraţi într-un spaţiu al stărilor de traversare. Spaţiul stărilor se memorează sub forma unei stive a cărei dimensiune creşte linear cu adâncimea arborelui de traversare. În fiecare pas succesiv, TDB expandează cel mai recent nod generat. Dacă acest nod nu are succesori, atunci TDB se întoarce şi expandează un nod diferit.

Stările neexplorate (nodurile grafului bazei de date) se memorează în stiva de stare a traversării împreună cu calea predecesorilor acestuia, pentru obţinerea informaţiilor de localizare a fiecărui nod, şi cu valoarea timpului de execuţie estimat, calculat ca atribut al nodului, necesar pentru controlul dinamic al granularităţii programului.

Fiecare procesor execută sarcini de calcul constând în operaţiile de generare de imagine pentru subarbori componenţi ai arborelui de traversare a bazei de date şi, pentru încărcarea echilibrată a procesoarelor, distribuirea acestor sarcini de calcul se face în mod dinamic, în cursul execuţiei sintezei de imagine.

Operaţia de traversare a bazei de date poate fi amplasată ca un cluster de procese într-un singur procesor, ceea ce echivalează cu o execuţie centralizată a traversării, sau poate fi descompusă în mai multe procese componente şi aceste procese să fie distribuite în toate procesoarele reţelei, ceea ce reprezintă partiţionarea spaţiului de traversare a bazei de date grafice, deci a stivei de stare a traversării.

Pentru încărcarea echilibrată a procesoarelor se aplică o metodă de încărcare dinamică, iniţiată de procesoarele inactive, care vor fi receptoare ale sarcinilor de calcul.

Fiecare procesor se poate afla, la un moment dat, într-una din următoarele două stări:

- *inactiv*, când nu are de executat nici o sarcină de calcul (cluster de procese);
- *activ*, când se află în cursul executării unui proces.

Pentru încărcarea echilibrată a procesoarelor se aplică o metodă de încărcare dinamică iniţiată de procesoarele inactive, care vor fi receptoare ale sarcinilor de calcul.

Programul începe prin traversarea bazei de date de către un procesor, numit procesor rădăcină, P_0, toate celelalte procesoare fiind inactive.

În traversarea centralizată a bazei de date, procesorul rădăcină P_0 execută algoritmul de traversare TDB. Orice procesor inactiv transmite un mesaj către procesorul P_0, care conține eticheta proprie și prin care anunță că poate accepta o sarcină de calcul. Un procesor receptor primește, ca răspuns la cererea sa de sarcină de calcul, un mesaj de la procesorul P_0, care conține identificatorul obiectului pe care trebuie să-l genereze. După executarea operațiilor de generare a imaginii obiectului, procesorul respectiv devine din nou inactiv și adresează o nouă cerere de sarcină de calcul către procesorul P_0.

Traversarea distribuită a bazei de date începe prin asignarea întregului spațiu de traversare a bazei de date unui singur procesor, P_0, în timp ce toate celelalte procesoare au un spațiu de traversare nul, deci sunt inactive.

Un procesor inactiv selectează un donator căruia îi transmite o cerere. Dacă procesorul inactiv primește ca răspuns o sarcină de calcul (o parte din spațiul stărilor de traversat) de la procesorul donator, el devine activ. Dacă primește un mesaj de *rejecție* (deoarece procesorul donator selectat nu are sarcină de calcul de distribuit), el selectează un alt donator către care trimite o cerere de sarcină de calcul. Acest proces continuă până când este epuizat întregul spațiu de traversare a bazei de date.

Granularitatea programului de sinteză de imagine este timpul mediu de execuție a sarcinilor de calcul distribuite procesoarelor. O sarcină de calcul este un cluster de procese de generare de imagine care se execută pe un procesor: sincronizarea execuției acesteia cu alte procese se face prin transfer de mesaje. Granularitatea sarcinilor de calcul se poate controla dinamic, în operația de distribuire a acestora, în funcție de modul de traversare a bazei de date, centralizat sau distribuit.

3. Mascarea timpului de comunicație prin comutarea între contexte multiple

Pentru distribuirea sarcinilor de calcul, procesoarele comunică între ele prin transfer de mesaje: fiecare procesor inactiv trimite un mesaj prin care solicită de la un alt procesor din rețea o sarcină de calcul și primește ca răspuns un mesaj care conține o partiție a spațiului de traversare a bazei de date, neexplorată încă.

Numărul de sarcini de calcul s, care se distribuie, este limitat inferior de numărul de procesoare p ale rețelei. Pentru $s < p$, vor exista (p-s) procesoare care nu primesc nici o sarcină de calcul (rămân inactive), și eficiența sistemului scade. La valori mari ale granularității programului, nu se pot obține performanțe scalabile ale sistemului și este necesar ca valoarea lui s să fie limitată inferior de numărul de procesoare, deci $s = \Omega(p)$.

Fie $s > p$ numărul de sarcini de calcul și fiecare sarcină urmează să fie atribuită, în mod dinamic, unui procesor. Pentru aceasta, un procesor inactiv lansează un mesaj de cerere de sarcină de calcul unui alt procesor din rețea și așteaptă de la acesta un mesaj de răspuns. Timpul de comunicație a acestor mesaje (timpul total dintre momentul de start al mesajului de cerere de sarcină de calcul și până în momentul în care procesorul receptor a primit mesajul de răspuns) are ca efect menținerea procesorului care a emis cererea în stare inactivă și deci acest timp reprezintă o componentă a costului suplimentar de execuție paralelă.

Considerând aceste mesaje de distribuire a sarcinilor de calcul de lungime constantă, cu un timp mediu de comunicație L, costul total suplimentar de distribuire este: $T_O = L * s$, unde L este o funcție care depinde de topologia și de dimensiunea rețelei.

Rezultă $T_O = \Omega(p)$ și deci la creșterea numărului de procesoare crește costul suplimentar și scade eficiența sistemului.

Costul suplimentar de distribuire a sarcinilor de calcul poate fi redus substanțial, dacă se folosește un mecanism de *mascare a timpului de comunicație* prin utilizarea de contexte multiple, care permite procesoarelor să comute de la un context la altul atunci când au loc operații de comunicație.

Pentru aceasta, distribuirea sarcinilor de calcul se face nu între cele p procesoare fizice ale rețelei, ci între un număr $v = c*p$ de procesoare virtuale, unde $c \geq 1$.

În fiecare procesor fizic, se amplasează c procesoare virtuale, fiecare procesor virtual definind un context de execuție. Atunci când un procesor virtual devine inactiv și începe o comunicație de obținere a unei sarcini de calcul, mecanismul de comutare a contextelor îl suspendă și activează alte contexte de execuție, astfel încât procesorul nu mai rămâne inactiv pe durata comunicației (figura 1).

Fie timpul mediu de execuție al procesorului pentru o sarcină de calcul R, care, conform definiției anterioare, reprezintă granularitatea programului de sinteză de imagine. Timpul de comunicație pentru distribuirea sarcinilor de calcul poate fi mascat prin comutarea între contexte, dar această comutare necesită la rândul ei un timp, C, care reprezintă un cost suplimentar.

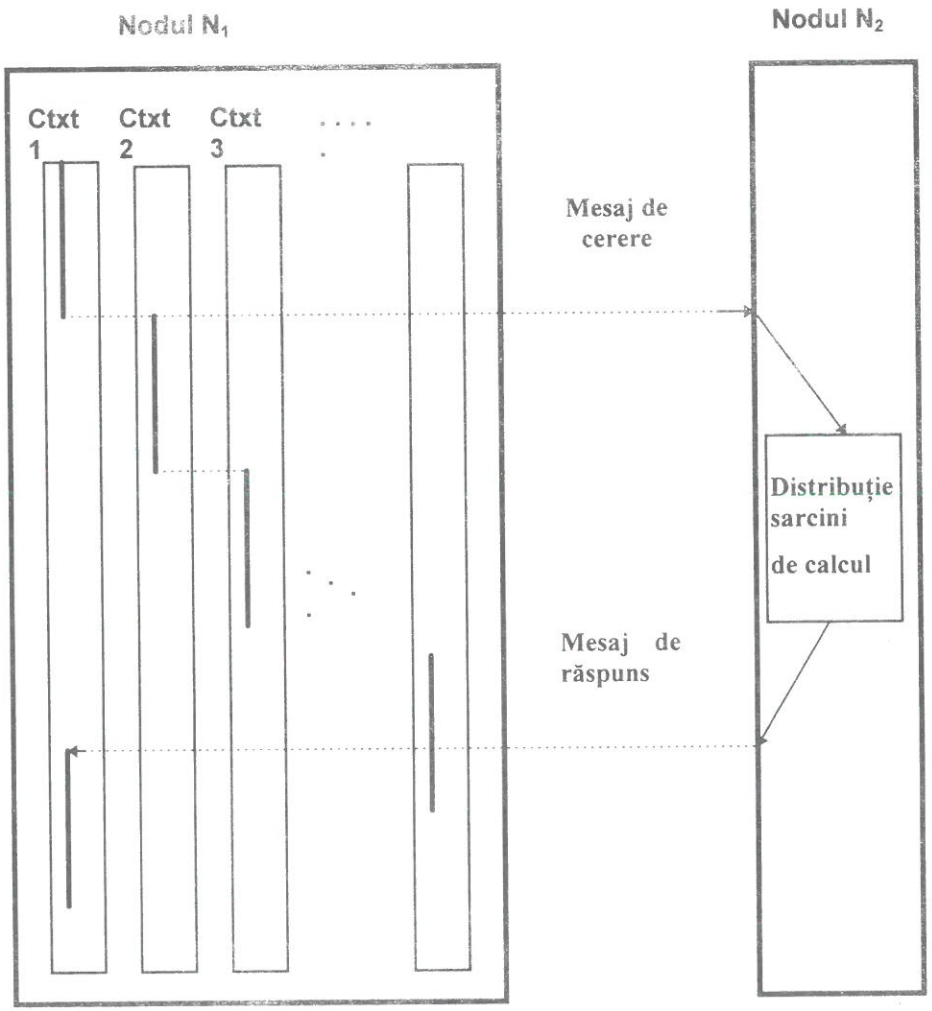

Figura 1. Contexte multiple de execuˇie într-un procesor

Dacă în fiecare procesor se asigură un număr de contexte suficient de mare, astfel încât să existe întotdeauna un context gata de execuție atunci când are loc o comutare, atunci procesorul nu va fi niciodată inactiv.

Eficiența de utilizare a procesorului, analizată în funcție de numărul de contexte c definite, evidențiază două regiuni de lucru, o *regiune saturatĺ* și o *regiune linearĺ* așa cum se vede în figura 2.

• **În regiunea saturată**, procesorul operează cu un coeficient de utilizare maximă. Durata de execuție a unei sarcini de calcul este dată de R+C, de unde rezultă eficiența sistemului:

$$E_{sat} = R / (R+C) = 1 / (1 + C/R) \qquad (1)$$

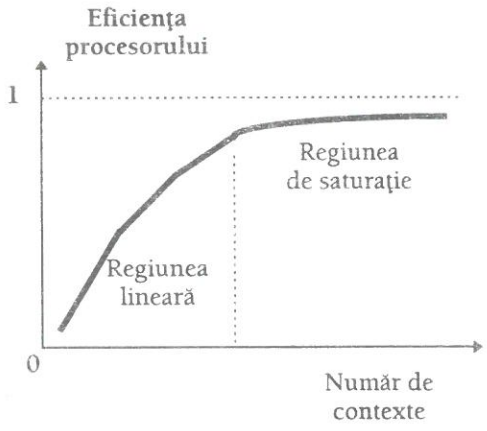

Figura 2. Eficiența de utilizare a procesorului

Eficiența de utilizare a procesoarelor în regiunea saturată este independentă de timpul de comunicație L și de numărul de contexte c, dacă banda de comunicație pe link-urile procesoarelor este nelimitată. Pentru valori limitate ale lărgimii de bandă, o dată cu creșterea numărului de contexte c, crește numărul de procesoare virtuale, care comunică între ele, deci crește volumul de date transferate pe link-uri. Acest lucru poate provoca congestia mesajelor pe link-uri, ceea ce duce la creșterea timpului de comunicație a mesajelor și, în final, la intrarea procesoarelor în stare de inactivitate. În regiunea de saturație, limitarea numărului de contexte care pot fi executate pe un procesor este datorată benzii de comunicație limitată pe link-uri.

Punctul de saturație se obține pentru un număr de contexte c_s pentru care timpul consumat de procesor pentru servirea a (c-1) contexte este egal cu timpul de comunicație, deci: (R+C)*(c-1) = L. Rezultă:

$$c_s = 1 + L/(R+C). \quad (2)$$

Pentru valori $c \geq c_s$ procesorul se menține în regiunea saturată, iar pentru $c < c_s$ procesorul intră în regiunea liniară.

• **În regiunea liniară**, este posibil să nu existe nici un context gata, atunci când unul dintre ele este comutat în starea de așteptare a unei sarcini de calcul și astfel procesorul poate deveni inactiv. Timpul necesar pentru comutarea la un alt context, executarea acestuia și așteptarea până primește răspuns este egal cu R+C+L. Presupunând că c este sub punctul de saturație, atunci, în acest timp toate celelalte contexte sunt executate o dată în procesor, deci eficiența procesorului este:

$$E_{lin} = cR/(R+C+L). \quad (3)$$

Se poate observa că eficiența crește linear cu numărul de contexte până se atinge punctul de saturație, după care rămâne constantă. Expresia pentru E_{sat} dă limita eficienței de utilizare a procesorului și ea arată importanța factorului C/R, adică faptul că durata de execuție a unui context trebuie să fie corelată cu durata de comutare a contextului.

Pentru un sistem paralel dat, valorile timpului de comunicație a mesajelor (L) și valorile timpului de comutare a contextului (C) sunt cunoscute și, ca urmare, se pot deduce valorile optime pentru numărul de contexte care să asigure funcționarea în regiunea de saturație a procesoarelor precum și pentru granularitatea programului (R).

Considerând că procesoarele lucrează în regiunea de saturație, condițiile impuse pentru granularitatea optimă a programului sunt:

(a) Pentru ca toate procesoarele virtuale, definind contextele de execuție în procesoarele fizice, să poată primi sarcini de calcul, este necesar să se distribuie un număr de sarcini de calcul s cel puțin egal cu numărul de procesoare virtuale, deci:

$$s \geq c*p \quad (4)$$

Granularitatea R se poate exprima ca raportul dintre timpul de execuție total, necesar generării imaginilor tuturor obiectelor vizibile, (T_S) și numărul de sarcini de calcul s, deci:

$$R = T_S / s \quad (5)$$

Pentru a menține procesoarele în regiunea saturată, mai este necesar ca numărul de contexte c să fie mai mare decât limita punctului de saturație, deci:

$$c > 1 + L/(R+C) \quad (6)$$

Din expresiile (4), (5), (6), rezultă condiția: $R < K(p)$, care limitează superior granularitatea la o valoare K(p), care depinde de caracteristicile de comunicație ale rețelei (topologie, banda de comunicație pe link-uri), de timpul de comutare a contextelor (care este o caracteristică a procesoarelor componente), precum și de dimensiunea p a rețelei.

(b) Condiția $R < K(p)$ impune adoptarea unei strategii de distribuire a unor sarcini de calcul cu granularitare (R) cât mai mică, ceea ce asigură funcționarea în regiunea de saturație a procesoarelor. În schimb, din ecuația (1), rezultă că pentru o utilizare eficientă a procesoarelor, este necesar ca $R >> C$, ceea ce limitează inferior valoarea granularității.

Valoarea granularității optime, R_{opt}, se poate deduce experimental pentru o rețea dată. În plus, pentru un același tip de rețea, ca tip de procesoare, topologie și bandă de comunicație pe link-uri, valoarea R_{opt}, depinde de numărul de procesoare p al rețelei.

4. Determinarea experimentală a granularității optime

S-a studiat experimental influența granularității programului asupra performanțelor sistemului paralel de sinteză de imagine într-o rețea de transputere, conectate în topologie grilă bidimensională, variind:

p - numărul de procesoare din rețea;

s - numărul de sarcini de calcul distribuite pentru generarea unei aceleiași imagini (T_S constant)

S-a măsurat *timpul de execuție paralelă* pentru un segment de bază de date grafică standard

(combinație de obiecte tridimensionale și reprezentări ale terenului), folosind funcțiile de "timer" ale limbajului Occam.

Măsurătorile efectuate pentru p = 4, p = 8, p = 16 procesoare indică un optim (minim) al timpului de execuție paralelă T_P pentru o valoare a lui s:

$$3\,p < s < 6\,p$$

Dacă se analizează rezultatele obținute, se observă că pentru s < 3 p, are loc o creștere accentuată a timpului de execuție paralelă. Acest lucru se explică prin aceea că, la o granularitate mare a programului, numărul de contexte care se pot crea în fiecare procesor nu asigură funcționarea acestora în regiunea de saturație, și timpul de execuție paralelă crește prin intrarea procesoarelor în stare inactivă în așteptarea terminării unicației mesajelor de distribuți a sarcinilor de calcul.

În extrem, pentru s = 1, avem de fapt o execuție secvențială pe un singur procesor din rețea, celelalte fiind neutilizate și creșterea puternică a timpului de execuție reflectă acest lucru. Pentru valori s > 6 p, timpul de execuție paralelă crește din nou, datorită congestiei dată de banda de comunicație limitată pe link-urile rețelei.

Pentru valori intermediare (3 p < s < 6 p) se atinge punctul de saturație, pentru care eficiența sistemului este optimă (timpul de execuție paralelă minim).

Definirea granularității programului prin numărul de sarcini de calcul (s) este o definire indirectă a acesteia. Sarcina totală de calcul T_S este divizată în $s = T_S / R$ sarcini de calcul, fiecare de durată medie de execuție (granularitate) R, care se distribuie în rețea. Rezultă $R = T_S / s$.

5. Controlul dinamic al granularității proceselor

Se presupunem că, pentru un sistem de sinteză de imagine dat, s-a determinat experimental valoarea granularității optime $R_{opt} = T_S / s_{opt}$. Această valoare este utilizată de programul de distribuție a sarcinilor de calcul pentru determinarea granularității acestora.

Așa cum s-a precizat în Secțiunea 2, fiecare nod n_i din graful bazei de date, atins în operația de

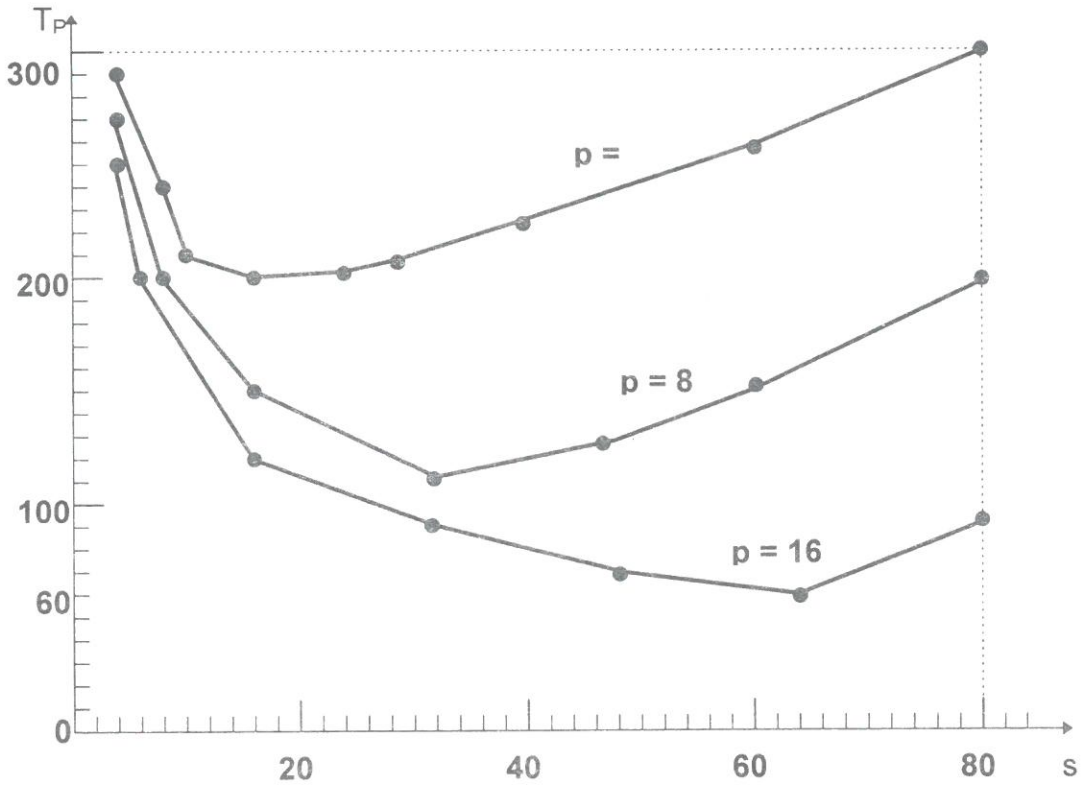

Figura 3. Dependența timpului de execuție paralelă (T_P) de numărul de sarcini de calcul distribuite (s)

traversare, are ca atribut memorat timpul estimat de calcul, (t_i) care este timpul total, necesar pentru generarea imaginii nodului respectiv şi a tuturor succesorilor lui.

În traversarea centralizată a bazei de date, procesorul P_0, în care se execută algoritmul TDB, răspunde unui procesor care solicită o sarcină de calcul cu identificatorul unui nod din baza de date grafică. Un nod n_i din graful bazei de date, care va fi rădăcina unui subarbore de traversare, este considerat ca o sarcină de calcul, care poate fi distribuită dacă timpul de execuţie estimat (t_i) , este mai mic sau egal decât valoarea granularităţii optime:

$$t_i \leq R_{opt} \qquad (7)$$

Dacă $t_i > R_{opt}$, atunci se continuă expandarea nodurilor, până se ajunge la noduri care îndeplinesc condiţia (7).

În traversarea distribuită a bazei de date, controlul dinamic al granularităţii proceselor se bazează pe controlul pragului de divizare a stivei stărilor de traversare. Un procesor donator transmite ca sarcină de calcul unui procesor inactiv, o parte a stivei sale de stare a traversării, prin divizarea acesteia. Pragul de divizare al stivei este dat de valoarea granularităţii optime, R_{opt} .

Dacă suma timpilor de execuţie estimaţi ai tuturor nodurilor înscrise în stivă este mai mare decât R_{opt} , deci dacă:

$$\Sigma t_i \geq R_{opt} \qquad (8)$$

atunci se împarte stiva în două părţi: o parte este reţinută de procesorul donator, iar cealaltă parte se transmite, ca sarcină de calcul, procesorului receptor.

O stivă de stare a traversării devine indivizibilă dacă suma timpilor de execuţie ai tuturor nodurilor pe care le conţine este mai mică decât granularitarea medie a proceselor, deci dacă: $\Sigma t_i < R_{opt}$.

6. Concluzii

Obiectivul acestui studiu a fost de a investiga posibilităţile de execuţie eficientă a proceselor paralele de sinteză de imagine într-un calculator paralel cu memorie distribuită şi transfer de mesaje. Distribuirea dinamică a sarcinilor de calcul, deci planificarea proceselor în procesoarele componente, are ca scop minimizarea costului suplimentar de execuţie paralelă prin încărcarea echilibrată a procesoarelor şi prin scăderea costului comunicaţiilor de distribuire a sarcinilor de calcul.

Acest deziderat s-a obţinut prin adaptarea granularităţii programului la caracteristicile constructive ale reţelei de procesoare (capacitatea de calcul a procesoarelor, capacitatea de comunicaţie pe link-uri, numărul de procesoare).

Controlul dinamic al granularităţii programului s-a realizat prin controlul nivelului de expandare a nodurilor grafului bazei de date grafice, pentru traversarea centralizată a bazei de date şi prin controlul pragului de divizare al stivei de stare a traversării, pentru traversarea distribuită a bazei de date.

Această tehnică a fost utilizată în realizarea mai multor sisteme de sinteză de imagine în timp real, incluse în simulatoare de zbor.

Bibliografie

1. **SADAPPAYAN, P., ERCAL, F., RAMANUJAM, J.**: Cluster partitioning approaches to mapping parallel programs onto a hypercube. În: Parallel Computing, No.13, 1990.
2. **IONESCU, F.**: Structuri paralele pentru sinteza de imagine în timp real. Teză de doctorat, Bucureşti, 1996.
3. **IONESCU, F.**: Baze de date grafice orientate pe obiecte. În: Revista Română de Informatică şi Automatică, vol.6, nr.1, 1996.
4. **GERASOULIS, A., YANG, T.**: On the granularity and clustering of directed acyclic task graphs. În: IEEE Transactions on Parallel and Distributed Systems, Vol.4, No. 6, June 1993.
5. **WILLEEBEK-LeMAIR, M. A., REEVERS, A.P.**: Strategies for dynamic load balancing on highly parallel computers. În: IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 9, September 1993.