Articole

ANALIZA UNOR ALGORITMI DE ACCES PENTRU BAZE DE DATE ÎN REŢELE RAPIDE

dr.ing. Ştefan Stăncescu
ing. Traian Ştefănescu

Universitatea Politehnica Bucureşti

Rezumat: Este abordată problema performanţelor de timp de acces în masive de baze de date distribuite şi stabilirea de criterii de proiectare a acestora. Pornind de la schemele cu pompă de date Datacycle şi schema 2PL (Two Phases with Lock), se analizează variante de scheme cu trimitere la cerere (send on demand). Se examinează, prin simulare, performanţele schemelor prezentate, cu decelarea diferenţelor în performanţele de access la date în diferite variante de solicitare a bazei de date: capacitatea reţelei, volumul şi tipul solicitărilor. Se stabilesc recomandari la alegerea protocolului de acces pentru aplicaţii cu masive de baze de date distribuite.

Cuvinte cheie: sisteme de baze de date distribuite, reţele de calculatoare, algoritmi de acces.

1. Introducere

Într-un sistem de baze de date distribuite (SBDD), informaţiile sunt împărţite în baze de date mai mici, distribuite fiecare pe calculatoare în reţea. În SBDD, se îndeplinesc funcţii de procesare a interogărilor şi de control al concurenţei. O dată cu dezvoltarea reţelelor de mare viteză (Gb/s), costurile de transmisie a datelor în reţea au scăzut şi a devenit necesară dezvoltarea unor noi algoritmi care să utilizeze eficient lărgimea de bandă (mai mare decât în cazul reţelelor obişnuite = Mb/s) disponibilă, adică să obţină la ieşirea sistemului o productivitate mai mare în rezolvarea tranzacţiilor.

2. Schemele Datacycle, 2-PL şi "trimite la cerere"

În schema Datacycle, [1,2], o pompă de date livrează periodic, în reţea, întregul conţinut al bazei de date. Volumul bazei de date de G Gb, împărţită în D elemente de date, de dimensiuni egale, se transmite de la un calculator central pe o magistrală de lungime 2L km. Capacitatea reţelei este C Gb/s. Cu aceşti parametri se calculează timpul necesar pentru transmiterea întregii baze de date pe bus-ul de transmisie T_{dc}, timpul datacycle. Fiecare element de date ocupă magistrala un slot de timp (T_s):

$$T_{dc} = \frac{G}{C} \text{ şi } T_s = \frac{T_{dc}}{D} \qquad (2.1)$$

În aceste circumstanţe, T_s şi T_{dc} reprezintă timpul minim respectiv maxim, pentru a obţine un element de date. Presupunând că o tranzacţie poate sosi oricând, în timpul unui datacycle, distribuţia timpului de acces la un singur element de date se poate considera uniformă, în intervalul (T_s, $T_{dc}+T_s$). Fie Φ variabila aleatoare, care indică timpul necesar pentru a obţine un element de date. Atunci, funcţia densitate de probabilitate a lui Φ va fi:

$$f_\Phi(t) = \begin{cases} \frac{1}{T_{dc}}, pentru\ T_s < t \le T_{dc} + T_s \\ 0, altfel \end{cases} \qquad (2.2)$$

Pentru schema 2-PL,[3], s-a ales topologia în inel cu fibră optică bidirecţională, cu cele N calculatoare dispuse la egală distanţă în inelul cu circumferinţa 2L. Fiecare element de date D este localizat într-unul din cele N calculatoare. Fiecare calculator are acelaşi număr de elemente de date. Timpii de aşteptare în cozi sunt neglijaţi. Întârzierea de propagare între două calculatoare adiacente se notează T_p şi timpul pentru a transmite un singur element de date este T_{tr},(c=300000 km/s). Se neglijează interblocarile.

$$T_p = 2 * \frac{L}{N * c} \qquad (2.3)$$

$$T_{tr} = \frac{G}{D * C} \qquad (2.4)$$

Timpul minim pentru a accesa un element de date de către un calculator este zero (cazul în care acel element de date se află chiar pe acel calculator), iar timpul maxim este $N*T_p+T_{tr}$. În acest caz, funcţia densitate de probabilitate a lui Φ este:

$$f_\Phi(t) = \begin{cases} \frac{1}{N * T_p + T_{tr}}, pentru\ \ 0 < t \le N * T_p + T_{tr} \\ 0, altfel \end{cases} \qquad (2.5)$$

Fie T_A si T_B timpul maxim, necesar pentru a accesa un element de date, în schema Datacycle, respectiv 2-PL. Egalând T_A cu T_B şi urmărind formulele din subcapitolele anterioare:

$$T_A = T_{dc}\left(= \frac{G}{x}\right) \qquad (2.6)$$

$$T_B = N * T_p + T_{tr}\left(= N * T_p + \frac{G}{D * x}\right) \qquad (2.7)$$

se obţine x, rata reţelei la care are loc intersecţia între performanţele celor două scheme. Aceasta sugerează că fiecare schemă este favorabilă într-o anumită gamă de variaţie a ratei reţelei. Într-adevăr, pentru schema Datacycle, managerul de actualizare central reduce performanţele sistemului, iar algoritmul multiversiune este prea optimist, şi nu este eficient în condiţiile concurenţei la elementele

de date pe reţele ultrarapide. Pentru schema 2-PL, există posibilitatea interblocărilor şi apare o supraîncărcare a reţelei, asociată cu secvenţa de mesaje: cerere de blocare a elementului de date, permisiune de blocare a elementului de date, eliberarea elementului de date, valabilă în timpul accesării oricărui element de date.

În protocolul "trimite la cerere", elementele de date sunt, dependente de performanţele calculatorului pe care rezidă, deoarece toate solicitările pentru a accesa un element de date trebuie să fie procesate de calculatorul pe care se află acel element de date. Aceasta conduce la o secvenţă de mesaje schimbate între calculatorul care iniţiază tranzacţia şi calculatorul pe care se află elementul de date. În efortul de a reduce numărul de secvenţe de mesaje şi de a promova o execuţie cooperativă între toate calculatoarele ce constituie BDD, se foloseşte o schemă de control concurent la elementele de date. În această nouă schemă, elementele de date nu mai sunt "limitate" la un anumit calculator, în mod special. Fiecare calculator ce face parte din BDD, întreţine o coadă de pretenţii, pentru fiecare element de date, care este localizat în mod curent pe acel calculator. Coada de pretenţii pentru un element de date conţine o listă a tranzacţiilor (cu identificatoarele calculatoarelor unde este executată tranzacţia) care cer acel element de date şi, de asemenea, specifică acţiunea inclusă în tranzacţie asupra acelui element de date (citire/scriere). Setul de operaţii de acces, împreună cu marca timpului de sosire a tranzacţiei T - TS(T) a fiecărei noi tranzacţii este transmis către toate calculatoarele din BDD. Figura 2.1 prezintă un mesaj tipic, trimis de un calculator care procesează o tranzacţie cu k elemente de date în setul său de operaţii de acces:

Figura 2.1

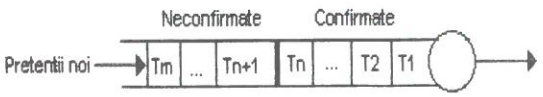

Figura 2.2

Durata confirmării unei tranzacţii este un timp mai mare sau egal cu cel necesar trimiterii informaţiei setului de operaţii acces spre toate calculatoarele din BDD. Acest lucru asigură ca, la fiecare calculator, coada de pretenţii (figura 2.2) să aibă intrări în ordine (în timp), şi, în acest fel, nu pot apărea interblocări.

La pornirea sistemului, toate elementele de date se află pe calculatoare oarecare. Când tranzacţiile încep să sosească, cozile de pretenţii încep să se ocupe. Toate calculatoarele îşi transmit elementele de date rezidente spre sistemul care a iniţiat prima tranzacţie confirmată, completând cozile de pretenţii respective. În timp ce coada de pretenţii este transmisă, alte tranzacţii necesitând acelaşi element de date, pot sosi în sistem. Informaţii despre aceste tranzacţii pot fi obţinute din informaţia de transmisie (vezi formatul unui mesaj de transmisie). Fiecare calculator care iniţiază o tranzacţie aşteaptă pentru ca întregul set de operaţii de acces să ajungă la locaţia lui, termină procesarea tranzacţiei şi apoi transmite elementele de date la alte calculatoare care se afla la rând în cozile de pretenţii respective. Acest mecanism elimină necesitatea de deblocare a elementelor de date, faţă de sistemele tradiţionale, în care, pentru a se menţine atomicitatea tranzacţiilor, se implementează protocoale speciale de validare (commit). Noul algoritm are nevoie doar de un protocol de validare local. Atât timp cât fiecare calculator întreţine cozi de pretenţii doar pentru acele elemente de date, care rezidă, în mod curent pe acel calculator, necesităţile de memorie nu sunt foarte mari. Fiecare calculator trebuie să aibă alocată memoria necesară stocării informaţiei despre setul de operaţii de acces al tranzacţiilor confirmate.

O tranzacţie tipică de actualizare are un set de operaţii de citire precum şi un set de operaţii de scriere. Astfel, o coadă de pretenţii tipică pentru un element de date are intrări de citire şi intrări de scriere. Intrările consecutive de citire pot fi procesate în paralel, în timp ce intrările de scriere vor fi servite serial. De exemplu, considerând coada de pretenţii a unui singur element de date D_j, unde se află două intrări de scriere, separate de câteva intrări de citire. după ce prima scriere a fost completată de calculatorul care deţine (acum) copia scrisă a lui D_j, se trimite mai departe versiunea actualizată a lui D_j către toate calculatoarele care au o intrare de citire în coada de pretenţii, înainte de scrierea următoare. Copia scrisă actuală este trimisă calculatorului care este responsabil pentru ultima intrare de citire înainte de a doua intrare de scriere. După ce este terminată fiecare tranzacţie care necesită citirea lui D_j, există un mecanism de informare a calculatorului, care vrea să-l modifice pe D_j. Acest mecanism este realizat prin calculatoarele care-l citesc pe D_j şi care trimit câte un mesaj către calculatorul care aşteaptă să-l scrie pe D_j; acesta aşteaptă momentul propice actualizării şi anume acela când s-a sfârşit citirea lui D_j. Calculatorul care are copia de scriere transmite această copie a lui D_j împreună cu mesajul şi apoi actualizarea la acel calculator poate începe.

Una din presupunerile importante, făcute la dezvoltarea acestui protocol, este şi aceea că perioada de timp a confirmării tranzacţiei este setată la o valoare aleasă, astfel încât este atinsă ordonarea totală a mesajelor (fiecare calculator

vede aceeaşi ordonare a tranzacţiilor confirmate într-o coadă de pretenţii). Într-un sistem real, mesajele pot fi întârziate şi mecanismul de transmisie poate fi destul de sofisticat pentru a garanta ordonarea totală chiar în aceste circumstanţe.

3. Protocol integrat de control al concurenţei

Algoritmul de control integrat al concurenţei este aplicabil unui SBDD general, unde sunt create şi procesate ambele cozi de citire (tranzacţii read-only) şi de actualizare (tranzacţii read/write). Algoritmul propus este hibrid între schema Datacycle şi schema "trimite la cerere". Operaţiile de citire vor fi tratate de mecanismul Datacycle, iar operaţiile de scriere sunt tratate de protocolul "trimite la cerere". Operaţiile de citire şi scriere trebuie integrate astfel încât să nu fie violat criteriul de consistenţă a datelor (de exemplu, setul de operaţii de citire a unei tranzacţii nu poate fi scris peste altă tranzacţie până când prima tranzacţie nu s-a terminat). Pentru operaţia de citire se alege versiunea distribuită a schemei Datacycle. Întreaga bază de date este divizată în fragmente şi câtorva calculatoare din BDD le revine responsabilitatea să transmită un fragment periodic către toate celelalte. Mărimile fragmentelor nu sunt egale, acestea depinzând de alţi factori. Fiecare element de date, din fiecare fragment, au ataşate informaţii despre marca de timp. Valoarea mărcii de timp este timpul la care acel element de date a fost actualizat de o tranzacţie şi, de asemenea, conţine identitatea tranzacţiei care a fost responsabilă pentru actualizare. Această valoare a mărcii de timp este iniţializată la zero la pornirea sistemului. Ca şi în arhitectura Datacycle, calculatoarele vor fi echipate cu filtre de date şi alt hardware necesar pentru a citi elementele de date "din zbor". Elementele de date pot fi citite de către calculatoare în fiecare ciclu de transmisie şi pot fi folosite în procesarea interogărilor. Atât timp cât toate elementele de date din setul de operaţii de citire a interogării sunt de aceeaşi "versiune", interogarea este garantată a da rezultatul corect. În contextul arhitecturii Datacycle, o cerere care poate să obţină întregul său set de operaţii de citire într-un singur ciclu, este garantată a fi corectă, atât timp cât actualizările sunt incluse în şirul de transmisie doar la începutul ciclului. Informaţia poate să nu fie cea mai recentă, dar rezultatul interogării va fi consistent.

Toate calculatoarele actualizează cozile de pretenţii, iar dacă sunt responsabile de transmiterea unui fragment din baza de date, adaugă informaţia mărcii de timp la elementul de date şi apoi şterge din cozile de pretenţii informaţia despre tranzacţia actualizată prin verificarea mărcii de timp.

Calculatoarele care procesează actualizări transmit întregul set de operaţii de acces (setul de operaţii de citire şi setul de operaţii de scriere) al actualizării către toate calculatoarele şi aşteaptă evenimentele (a) şi (b). Evenimentul (a) apare, dacă sunt primite toate elementele de date din setul de operaţii de acces, iar evenimentul (b), dacă mesajul "completat" a ajuns de la tranzacţiile care deţin o copie de citire a elementelor de date, în setul lor de operaţii de citire.În acest din urmă caz, se execută actualizarea, se trimite o copie a fiecărei actualizări (o dată cu timpul la care s-a executat actualizarea) către calculatoarele respective, responsabile pentru transmiterea lor. Dacă se deţine o copie de citire a elementului de date D_j, se trimite o completare a mesajului către următoarea scriere în coada de pretenţii pentru D_j, iar dacă se deţine o copie de scriere a lui D_j, se trimite o copie de citire a lui D_j către toate calculatoarele care au o intrare de citire în coada de pretenţii, înainte de scrierea următoare şi se trimite copia de scriere către următoarea scriere.

Calculatoarele care procesează interogări scanează canalele de transmisie, pe care elementele de date sunt transmise, urmărind elementele de date din setul de operaţii de acces al interogării, citesc doar acele còpii care sunt consistente mutual prin verificarea informaţiei mărcii de timp respective şi execută interogarea.

Pentru a analiza acest model, folosim diagrama din figura 3.3 care prezintă într-o ordine cronologică evenimentele care au loc într-o tranzacţie tipică de actualizare.

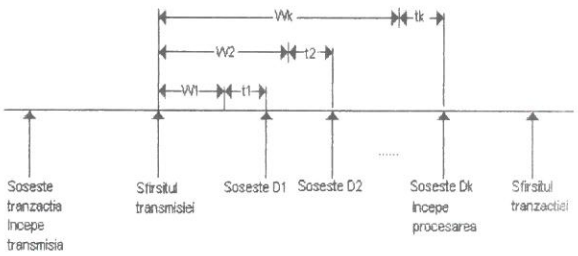

Figura 3.3

Când o tranzacţie ajunge în poziţia de început, ea intra într-o fază de transmisie de durata B, în timpul căreia setul de operaţii de acces al tranzacţiei este transmis către toate calculatoarele din bază de date. Durata confirmării este stabilită la valoarea B. În analiza ce urmează, se presupune că durata de transmisie se cunoaşte cu exactitate şi, în acest fel, durata confirmării este setată la această valoare. Evident, această situaţie nu este una reală, în realitate mesajele pot fi întârziate. După ce tranzacţia este confirmată, calculatorul ce procesează tranzacţia de actualizare trebuie să aştepte ca fiecare dintre obiectele de date k să ajungă la acel calculator, înainte ca procesarea lor să înceapă. Durata executării actualizării este notată cu E şi este o

variabilă aleatoare. După ce transmisia s-a terminat, cererile pentru fiecare element de date din setul de operaţii de acces al tranzacţiei intră în cozile de pretenţii respective, ale fiecărui obiect de date. Acum, fiecare interogare trebuie să ajungă prima în respectiva coadă de pretenţii, înainte ca elementul de date să poată fi trimis către calculatorul aflat în discuţie. Timpul de servire şi timpul de aşteptare pentru coada de pretenţii ale elementului de date D_j este notat prin variabila aleatoare X_j si, respectiv, W_j. Variabila aleatoare t_j, a cărei distribuţie este cunoscută, indică timpul (transmisie + propagare) necesar elementului de date D_j pentru a ajunge la calculatorul aflat în discuţie. Timpul de servire pentru o tranzacţie ce accesează k elemente de date la oricare calculator se notează S^k. Din diagrama din figura 3.3 se pot deduce următoarele relaţii:

$$S^k=B+\max\{W_1+t_1, W_2+t_2,\ldots,W_k+t_k\}+E \qquad (3.1)$$

$$X_I=S^k - W_I - B, \qquad I=1,..,k \qquad (3.2)$$

Atât timp cât W_I este funcţie de X_I, din ecuaţia (3.1) rezultă că S^k este funcţie de X_I, şi, din ecuaţia (3.2), X_I este funcţie de S^k. Pentru rezolvarea acestei probleme, se foloseşte o metodă iterativă [3]. Pentru a simplifica notaţiile, ce este notat cu S^k vom nota cu S, presupunând că toate actualizările accesează acelaşi număr (K) de elemente de date. Timpul mediu de răspuns al oricărei actualizări este definit ca fiind timpul total, cheltuit de actualizare în SBDD. Pentru aceasta, avem nevoie să găsim timpul mediu de aşteptare ($\overline{W}$) al oricărei tranzacţii, care depinde de $\overline{S}$ si $\overline{S^2}$, primul şi, respectiv, cel de-al doilea moment al variabilei S. Procesul de sosire al actualizărilor în sistem este presupus Poisson cu frecvenţa de sosire λ, iar nivelul de multiprogramare al fiecărei tranzacţii la fiecare calculator este presupus unitar:

$$\overline{W}=\frac{\lambda*\overline{S^2}}{2*(1-\lambda*\overline{S})} \qquad (3.3)$$

Timpul mediu de răspuns $\overline{R}$ este dat de:

$$\overline{R}=\overline{S}+\overline{W} \qquad (3.4)$$

Problema se rezumă la a calcula $\overline{S}$ şi $\overline{S^2}$. Fie $Z=\max\{ W_1+t_1, W_2+t_2, \ldots, W_k+t_k\}$ şi fie variabilele aleatoare (W_I+t_I), I=1,...,K, care au aceeaşi funcţie de densitate de probabilitate, notată cu $f_{W+t}(x)$. Asemănător calculelor făcute în analiza 2-PL şi Datacycle, presupunând că fiecare din aceste K variabile aleatoare sunt independente, funcţia densitate de probabilitate a lui Z se poate obţine din teoria probabilităţilor:

$$f_z(z)=K*f_{W+t}(z)*F_{W+t}^{K-1}(z) \qquad (3.5)$$

unde cu $F_{W+t}(z)$ s-a notat funcţia de repartiţie în probabilitate corespunzătoare lui $f_{W+t}(z)$. $\overline{Z}$ si $\overline{Z^2}$ se poate calcula dacă se cunoaşte $F_{W+t}(z)$. Timpul de aşteptare principal ($\overline{W_I}$) într-o coadă de pretenţii va fi:

$$\overline{W_I}=\frac{\lambda_c*\overline{X_I^2}}{2*(1-\lambda_c*\overline{X_I})} \qquad (3.6)$$

unde λ_c este frecvenţa de sosire a cererilor pentru acest element de date particular, de la toate calculatoarele din SBDD. Vom presupune că toate elementele de date sunt accesate uniform de fiecare tranzacţie care soseşte. Într-o bază de date distribuită pe N calculatoare, cu D elemente de date care sunt accesate uniform şi în care fiecare actualizare accesează K elemente de date, apare următoarea relaţie:

$$\lambda_c=\frac{\lambda*K*N}{D} \qquad (3.7)$$

Algoritmul iterativ cu care se calculează $\overline{Z}$ şi $\overline{Z^2}$ este descris de Shyu şi Li [3]. În momentul în care $\overline{Z}$ şi $\overline{Z^2}$ au fost calculate, se pot calcula $\overline{S}$ şi $\overline{S^2}$ cu formulele de mai jos:

$$\begin{aligned}&\overline{S}=B+\overline{Z}+\overline{E} \text{ şi}\\ &\overline{S^2}=\overline{(B+Z+E)^2}=\\ &=B^2+\overline{Z^2}+\overline{E^2}+2*B*\overline{Z}+\\ &+2*B*\overline{E}+2*\overline{E}*\overline{Z}\end{aligned} \qquad (3.8)$$

4. Rezultate ale simulării

Vom considera o BDD cu trei calculatoare, pe fiecare calculator existând câte un singur element de date. Legăturile între noduri sunt etichetate cu timpul de comunicare (timp de transmisie + timp de întârziere de propagare). Conform relaţiilor obţinute pe baza modelelor matematice, construite anterior, autorii au elaborat un program în Visual C, care alcătuieşte graficele analizelor schemelor Datacycle şi 2-PL şi care face şi o simulare a protocolului "trimite la cerere" pentru a valida rezultatele analizei acestuia.

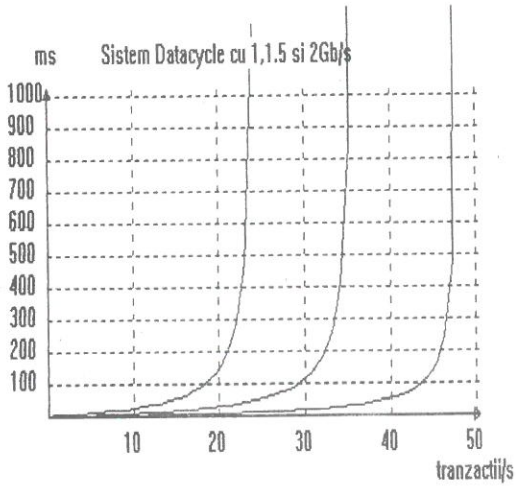

Figura 4.4

Figura 4.4 prezintă comportarea schemei Datacycle pentru un sistem cu tranzacţii numai cu operaţii de citire. La analiză, s-au folosit parametrii următori: G=0.05Gb, L=6000 km, D=100 N=10. S-a considerat timpul de acces al unui singur element de date, distribuit uniform. S-a presupus K (maximul cardinalitatii setului de operaţii de acces) ca fiind 10, cu probabilitatea g_k=1/K (toate cardinalitatile setului de operaţii de acces între 1 si K sunt egal probabile). Timpul de execuţie E a fost presupus zero. O dată cu creşterea vitezei reţelei, rata tranzacţiilor sosite creşte. În cazul în care timpul mediu de execuţie al interogării creşte, creşterea ratei reţelei nu este la fel de eficientă în îmbunătăţirea productivităţii sistemului. Principalul dispozitiv care încetineşte performanţele acum este maşina de baze de date, care procesează interogările.

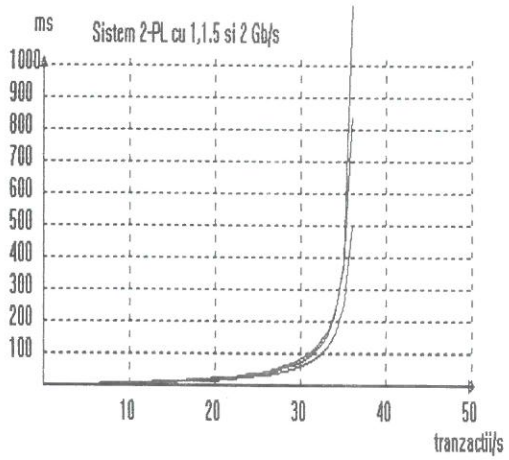

Figura 4.5

Figura 4.5 prezintă comportarea schemei 2-PL pentru un sistem cu tranzacţii numai cu operaţii de citire. Comparând figura 4.4 cu figura 4.5 este evident că schema Datacycle are performanţe mult mai slabe decât schema 2-PL la o rată scăzuta a reţelei (1Gb/s); cele două scheme au performanţe aproape egale la o rată a reţelei de 1.5Gb/s, iar la 2Gb/s schema Datacycle depăşeşte, în performanţe, schema 2-PL.

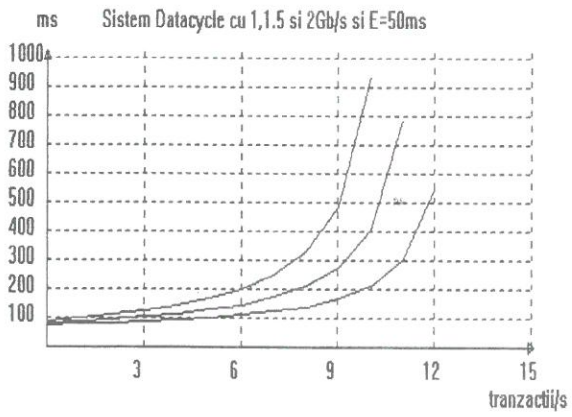

Figura 4.6

Figura 4.6 (obţinută pentru un timp mediu de execuţie a interogărilor de E=50ms) evidenţiază faptul că există o problemă în implementarea bazelor de date, distribuite în reţele ultrarapide, dacă posibilităţile maşinii de baze de date de procesare a interogărilor nu pot fi ţinute la acelaşi nivel (timp mediu de execuţie cât mai mic).

Analizând din nou figura 4.5 se observă că performanţele schemei 2-PL nu se schimbă mult, o dată cu creşterea ratei reţelei. Acest lucru era de aşteptat atât timp cât 2-PL este limitată de întârzierea de propagare între calculatoare, care nu depinde de rata reţelei. Fiindcă performanţa schemei Datacycle depinde direct de timpul de transmisie al întregii baze de date pe magistrala de transmisie, o dată cu creşterea ratei reţelei este de aşteptat ca şi performanţele să se îmbunătăţească.

Între performanţele celor două scheme există un punct de intersecţie pentru o valoare intermediară a ratei reţelei. Schema Datacycle care procesează numai tranzacţii read-only se comportă mai bine decât protocolul 2-PL pentru valori mai mari ale ratei reţelei.. Apare, în schimb, un cost suplimentar, implicat de echiparea calculatoarelor cu mecanisme pentru citirea datelor "în zbor" de pe magistrala comună şi pentru pompa de date centralizată, care trebuie să fie deosebit de sigură în funcţionare.

În analiza actualizărilor în schema Datacycle s-au făcut câteva presupuneri pentru a simplifica calculele. S-a considerat acelaşi sistem de baze de date distribuite, cu aceleaşi valori ca la analiza schemei Datacycle cu tranzacţii numai cu operaţii de scriere. Calculatoarele au fost considerate ca fiind dispuse la distanţe egale pe bus. În cazul actualizărilor, performantele schemei Datacycle se vor micşora, deoarece actualizările presupun un protocol de acces concurent, implementat la nivel de SGBDD, în timp ce tranzacţiile numai cu operaţii de citire pot accesa elementele de date în mod concurent, gestionate de mecanismele de acces concurent la resurse ale sistemului de operare. Schema Datacycle funcţionează mai bine în cazul procesării tranzacţiilor numai cu operaţii de citire. Acest lucru a fost folosit în implementarea noului protocol hibrid "trimite la cerere".

S-a făcut analiza protocolului "trimite la cerere" conform modelului matematic prezentat mai sus. S-a considerat acelaşi SBDD, cu aceiaşi parametri ca în cazul anterior. De această dată, s-au considerat doar tranzacţiile de actualizare, analiza tranzacţiilor de citire făcându-se pe schema Datacycle.

În figura 4.7, se găsesc rezultatele acestei analize pentru capacitatea reţelei de 1Gb/s şi pentru două valori constante ale cardinalităţii setului de operaţii de acces: K=1 si K=4. După cum era de aşteptat, productivitatea sistemului scade o dată cu creşterea numărului de elemente de date solicitat.

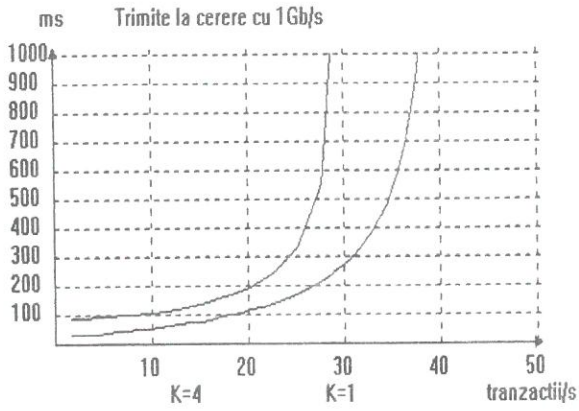

Figura 4.7

Timpul de execuţie E a fost considerat zero, iar timpul mesajului de transmisie B a fost stabilit la o valoare constantă, egala cu timpul necesar unui mesaj ca să se propage în jumătate din reţea. Timpul de transmisie al unui element de date s-a presupus constant. $f_{W+t}(z)$ şi $f_X(z)$ s-au presupus funcţii de densitate uniforme. Pentru a valida rezultatele analizei protocolului "trimite la cerere" s-a făcut o simulare a acestuia cu următorii parametri: G=0.05Gb, L=6000 km, D=100 N=10, iar rata reţelei C=1 Gb/s. Rezultatele simulării sunt evidenţiate în figura 4.9. Programul de simulare este scris în Visual C şi execută 130000 de tranzacţii (10000 de tranzacţii pentru fiecare punct din grafic) la fiecare rulare. Timpul de execuţie E a fost considerat zero, iar durata mesajului de transmisie B a fost setată la o valoare constantă, egală cu timpul necesar unui mesaj să se propage pe o distanţă de 3000 km din reţea.

Simularea s-a făcut doar pentru tranzacţii ce au cardinalitatea setului lor de operaţii de acces K=1. Fiecare punct calculat, în aceasta simulare, necesită un timp de calcul de aproximativ 7 minute pe un sistem Pentium 200MMX cu 32 MB de RAM.

Analiza şi simularea se corespund destul de bine la frecvenţe mici ale sosirii tranzacţiilor în sistem (figura 4.8). La frecvenţe mari de sosire, sunt doi factori care determină ca rezultatele simulării să difere de rezultatele analizei. Primul factor este, eroarea introdusă de aproximaţiile făcute funcţiilor de densitate de probabilitate $f_{W+t}(z)$ si $f_X(z)$. Al doilea factor se datorează dependentelor introduse de ordinea în care anumite pretenţii sosesc în diferitele cozi de pretenţii. În acest protocol, elementele de date se mută de pe un calculator pe altul, în funcţie de cozile lor de pretenţii alcătuite conform tranzacţiilor care le solicită. Timpul necesar unui element de date ca să ajungă la un anumit calculator depinde de calculatorul pe care acesta s-a aflat anterior. Această diferenţă între analiză şi simulare se poate reduce mediind rezultatele mai multor simulări.

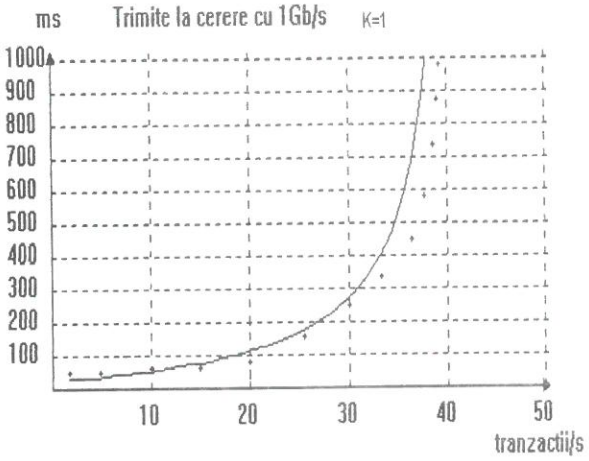

Figura 4.8

5. Concluzii

S-au simulat schemele clasice Datacycle, 2-PL, "trimite la cerere" clasic şi o schemă de control concurenţial, aplicabilă SBDD într-un mediu ultrarapid. S-a verificat, pe un caz particular, că algoritmul care foloseşte schema Datacycle pentru tranzacţii numai cu operaţii de citire este superior, prin diagramele 4.4– 4.8 evidenţiindu-se deosebirile între performanţele de access la date în funcţie de capacitatea reţelei, volumul şi tipul solicitărilor. Analiza şi simulările s-a facut numai asupra sistemelor sigure ca funcţionare (s-a neglijat rezistenţa la defectare şi prin aceasta controlul replicilor, precum şi metodele de recuperare in cazul căderii sistemului) şi numai pentru sisteme care lucrează cu tranzacţii cu seturi de operaţii de acces predeclarate (nu s-a tratat si cazul tranzacţiilor dependente de elementele de date).

Bibliografie

1. **BOWEN, T., GOPAL, G., HERMAN, G., HICKEY, T.M., LEE, K. C., MANSFIELD, W., RAITZ, J., WEINRIB, A.**: The Datacycle Architecture. În: Communications of the ACM, vol. 35, December 1992, pp.71-81.

2. **HERMAN, G., GOPAL, G., LEE, K.C., WEINRIB, A.**: The Datacycle Architecture for very High Throughput Database Systems. În: Proc. of the ACM, SIGMOD, 1987, pp.97-103.

3. **SHYU, S.-C., LI, V.O.K**: Performance Analysis of Static Locking in Distributed Database Systems. În: IEEE Transactions on Computers, vol. 39, June 1990, pp. 741-751.

4. **SCHWARTZ, M.**: Computer Communication Network Design and Analysis, Prentice Hall, 1977.