Modalitati de optimizare a interogarilor asupra bazelor de date în sisteme cu arhitecturi paralele

Dr.ing. Georgios KAPENEKAS
Technical Education Institut of Halkida, Grecia

Paralelismul operatiilor de intrare/iesire se refera la reducerea timpului necesar pentru obtinerea relatiilor de pe suportul disc prin pozitionarea acestora pe mai multe discuri de stocare a informatiei. Forma cea mai comuna de partitionare a datei în mediile baza de date paralele este cea a partitionarii orizontale.
Cuvinte cheie: *optimizare, paralelism, tuplu, disc, relatie, interogare.*

Paralelismul operatiilor de intrare/iesire

În acest caz tupli unei relatii sunt împartiti între mai multe discuri, astfel încât fiecare tuplu sa fie continut în întregime de un singur disc. Pentru a exemplifica diferite strategii de partitionare a datelor presupu-nem ca exista n discuri D_0, D_1, ..., D_{n-1} în care poate fi pastrata informatia.

Odata ce relatia a fost împartita pe diferite discuri, aceasta poate fi refacuta prin citire în paralel folosind toate discurile. În mod analog o relatie poate fi scrisa pe mai multe unitati de disc în paralel. Rata de transfer la citire sau scriere pentru întreaga relatie este mult mai rapida la operatii de intrare/iesire în paralel decât în cazul secvential. Pe de alta parte, în cazul partitionarii relatiilor pe mai multe discuri trebuie tinut cont de faptul ca exista mai multe tipuri de acces la baza de date, functie de cantitatea de infor-matie care este necesar a fi obtinuta:

- acces la întreaga relatie prin parcurgerea tuturor tuplilor;
- localizarea unui tuplu sau cautarea unor tupli care au o valoare specificata pentru un anumit atribut;
- localizarea tuturor tuplilor care contin pentru un atribut dat o valoare care se încadreaza într-un anumit interval.

O prima tehnica posibila de împartire a tuplilor pe discuri este aceea a parcurgerii relatiei într-o ordine oarecare si fiecare tuplu i sa fie transmis catre discul $D_{i \bmod n}$. Aceasta schema asigura o distributie omo-gena a tuplilor pe discuri, fiecare disc are aproximativ acelasi numar de tupli. Meto-da este utila pentru aplicatiile care necesita citirea întregii relatii, în mod secvential, în cadrul interogarilor. Totusi interogarile ca-re localizeaza doar un tuplu sau un interval de tupli sunt complicat de procesat deoa-rece toate cele n discuri trebuie folosite pentru cautare.

O alta tehnica ce poate fi luata în discutie este aceea a partitionarii dispersate. În cazul acestei strategii se pot alege unul sau mai multe atribute din relatiile date ca atribute de dispersie. Se alege o functie de dispersie a carei valori sa fie în intervalul [0, 1, ... , n-1], iar fiecare tuplu al relatiei originale este transmis catre un disc functie de valoarea atributelor selectate si rezul-tatul functiei de dispersie, în sensul ca daca functia de dispersie produce numarul i, tuplul va fi pastrat pe discul D_i. Metoda este deosebit de eficienta în cazul intero-garilor care localizeaza un tuplu. Pornind de la valoarea atributului din tuplul res-pectiv prin aplicarea functiei de dispersie este localizat discul care pastreaza tuplul. Prin directionarea catre un singur disc este redus costul necesar initializarii interogarii pentru discuri multiple si în plus celelalte discuri ramân disponibile pentru a procesa alte

interogari. Aceasta strategie este utila si pentru parcurgerea secventiala a întregii relatii. Daca functia de dispersie realizeaza o împartire aleatoare buna, iar atributele dupa care se face dispersia formeaza o cheie a relatiei, atunci numarul tuplilor pe fiecare disc este aproximativ acelasi cu o mica variatie. Datorita împartirii uniforme timpul necesar parcurgerii relatiei este aproximativ 1/n din timpul necesar par-curgerii relatiei daca ar fi pastrata pe un singur disc.

Totusi aceasta strategie nu este potrivita pentru interogari ce localizeaza tupli dupa atribute diferite de cele care au fost utili-zate în dispersia pe discuri si nici în cazul interogarilor ce au ca rezultat un interval din tupli relatiei deoarece în mod curent functia de dispersie, prin împartirea alea-toare pe discuri, nu pastreaza intervalul unitar, fiind necesar ca toate discurile sa fie parcurse pentru a raspunde la o interogare cu rezultat un interval.

O ultima strategie posibila este partitio-narea pe intervale. Aceasta metoda presu-pune distribuirea de intervale continue ale valorilor atributelor pe fiecare disc. Se alege un atribut de partitionare care va forma vectorul de partitionare. Fie un astfel de vector de valori $[v_0, v_1,...v_{n-2}]$ ordonat crescator, astfel încât daca $i<j$ atunci si $v_i<v_j$. În acest caz relatia va fi împartita dupa cum urmeaza: pentru fiecare tuplu al relatiei daca valoarea atributului ales ca atribut de partitionare este mai mica devât v_0 atunci tuplul este pastrat pe discul D_0, daca valoarea este mai mare decât v_{n-2} tuplul este plasat pe discul D_{n-1},iar daca valoarea a respecta relatia $v_i <a< v_{i+1}$ atunci tuplul este plasat pe discul D_{i+1}. Acest mod de repartizare a tuplilor este deosebit de potrivit în cazul interogarilor care localizeaza o înregistrare sau un interval de înregistrari dupa atributul folo-sit la partitionare. În cazul regasirii unei înregistrari este parcurs vectorul de parti-tionare si este localizat discul pe care se afla pastrat tuplul cautat. În cazul interva-lelor de tuplii din vectorul de partitionare se gasesc unul sau mai multe discuri pe care sunt pastrati tuplii. În ambele cazuri cautarea se continua doar pe discurile care ar putea contine tuplii care prezinta interes în interogare. Aceasta este si un avantaj , dar si un dezavantaj. Avantajul este în cazul în care exista un numar mic de tupli, interogarea va lucra cu un singur disc si nu cu toate. În acest caz celelalte discuri pot fi folosite la alte interogari ceea ce creste foarte mult timpul de raspuns pentru interogari. Pe de alta parte daca exista un numar foarte mare de tupli ce rezulta din interogarea de tip interval trebuiesc cititi mai multi tupli de pe un numar mic de discuri, ceea ce conduce la o gâtuire a operatiilor de intrare/iesire pentru aceste discuri. În aceeasi situatie primele doua strategii de partitionare implica un numar mai mare de discuri asigurând un timp de raspuns mai bun pentru operatiile de intra-re/iesire.

Într-un sistem de calcul cu mai multe discuri, numarul de discuri pe care va fi repartizata relatia va tine cont si de numarul de tuplii din relatie. Daca o relatie contine un numar mic de tupli, care ar putea încapea într-un singur bloc de pe disc, este preferabila pastrarea întregii rela-tii pe un singur disc. Relatiile cu un numar foarte mare de tupli este preferabil a fi îm-partite între toate discurile disponibile. Da-ca relatia consta din m blocuri disc si exis-ta n discuri disponibile în sistem atunci re-latia va trebui alocata pe min(m,n) discuri.

O atentie deosebita trebuie acordata distri-butiei tuplilor când o relatie este partitio-nata, cu exceptia primei metode, deoarece poate aparea o asimetrie în repartitie, cu un procent ridicat de tupli care sunt plasati pe anumite discuri si cu un numar redus pe altele. Asimetria poate aparea din doua cauze: asimetrie datorata valorii atributelor din relatie si o asimetrie de partitionare.

Asimetria atributelor se refera la faptul ca anumite valori apar în cadrul atributelor folosite la partitionare în multi tupli. Toti tuplii care au aceeasi valoare vor fi în final pe acelasi disc generând asimetria. Aceasta poate aparea atât în cazul partitionarii pe intervale, cât si în metoda cu functie de dispersie, deoarece vectorul de partitionare nu a fost ales corespunzator, fie în cazul al doilea functia de dispersie nu este buna.

În cazul în care atributele folosite în repartizarea tuplilor pe discuri formeaza o cheie pentru relatie vectorul de partitionare poate fi construit apelând la o sortare. Se porneste prin sortarea relatiei dupa atribu-tele de partitionare, dupa care relatia este parcursa în ordinea sortarii. Dupa fiecare a 1/n- a parte din relatie valoarea atributelor urmatorului tuplu este adaugata la vectorul de partitionare. În acest mod împartirea pe cele n discuri va fi uniforma. Dezavantajul metodei consta în operatiile suplimentare de intrare/iesire necesare pentru sortarea initiala.

Reducerea operatiilor de intrare/iesire prin optimizarea configuratiei hard

Chiar daca se foloseste un hardware deo-sebit de performant, nu întodeauna este suficient pentru a se obtine o viteza de prelucrare ridicata, principalele obstacole fiind furnizate de viteza relativ scazuta a discurilor care întârzie operatiile de intrare/ iesire, precum si conflictele care apar la scrierea si citirea datelor utilizate în co-mun. Este cunoscut ca operatiile de intrare/ iesire pe disc constituie o strangulare im-portanta a sistemului, fiind necesar un timp de ordinul a 10 milisecunde pentru o astfel de operatie. Pentru a reduce cât mai mult posibil influenta acestor factori se impune pastrarea în memoria principala a relatiilor bazei de date care se utilizeaza. Cresterea numarului de tupli care sunt pastrati în memoria calculatorului conduce la reduce-rea influentei negative a operatiilor de intrare/iesire. Odata ce dimensiunea me-moriei principale creste, tot mai multe aplicatii pot fi executate direct, fara acce-sare la discuri.

Structura hardware pe care o propun pentru a putea utiliza baze de date localizate în memoria principala consta dintr-o arhitec-tura paralela compusa dintr-un numar im-portant de noduri conectate printr-o retea de comunicatie de tip hipercub. În cadrul unor noduri din aceasta structura sunt localizate discuri pentru stocarea externa a datelor. Aceste noduri vor contine mai multe procesoare care pot accesa în comun discurile, precum si memorie cu o dimensiune apreciabila, de ordinul Gb, în care sunt pastrate buffere pentru relatiile bazei de date. Unul din procesoarele nodului este dedicat operatiilor de intrare/ iesire, restul procesoarelor din acelasi nod fiind utilizate în executia aplicatiilor si în prelucrarea paralela a datelor. În cazul în care procesorul specializat pentru operatiile de intrare/iesire se defecteaza, sarcinile sale pot fi preluate de oricare alt procesor aflat în acelasi nod, deoarece toate pot accesa atât memoria interna cât si discurile de stocare.

Toate cererile de citire a datelor vor fi directionate catre procesorul dedicat opera-tiilor de intrare/iesire. Acesta preia cererea de date pe care o analizeaza, iar daca informatia ceruta se afla în bufferul din memoria interna o va transmite imediat catre procesorul sau procesoarele care o solicita. În cazul în care data nu se afla în memoria interna, va fi citita de pe disc. Pentru aceasta mai întâi se verifica daca exista spatiu disponibil pentru a pastra o copie a relatiei în discutie. Daca exista se realizeaza citirea, în caz contrar vor fi analizate restul relatiilor existente în me-morie, iar acelea cu vechimea mai mare si care nu sunt utilizate vor fi eliberate pentru a putea pastra noua relatie.

În cazul în care se doreste scrierea rezultatelor prelucrarii pot aparea urmatoarele situatii:

-datele care urmeaza a fi scrise constituie o relatie care exista anterior. În acest caz va fi actualizata copia din memorie a bazei de date cu marcarea blocurilor care au fost modificate, dupa care va avea loc scrierea efectiva pe disc.

-datele care urmeaza a fi scrise compun o noua relatie. În acest caz se verifica daca mai exista spatiu în memorie, iar daca acesta lipseste, este eliberat spatiul ocupat de relatiile mai vechi. Dupa crearea relatiei în memorie se va trece la inserarea ei si pe disc.

O alta optimizare pe care o propun si este suportata de aceasta arhitectura este aceea a anticiparii de catre schedulerul de tran-zactii a relatiilor ce urmeaza a fi utilizate. Prin aceasta pe durata prelucrarii datelor existente în memorie la un anumit pas se transmite o cerere de aducere în memoria interna de catre procesoarele asociate dis-curilor a relatiilor care vor fi prelucrate în continuare.

Trebuie notat ca blocurile care compun copiile relatiilor pastrate în memoria inter-na care sunt marcate ca modificate de tranzactiile care s-au încheiat vor trebui scrise si pe disc pentru ca în caz de cadere a sistemului sa poata avea loc recuperarea datelor. Pentru a reduce operatiile de intra-re/iesire, în cazul folosirii unor copii a bazei de date pastrate în memorie este utila întârzierea cât mai mult posibil a acestor operatii. Hotarârea scrierii pe disc poate fi luata dupa încheierea a mai multe tran-zactii, dupa un anume interval de timp. Prin aceasta întârzire este de asteptat ca blocurile care sunt scrise sa contina înregistrari modificate de mai multe tran-zactii, care ar fi presupus scrierea aceluiasi bloc pentru fiecare modificare, ceea ce conduce în cele mai multe cazuri la un numar mai scazut de operatii de iesire.

Comunicatia în arhitecturi paralele

În sistemele baza de date informatia este pastrata pe un suport extern. Pentru a putea fi prelucrata ea trebuie citita si transmisa catre procesoare. Cum în general numarul unitatilor de stocare externa este inferior numarului de procesoare, o prima solutie de-a obtine datele necesare procesoarelor ar fi aceea a asigurarii accesului fiecarui procesor la memoria secundara. Principa-lele dezavantaje ale acestei solutii sunt: pe masura ce numarul procesoarelor creste apare o gîtuire pe reteaua de comunicatie, deoarece un singur procesor poate accesa o unitate la un moment dat; durata operatiilor de intrare/iesire este mare, aceeasi infor-matie putând fi citita de mai multe pro-cesoare.

O a doua varianta este ca sa existe procesoare dedicate pentru fiecare unitate de stocare, informatia pastrata pe un anumit disc sa fie citita doar de catre acest procesor asociat si apoi transmisa catre procesoarele care o vor prelucra. Proce-soarele care realizeaza operatiile de intrare/ iesire pe perioada în care nu sunt solicitate pentru citire sau scriere pot fi adaugate la multimea procesoarelor ce participa la prelucrarea datelor. Metoda de specializare a unor procesoare în operatiile de intrare/ iesire are avantajul ca asigura o gestionare mai simpla a datelor, iar daca o aceeasi informatie trebuie transmisa catre mai multe procesoare se realizeaza o singura operatie de intrare consumatoare de timp, dupa care informatia este transmisa prin reteaua de intercomunicatie mult mai rapid catre diferite destinatii. De mentionat ca în acest caz distribuirea datelor de intrare catre procesoare se poate realiza mult mai flexibil si într-un timp mult mai scurt. Sa presupunem ca informatia este citita de catre procesorul1. În primul pas acesta o va transmite catre procesorul2. Iterativ, la fiecare pas, toate procesoarele care au obtinut data o vor

transmite catre alte procesoare care înca nu au primit-o. În acest mod la pasul al doilea procesorul1 transmite data catre procesorul3, iar proce-sorul2 catre procesorul4. În cazul general, putem spune ca daca exista p procesoare care trebuie sa primeasca informatia, acest algoritm permite distribuirea datei de intra-re în log(p) pasi, fata de p-1 pasi daca un acelasi procesor ar trimite informatia catre toate destinatiile.

O alta situatie care apare în cadrul pre-lucrarii paralele este cea în care o multime data de procesoare trebuie sa comunice datele pe care le detin la un moment dat unele catre celelalte astfel ca la sfârsitul comunicatiei toate sa contina aceleasi date. În rezolvarea acestei probleme trebuie avuta în vedere si structura hardware a masinii si anume, principalele doua cate-gorii: o structura de tip inel si una de tip hipercub.

Pentru cazul în care structura este de tip inel se selecteaza o directie în care are loc transmisia datelor, iar fiecare procesor va transmite datele catre procesorul vecin pe aceasta directie. La pasul urmator fiecare procesor va continua transmisia folosind datele pe care le-a receptionat la pasul anterior.

Procesor 1		Procesor 2		Procesor 3		Procesor 4	
data	trans mite	data	trans mite	data	trans mite	data	trans mite
d1	d1	d2	d2	d3	d3	d4	d4
d1d4	d4	d1d2	d1	d2d3	d2	d3d4	d3
d1d3d4	d3	d1d2d4	d4	d1d2d3	d1	d2d3d4	d2
d1d2d3d4		d1d2d3d4		d1d2d3d4		d1d2d3d4	

Din tabel se observa ca daca exista p procesoare conectate în inel pentru ca datele continute de fiecare în parte sa fie comunicate catre toate procesele sunt nu-meric (p-1) pasi de comunicatie dupa metoda descrisa.

Presupunem ca avem $p=2^d$ procesoare iar structura de conectare a procesoarelor este de tip hipercub. În acest caz putem spune ca avem d directii dupa care sunt legate procesoarele. La primul pas al algoritmului se selecteaza o directie, cu ajutorul careia procesoarele sunt împartite în doua grupe egale, iar fiecare procesor din primul grup este conectat direct cu un procesor din al doilea grup. Între toate aceste perechi de procesoare are loc schimbul tuturor datelor pe care le pastreaza fiecare. În urma acestui pas exista p/2 perechi de procesoare care contin aceleasi date. La pasul urmator se alege alta directie care va împarti multimea tuturor procesoarelor în perechi care vor comunica direct toate datele care le detin. În acest mod în fiecare etapa fiecare procesor îsi va dubla datele pe care le contine. Numarul total de pasi în care se încheie comunicatie este $d=\log_2(p)$.

O alta problema care apare în prelucrarea paralela este aceea a modului în care se împart datele între procesoarele care con-cura la realizarea unei operatii relationale. Presupunem ca exista o relatie R care contine n înregistrari. Deoarece se doreste ca munca sa fie împartita egal intre cele p procesoare este necesar ca numarul de înregistrari care se transmite fiecarui procesor sa fie aproximativ acelasi, si anume n/p articole. Sugeram trei metode de distribuire:

- distribuire în bloc. În acest caz daca numarul articolelor este divizibil prin numarul procesoarelor p primele n/p elemente sunt transmise catre un procesor, urmatoarele n/p catre altul si asa mai departe catre toate

procesoarele. Daca n nu este divizibil cu p ,deci n=qp+r, se transmit grupuri de (n/p)+1 aricole catre primele r procesoare si (n/p) articole catre restul procesoarelor.
- distribuire ciclica. Se transmite câte un articol este catre un alt procesor. Când fiecare din cele p procesoare a primit câte o noua înregistrare si mai sunt înregistrari se reia distribuirea articolelor cu primul procesor, pâna când toate elementele au fost transmise.
- distribuire combinata. În acest caz se transmite un numar de înregistrari catre fiecare procesor în parte. Dupa ce a fost atribuit câte un bloc catre fiecare procesor, împartirea articolelor se continua iterativ cu blocuri de aceeasi lungime începând cu primul procesor.
Un alt mod de obtinere a datelor de prelucrat este dat de utilizarea unei struc-turi pipe. Astfel, daca o multime data de procesoare realizeaza o operatie relationa-la, pe masura ce tuplii sunt obtinuti, acestia vor fi transmisi catre un alt set de procesoare pentru a realizeaza o noua operatie implicata în interogare si care necesita ca operand rezultatul obtinut anterior. În acest caz se reduce numarul de fisiere temporare, ceea ce diminueaza numarul operatiilor de intrare/iesire. Prac-tic pentru fiecare fisier temporar este necesara o operatie de scriere, urmata de una de citire. Daca rezultatul intermediar este implicat în mai multe operatii de din interogare, poate fi transmis catre mai multe seturi de procesoare care realizeaza operatii diferite în paralel.

Paralelismul interogarilor

Pentru a îmbunatati timpul de raspuns la diferite interogari pot fi luate în calcul doua posibilitati:
- executia în paralel a diferitelor interogari si tranzactii;
- executia paralela a interogarilor sau a tranzactiei.

Prin executia în paralel a mai multor interogari sau tranzactii se îmbunatatesc performantele sistemului de gestiune baza de date pe ansamblu, dar timpul de raspuns pentru tranzactiile individuale nu este mai bun decât în cazul în care tranzactiile s-ar realiza individual.
Executia în paralel a tranzactiilor este forma cea mai usoara de paralelism care poate fi implementata într-un sistem baza de date, în particular, într-un sistem paralel cu partajarea memoriei. Sistemele baza de date dezvoltate pentru sisteme cu un singur procesor pot fi folosite cu mici schimbari în arhitecturile paralele cu partajarea memoriei, deoarece chiar si sistemele baza de date secventiale pot implementa procesarea concurenta. Tranzactiile care se pot desfasura pe o masina secventiala folosind concurenta prin partajarea temporara pot opera în paralel pe arhitecturile paralele cu parta-jarea memoriei.
Implementarea executiei în paralel a tran-zactiilor este mult mai complicat de realizat în arhitecturi cu partajarea discu-rilor sau în cele fara partajare. În acest caz procesoarele trebuie sa realizeze anumite operatii, cum ar fi blocarea si deblocarea, în mod unitar, ceea ce implica necesitatea schimbului de mesaje între ele. Sistemul baza de date paralel trebuie sa evite ca doua procesoare sa actualizeze aceeasi data independent în acelasi timp. În plus, când un procesor acceseaza sau actualizeaza o data, sistemul baza de date trebuie sa asigure ca acest procesor sa utilizeze ultima versiune a datei, problema cunos-cuta si sub denumirea de coerenta cache. În general protocolul ce asigura coerenta datelor este stâns legat de protocolul care asigura controlul concurentei. Un astfel de protocol pentru un sistem cu partajarea discurilor trebuie sa asigure:
- înainte de orice acces pentru citire sau scriere a unei pagini, tranzactia blocheaza pagina fie în mod partajat, fie în mod exclusiv functie de operatia care o va executa. Dupa

ce tranzactia reuseste sa blocheze pagina de care are nevoie va citi copia cea mai recenta a paginii de pe discul comun;
- înainte ca tranzactia sa elibereze un blocaj de tip exclusiv pentru o pagina, va trebui sa scrie pagina actualizata pe discul comun, dupa care se poate elibera blocajul pe pagina respectiva.

Acest protocol asigura ca atunci când o tranzactie fixeaza un blocaj comun sau exclusiv pentru o pagina, ea va obtine o copie corecta a paginii.

Protocolul poate fi îmbunatatit pentru a evita scrierea si citirea în mod repetat pe disc. Pentru aceasta paginile nu mai sunt înscrise pe disc atunci când blocajul de tip exclusiv este eliberat. Când un blocaj partajat sau exclusiv se obtine versiunea cea mai recenta a paginii este preluata direct din memoria procesorului care a actualizat-o ultimul.

Trebuie subliniat ca protocolul descris mai sus este valabil pentru sisteme în care discurile sunt utilizate în comun. Acest protocol se poate extinde si pentru arhi-tecturi în care nu exista componente folosite în comun. Astfel, fiecare pagina are un procesor P_i propriu si este pastrata pe discul D_i. Când un alt procesor doreste sa citeasca sau sa scrie o pagina, acesta va trimite o cerere catre procesorul P_i propriu paginii, deoarece el nu poate comunica direct cu discul.

Executia paralela a unei tranzactii presu-pune realizarea unei singure interogari în paralel pe mai multe procesoare sau discuri. Folosirea acestei metode este importanta în îmbunatatirea performantelor pentru interogari ce necesita un volum mare de date si implicit, un timp de executie mare.

Pentru a ilustra evaluarea paralela a unei interogari, vom considera necesitatea de a sorta o relatie. Relatia poate fi împartita pe mai multe discuri folosind partitionarea pe intervale dupa un anumit atribut, acelasi atribut dupa care se doreste realizarea sortarii. În continuare operatia de sortare se poate realiza în paralel pe fiecare partitie în parte, iar în final partitiile sortate vor fi concatenate pentru a obtine relatia finala. Aceasta este varianta de executie paralela a unei interogari prin paralelizarea operatii-lor individuale. Paralelismul poate merge si mai departe în evaluarea unei interogari, dupa formarea arborelui de operatii pentru o interogare ce contine mai multe operatii, acesta poate fi evaluat prin executie în paralel a operatiilor care nu depind una de alta sau mai mult, se poate folosi efectul de conducta, în sensul ca iesirea unei operatii executata pe un procesor constituie o in-trare pentru alta operatie executata pe un alt procesor separat. Se poate concluziona ca executia paralela a unei singure inte-rogari poate fi realizata în doua moduri:
- se poate îmbunatati performanta siste-mului paralel în procesarea unei interogari prin executia paralela a fiecarei instructiuni în parte cum ar fi sortarea, selectia, pro-iectia sau jonctiunea;
- sau prin executia în paralel a diferitelor operatii ce compun expresia interogarii.

Aceste doua forme de paralelism sunt com-plementare si pot fi utilizate simultan în interogari. Numarul operatiilor în cadrul unei interogari este relativ mic în com-paratie cu numarul de tupli care sunt preluati de fiecare operatie în parte, ceea ce conduce la rezultate bune privind cresterea performantelor sistemului daca se utilizea-za prima forma de paralelism. Pe de alta parte în cazul în care numarul proce-soarelor din sistem este mic ambele forme ale paralelismului sunt importante.

Pentru algoritmii prezentati în continuare presupunem ca exista n procesoare P_0, ...,P_{n-1} si n discuri D_0, ..., D_{n-1} în care fiecare disc D_i este asociat unui procesor P_i. Vom presupune utilizarea unei arhi-tecturi în care nu exista nici o resursa în comun, ceea ce implica

descrierea si a momentului în care se realizeaza transferul datelor de la un procesor la altul. De notat ca s-a ales cazul general, acesta putând fi usor simulat si pe alte arhitecturi, deoarece transferul datelor poate fi realizat în alte cazuri fie cu ajutorul memoriei comune în cazul arhitecturii cu partajarea memoriei, fie cu discurile comune în cazul arhitecturii cu partajarea discurilor.

Sortarea paralela a unei relatii

Presupunem ca se va sorta o relatie care este localizata pe n discuri D_0, ..., D_{n-1}. Daca relatia este partitionata în intervale functie de atributul dupa care va fi sortata, se poate realiza direct sortarea fiecarei partitii în parte, iar la sfârsit rezultatele obtinute vor fi concatenate pentru a obtine întreaga relatie sortata. Cum tuplii sunt împartiti pe n discuri timpul necesar citirii întregii relatii va fi redus si de accesul paralel la date. În cazul în care relatia a fost împartita folosind o metoda de partitionare se pot utiliza doi algoritmi de sortare:

Algoritm de sortare cu împartire pe intervale

Se poate realiza partitionarea dupa atribu-tele dupa care se doreste sortarea, iar fiecare grup de tupli obtinut va fi sortat separat. La sortarea cu partitionarea în intervale a relatiei, cazul cel mai complex este acela în care împartirea relatiei nu se realizeaza cu aceleasi procesoare sau dis-curi cu care relatia va fi sortata. Presu-punem ca folosim m<n procesoare P_0, P_1,...P_m pentru a sorta relatia. În acest caz operatia se desfasoara în trei etape:

1. relatia este redistribuita folosind o strategie de partitionare în intervale, astfel încât toti tuplii care sunt în intervalul i sunt transmisi catre procesorul P_i si care va pastra tupli temporar pe discul D_i. Aceasta faza a prelucrarii presupune operatii suplimentare de intrare/iesire catre unitatile de disc si comunicatie între procesoare.
2. Fiecare procesor sorteaza portiunea repartizata din relatie, local, fara a interactiona cu alte procesoare. Fiecare procesor executa aceeasi operatie, dar pe o multime diferita de date. De remarcat în acest caz necesitatea utilizarii unui vector de partitionare echilibrat pentru a asigura un numar egal de tupli.
3. Când fiecare procesor a sortat tupli din relatie se executa operatia de fuzionare în care functie de locul în care se doreste pastrarea relatiei va necesita sau nu transmisie de date între procesoare si operatii de intrare/iesire cu discul

Algoritm de sortare externa cu con-catenare

Presupunem ca relatia care se sorteaza este partitionata pe n discuri D_0, ..., D_{n-1}. În acest caz sortarea externa presupune urmatoarele etape:

1. fiecare procesor P_i va sorta local data aflata pe discul D_i.
2. portiunea sortata de fiecare procesor P_i este împartita în intervale, folosind un ace-lasi vector de catre toate procesorele, fiind transmise catre procesoarele P_o,...P_{m-1}. Transmiterea tuplilor se realizeaza în ordinea sortarii, astfel încât fiecare proce-sor va receptiona tuplii aflati într-un sir sortat;
3. dupa ce partitionarea a fost realizata si comunicatia dintre procesoare s-a încheiat, fiecare procesor P_i va fuziona sirurile primite pentru a forma un singur sir ordonat;
4. sirurile obtinute de fiecare procesor în parte P_o,... P_{m-1} sunt concatenate pentru a obtine relatia sortata.

Algoritmi de jonctiune paralela a relatiilor

Operatia de jonctiune paralela a relatiilor presupune testarea perechilor de tupli din relatii diferite pentru a verifica daca sati-sfac conditia jonctiunii. Daca aceasta con-ditie este îndeplinita perechea este adau-gata relatiei

jonctiune de iesire, altfel se trece la testarea altei perechi de tupli. Algoritmii de executie paralela a jonctiunii presupun împartirea perechilor care vor fi testate la mai multe procesoare. Fiecare procesor va prelua local datele primite. În final trebuie refacuta relatia din rezultatele obtinute de fiecare procesor. Modul în care sunt repartizati tuplii catre procesoare pen-tru a fi prelucrati determina variante de realizare paralela a acestei operatii.

Jonctiunea partitionata

Acest algoritm de calcul este util în cazul anumitor tipuri de jonctiuni, cum ar fi jonctiunea naturala si equi-jonctiune, deoa-rece este posibila partitionarea tuplilor celor doua relatii de intrare pe procesoare si calcularea jonctiunii local la fiecare procesor. Presupunem ca se folosesc n procesoare, iar cele doua relatii pentru care se realizeaza jonctiunea sunt R si S. Fiecare relatie va fi împartita în n partitii, notate R_0, R_1,... R_{n-1} si respectiv S_0, S_1, ... S_{n-1}. Fiecare pereche de partitii R_i si S_i, i=0, n-1 va fi transmisa catre procesorul P_i, unde se vor desfasura calculele. Aceasta tehnica lucreaza optim daca jonctiunea este de tipul R ∞ S în care avem clauza R.A. = S.B., iar partitia relatiilor R si S se realizeaza utilizând aceeasi functie de partitionare folosind atributele implicate în jonctiune. Se poate apela la doua moduri de împartire a relatiilor R si S: partitiona-rea pe intervale dupa atributele jonctiunii sau partitionarea cu dispersie. În ambele cazuri functia de partitionare trebuie sa fie aceeasi, ceea ce în cazul partitionarii pe intervale implica utilizarea aceluiasi vector de partitionare pentru ambele relatii R si S, iar pentru partitionarea cu dispersie functia sa fie aceeasi.

Dupa partitionarea pe grupe a relatiilor se poate apela local pentru fiecare procesor P_i la oricare tehnica de calcul a jonctiunii pentru R_i si S_i.

Daca una sau chiar amândoua din relatiile R si S sunt deja partitionate dupa atributele implicate în jonctiune numarul operatiilor pentru prelucrare scade foarte mult. Daca relatiile nu sunt partitionate sau partitiona-rea a fost facuta dupa alte atribute, atunci tuplii relatiilor vor trebui grupati din nou. Fiecare procesor P_i va citi tupli de pe discul D_i, va calcula noua partitie j careia tuplul îi apartine si va trimite data catre procesorul P_j. Odata tuplul receptionat de catre procesorul P_j va fi depus pe discul D_j. Se poate optimiza algoritmul prin reduce-rea operatiilor de intrare/iesire prin pastra-rea unor tupli într-un buffer din memorie, fara a scrie pe disc.

O alta problema ce trebuie subliniata este cea a nesimetriei ce poate aparea atunci când se foloseste partitionarea pe intervale. Daca un vector de partitionare poate împarti una din relatii în grupuri relativ egale ca dimensiune, este posibil ca cea de-a doua relatie sa fie împartita în grupuri la care numarul tuplilor sa varieze foarte mult. În acest caz este preferabila selectia unui vector de partitionare pentru care sumele partiale ale tuplilor din relatia R_i si S_i cu i=0, n-1 sa fie relativ egale. În cazul utilizarii unei functii de dispersie la partitionarea relatiilor, prin selectia unei functii adecvate pentru o relatie este foarte posibil sa se elimine asimetria pentru ambele relatii.

Presupunem ca exista n procesoare P_0, P_1, ... P_{n-1} si doua relatii R si S care sunt pastrate partitionat pe mai multe discuri. Pentru a calcula jonctiunea se începe cu relatia ce contine mai putini tupli. Fie relatia mai mica S. Pentru a realiza ope-ratia de jonctiune a celor doua relatii se procedeaza astfel:

- se alege o functie de dispersie h care primeste ca argument fiecare valoare a atributelor implicate în jonctiune din tupli lui S si asociaza întregul tuplu unui procesor din cele n. În acest mod fiecare procesor P_i va citi

tupli relatiei S care se afla pe propriul disc D_i si transmite procesorului rezultat în urma evaluarii functiei de dispersie h_1. Notam cu S_i tuplii din relatia S care sunt transmisi catre procesorul P_i.

- dupa ce tuplii relatiei S au fost primiti la procesorul destinatie P_i vor fi împartiti în continuare folosind o alta functie de dispersie h_2, pe care procesorul o foloseste pentru a calcula jonctiunea local. Aceasta faza de împartire pe grupe a tuplilor se executa de catre fiecare procesor P_i independent de restul procesorelor.
- dupa redistribuirea lui S, sistemul va trece la redistribuirea relatiei R pentru cele m procesoare folosind functia de dispersie h_1. Pe masura ce tupli sunt receptionati procesorul destinatie îi va repartitiona utilizând functia de dispersie h_2.
- în acest moment fiecare procesor P_i are doua relatii R_i si S_i formate din tuplii relatiilor initiale R si S implicate în jonctiune care sunt împartite în subgrupe $R_{i,j}$ si S_{ij} cu ajutorul functiei de dispersie h_2. Toate subgrupele R_{ij} cu j=1,m formeaza relatia R_i si similar S_j cu j=1,m formeaza relatia S_i. În continuare fiecare procesor P_i va încarca în memoria proprie câte un grup S_{ij} de tupli pe care îi va compara cu toti tuplii din grupul R_{ij} pentru a obtine jonctiunea relatiilor. Prin repartitia folosind functia de dispersie h_2 se asigura ca nu exista alte subgrupe S_{in} si R_{im} care sa contina aceleasi valori pentru atributele implicate în jonctiune decât cele pentru care n=m.

Jonctiunea fragmentata

Metoda descrisa anterior nu poate fi aplicata în toate cazurile. Daca clauza pentru jonctiune este de tip inegalitate R.A < S.B (R ∞ R.A<S) este posibil ca toti tuplii dintr-o relatie sa poata realiza oparatia de jonctiune cu anumiti tupli din cealalta relatie, ceea ce face imposibila partitionarea relatiilor pe mai multe proce-soare dupa metoda anterioara. Pentru a realiza o prelucrare paralela în acest caz se foloseste o alta tehnica pentru jonctiune, si anume, fragmentarea si copierea. Modul de lucru al acesteia este urmatorul:

- una din relatii va fi partitionata dupa o tehnica oarecare: împartirea pe intervale, dispersia sau repartizarea uniforma. Presu-punem ca se utilizeaza relatia R.
- cea de-a doua relatie este transmisa catre toate procesoarele care contin un grup al primei relatii;
- fiecare procesor P_i va calcula local jonctiunea relatiilor R_i si S.
- relatia jonctiune a celor doua R si S se obtine din totalitatea rezultatelor de fiecare procesor în parte.

Daca numarul de procesoare este mare tehnica descrisa anterior poate fi genera-lizata. Pentru aceasta relatia R este îm-partita în n relatii R_0, R_1, ..., R_{n-1}, iar relatia S este împartita m relatii S_0, S_1, ..., S_{m-1}. Cele doua valori n si m nu este obligatoriu sa fie egale, singura cerinta este ca alegerea lor sa fie astfel realizata încât sa existe cel putin mxn procesoare. Daca procesoarele sunt notate $P_{0,0}$, $P_{0,1}$, …, $P_{n-1,\ m-1}$ atunci putem spune ca procesorul P_{ij} va calcula jonctiunea între relatiile R_i si S_j. De aici se observa cum copii ale relatiei R_i trebuie transmise catre procesoarele $P_{i,0}$, $P_{i,1}$, ...$P_{i,m-1}$ iar copii ale relatiei S_i vor fi trimise catre procesoarele $P_{o,i}$, $P_{1,i}$, ... $P_{n-1,i}$. Dupa ce copiile au fost transmise catre proce-soare, acestea pot folosi orice tehnica de realizare a jonctiunii.

Aceasta metoda de calcul a jonctiunii poate accepta orice clauza întrucât fiecare tuplu din R este testat cu fiecare tuplu din S.

Metoda cu fragmentare si copiere are un cost mai mare decât metoda cu partitionare în cazul în care ambele relatii sunt de aceeasi marime, deoarece cel putin o relatie va trebui transmisa catre toate procesoarele. Totusi, daca una din relatii are un numar mai mic de

tupli este mai economic sa fie transmisa catre toate procesoarele, decât sa se realizeze o repar-titie a relatiilor R si S dupa atributele din jonctiune.

Bibliografie

[1]. M.Petrescu, *Baze de date* - note de curs, 1992;
[2]. M.Petrescu, *Proiectarea bazelor de date* - note de curs, 1993;
[3]. D. Bell, J. Grimson - *Distributed Database System* - Adisson Wesley 1992;
[4]. E. F. Codd - *The Relational Model for Database Management* - Adisson Wesley 1990;
[5]. T. Carmen; C. Leiserson, R. Rivest - *An Introduction to Algortims* Mc Graw Hill, 1990;
[6]. W. Kim - *Modern Database System* - Adisson Wesley, 1995;
[7]. George S. Almasi, Allan Gottlieb *Highly Parallel Computing* Benjamin/Cummings,1994;
[8]. Ian Foster *Designing and Building Parallel Programs* Addison-Wesley, 1995;
[9]. Geoffrey Fox *Solving Problems on Concurrent Processors* Prentice Hall, 1988;
[10]. Gene Golub, James M. *Ortega Scientific Computing:An introduction with Parallel Computing* Academic Press, 1993;
[11]. Vipin Kumer *Introduction to Parallel Computing: Design and Analysis of Algorithms* Benjamin/Cummings,1994;
[12]. Peter S. Pecheco *Parallel programing with MPI* Morgan Kaufmann Publishers, 1997.