STUDIA INFORMATICA 2004
Volume 25 Number 3 (59)

Katarzyna HARĘŻLAK
Politechnika Śląska, Instytut Informatyki

EFEKTYWNOŚĆ REALIZACJI TRANSAKCJI ROZPROSZONYCH Z WYKORZYSTANIEM WIRTUALNIE WSPÓŁDZIELONEJ PAMIĘCI

Streszczenie. W artykule przedstawiona została asynchroniczna metoda aktualizacji kopii danych w środowisku wirtualnie współdzielonej pamięci. W metodzie tej zaproponowano mechanizmy umożliwiające wykonanie transakcji rozproszonej tylko w jednej kopii danych, pozostałe uaktualnia się po jej zakończeniu. Dla zaproponowanych rozwiązań przeprowadzono analizę czynników wpływających na efektywność realizacji transakcji.

Słowa kluczowe: rozproszona baza danych, transakcje, wirtualnie współdzielona pamięć.

EFFICIECY OF DISTRIBUTED TRANSACTION REALIZATION WITH USAGE OF VIRTUAL SHARED MEMORY

Summary. The paper presents the new, asynchronous update method of replicated data in virtual shared memory environment. The method advantage is possibility of utilizing only one data copy for distributed transaction realization contemporarily guarding against replicated data update conflicts. The problem of distributed database transaction performing efficiency was also discussed.

Keywords: distributed database, transactions, virtual shred memory.

1. Wstęp

Wraz z dynamicznym rozwojem sieci komputerowych otworzyły się szersze możliwości dostępu do różnorodnych informacji, a w szczególności do tych, które przechowywane są w bazach danych znajdujących się w różnych węzłach sieci. W efekcie nastąpił rozwój systemów zarządzania rozproszoną bazą danych, dzięki którym instytucje o złożonej

i rozproszonej strukturze organizacyjnej uzyskały narzędzia do wydajnego zarządzania ich procesami biznesowymi.

Jedną z zalet takich systemów rozproszonych jest zdolność powielania (replikacji) danych, tzn. tworzenia kopii w różnych węzłach sieci komputerowej. Zaletą takiego podejścia jest zwiększenie efektywności dostępu do danych oraz zwiększenie niezawodności systemu, który w przypadku awarii jednej bądź kilku kopii danych może, na potrzeby przetwarzania biznesowego, dostarczyć dane z innych funkcjonujących węzłów systemu.

Wykorzystywanie kopii danych zwiększa jednak złożoność mechanizmów stosowanych do zarządzania danymi. Dotyczy to między innymi wykonywania transakcji. Zachowanie spójności danych we wszystkich kopiach wymaga stosowania odpowiednich protokołów realizacji transakcji [3, 7, 8], które odzwierciedlają zmiany dokonane w jednej kopii, także w pozostałych kopiach danych. Przeprowadzenie takich transakcji powinno być wspierane przez odpowiednią warstwę komunikacyjną systemu rozproszonego.

2. Wirtualnie współdzielona pamięć

Wśród narzędzi programowych wykorzystywanych do komunikacji pomiędzy elementami systemów rozproszonych można wyróżnić dwie grupy, biorąc jako podstawę ich podziału sposób wymiany informacji pomiędzy programami realizującymi obliczenia. Może ona odbywać się z zastosowaniem modelu opartego na przesyle komunikatów lub modelu pamięci współdzielonej [11, 12]. Do pierwszej grupy należą między innymi takie rozwiązania, jak PVM, MPI, RPC, do drugiej zaliczamy systemy wykorzystujące model Lindy, np. system Paradise.

Ten drugi rodzaj komunikacji wykorzystany został w eksperymentalnym systemie zarządzania rozproszoną bazą danych (ESZRBD) [5, 6], w którym prowadzone były badania dotyczące realizacji transakcji rozproszonej, w szczególności odnoszącej się do kopii danych [9, 10]. Zastosowany tam system Paradise stanowi środowisko równoległego i rozproszonego programowania, wykorzystujące model wirtualnej współdzielonej pamięci VSM (ang. *Virtual Shared Memory*). Model ten tworzy wiele przestrzeni danych, które stanowią wspólną pamięć dla wielu niezależnych programów. Niektóre z przestrzeni mogą posiadać atrybut trwałości, co oznacza, że są zapisywane w pamięci zewnętrznej [14]. Ogólna zasada korzystania z wirtualnie współdzielonej przestrzeni polega na tworzeniu i wymianie obiektów, którymi w przypadku systemu Paradise są krotki (ang. *tuples*). Stąd w tym systemie wirtualnie współdzielona przestrzeń określana jest jako przestrzeń krotek *TS* (ang. *Tuple Space*).

3. Asynchroniczna metoda aktualizacji kopii danych

Transakcje dotyczące kopii danych mogą być realizowane w sposób synchroniczny z udziałem wszystkich węzłów, zawierających kopie danych, bądź asynchronicznie z wykorzystaniem jednego lub kilku węzłów. Głównymi problemami tego drugiego rozwiązania jest zabezpieczenie przed dostępem do nieaktualnych danych oraz zapewnienie ich uaktualnienia w czasie możliwie jak najkrótszym [1, 2, 3, 4].

W ESZRBD opracowano asynchroniczną metodę aktualizacji kopii danych, w której rozwiązanie tych problemów odbywa się z wykorzystaniem mechanizmów *lokalnej* i *globalnej wersji* danych oraz *historii wersji i historii transakcji*.

Numerem lokalnej wersji NLW kopii danej *x* jest liczba całkowita, która określa liczbę zatwierdzonych transakcji w węźle *s*. W stanie początkowym numer ten, utrzymywany przez zarządcę kopii pracującego w tym węźle, przyjmuje wartość 0 i jest inkrementowany o 1 po każdym wykonaniu tam transakcji.

Numerem globalnej wersji NGW danej *x* jest maksimum z numerów lokalnych wersji wszystkich jej kopii. Numer ten reprezentowany jest przez dryfującą w przestrzeni krotek krotkę, która dodatkowo zawiera pola informujące o stanie aktualnych blokad dotyczących operacji zapisu. Struktura krotki przedstawia się następująco:

{identyfikator krotki, nr jednostki x, NGW, blokada, nr węzła blokującego, czas założenia}.

Zarządca, chcący zrealizować transakcję na swojej kopii danych, musi najpierw zablokować do niej dostęp poprzez wprowadzenie odpowiednich wartości do pól krotki numeru globalnej wersji danych. W przypadku kiedy kopia jest już zablokowana, musi ustawić się w kolejce transakcji oczekujących na transakcję. Kolejka ta budowana jest w przestrzeni krotek z obiektów o postaci:

{identyfikator krotki, nr jednostki x, nr węzła oczekującego, nr w kolejce, czas ustawienia}.

Numer w kolejce określany jest na podstawie krotki:

{identyfikator krotki, nr jednostki x, maksymalny numer w kolejce}.

Drugim warunkiem, który musi być spełniony, by transakcja mogła być realizowana, jest zgodność numeru lokalnej wersji jednostki *x* w węźle *s* z jej numerem globalnej wersji. W przypadku gdy nie są one zgodne, zarządca kopii musi najpierw zrealizować wszystkie zaległe transakcje znajdujące się w historii transakcji, które powodują różnice w wersjach danych. ***Historia transakcji*** to uporządkowany zbiór dryfujących krotek zawierający opis transakcji wykonanych na którejkolwiek kopii danej:

{identyfikator krotki, nr transakcji, nr węzła realizującego transakcję, liczba operacji, zapis w języku SQL, liczba serwerów mających wykonać transakcję}.

Dostęp do tej historii odbywa się poprzez ***historię wersji***, która odwzorowuje kolejne wersje danych na transakcje, które je utworzyły: {identyfikator krotki, nr wersji, nr transakcji, nr węzła realizującego transakcję}.

W prezentowanej metodzie realizacja transakcji przebiega w następujących fazach:

Faza 1

1. Pobranie i dekompozycja operacji należącej do transakcji.
2. Sprawdzenie stanu blokad.
3. Jeżeli nie założono konfliktowej blokady – zablokowanie jednostki *x*.
4. Porównanie lokalnej i globalnej wersji danych.
5. Gdy wersje nie są równe – uaktualnienie lokalnej wersji danych przez realizację transakcji zawartych w historii transakcji.

Faza 2

1. Realizacja bieżących operacji na lokalnej bazie danych.

Faza 3

1. Wprowadzenie transakcji do historii transakcji.
2. Wprowadzenie nowego elementu do historii wersji.
3. Uaktualnienie numeru lokalnej wersji danych.
4. Uaktualnienie numeru globalnej wersji danych i zwolnienie blokad.

Przebieg transakcji można opisać diagramami przedstawionymi na rysunkach 1 i 2.

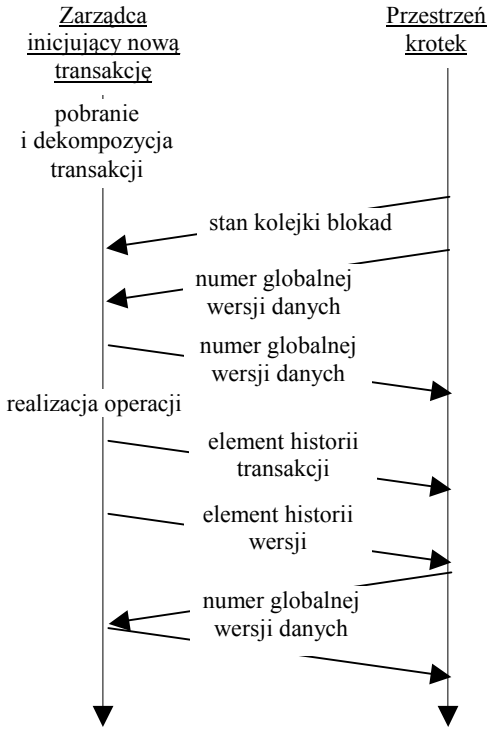
4. Analiza efektywności asynchronicznej metody aktualizacji kopii danych

Analizując składniki, które wpływają na czas wykonania transakcji zgodnie z proponowaną metodą, można stwierdzić, że są to:

- czas jednokrotnego kontaktu z przestrzenią krotek, oznaczmy go jako t_{TS},
- czas wstawienia krotki do przestrzeni krotek, t_{OUT},
- czas odszukania właściwej informacji, t_{LOOK},
- czas realizacji transakcji w lokalnej bazie danych, t_{BD}.

Całkowity czas wykonania transakcji opisany jest za pomocą wzoru:

$$T_{CAŁK} = T_{CAŁK\_TS} + T_{CAŁK\_OUT} + T_{CAŁK\_LOOK} + T_{CAŁK\_BD} \quad (1)$$



Rys. 1. Diagram zależności czasowych realizacji transakcji w warunkach zgodności wersji i braku współbieżnie realizowanej transakcji

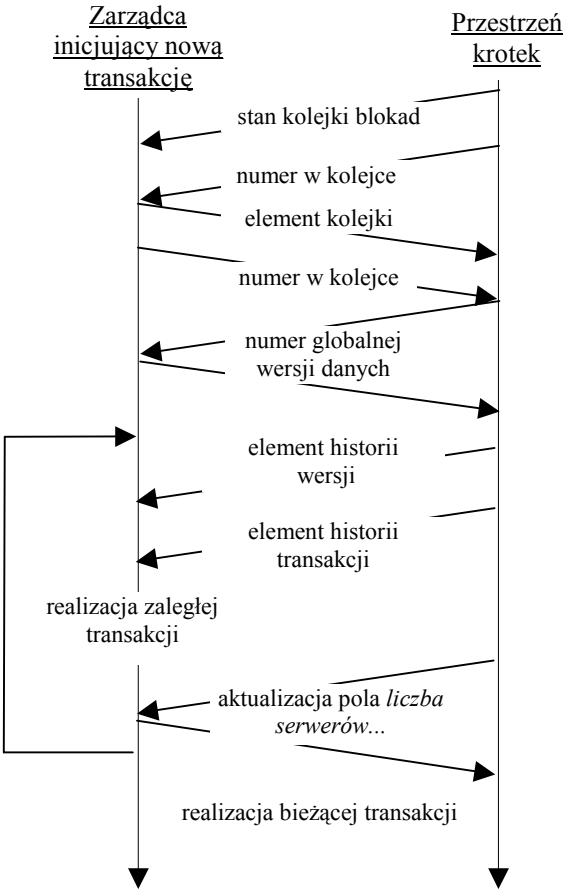
Fig. 1. The time dependency diagram of transaction performing in conditions of versions equality and absence of any concurrent transaction

Dla sytuacji, w której transakcja uzyskuje dostęp do globalnej wersji danych bez czekania w kolejce oraz operuje na aktualnych danych, koszt kontaktu z przestrzenią krotek można rozbić na następujące składniki:

- czas związany z analizą stanu kolejki,
- cztery kontakty związane z założeniem i zdjęciem blokad,
- dwa kontakty związane ze wstawianiem elementów historii wersji oraz historii transakcji.

Natomiast koszt poszukiwania i wstawiania informacji obejmuje:

- znalezienie elementu kolejki, w tym przypadku stwierdzenie jej braku,
- dwukrotny koszt wstawiania globalnej wersji danych przy zakładaniu i zwalnianiu blokad oraz dwukrotny koszt związany ze wstawieniem krotek historii wersji i historii transakcji, dwukrotny koszt poszukiwania globalnej wersji danych w celu założenia i zwolnienia blokady.



Rys. 2. Diagram zależności czasowych realizacji transakcji przy braku zgodności wersji, z istnieniem współbieżnie realizowanych transakcji

Fig. 2. The time dependency diagram of transaction performing in conditions of lack versions equality and concurrent transactions occurrence

Wynika z tego, że:

$$T_{CAŁK\_TS} = 7*t_{TS}$$
$$T_{CAŁK\_OUT} = 4*t_{OUT}$$
$$T_{CAŁK\_LOOK} = 3*t_{LOOK} \quad (2)$$
$$T_{CAŁK\_BD} = t_{BD}$$

Jeżeli uwzględni się możliwość trafienia na współbieżnie realizowaną transakcję i konieczność ustawienia się w kolejce oraz wynikającej z tego faktu konieczności uaktualnienia danych, to koszt wykonania transakcji wzrasta o następujące składniki:

– koszty kontaktów z przestrzenią krotek – dwukrotny kontakt w celu ustalenia numeru w kolejce, jeden związany ze wstawieniem transakcji do kolejki oraz 4*(NGW – NLW) kontakty przy uaktualnianiu danych,

– koszty wstawiania związane z ustawieniem się w kolejce (zaktualizowany maksymalny numer w kolejce oraz element kolejki) oraz wstawienie historii transakcji po zaktualizowaniu pola *liczba serwerów mających zrealizować transakcję*: (NGW – NLW),
– koszty wyszukiwania krotek – znalezienie krotki z maksymalnym numerem w kolejce oraz 3*(NGW – NLW) operacji wyszukiwania związanych z odczytem krotek historii wersji oraz historii transakcji.

Zatem uzyskujemy:

$$T_{CAŁK\_TS} = (10 + 4*(NGW - NLW))*t_{TS}$$
$$T_{CAŁK\_OUT} = (6 + NGW - NLW)*t_{OUT}$$
$$T_{CAŁK\_LOOK} = (4 + 3*(NGW - NLW))*t_{LOOK} \quad (3)$$
$$T_{CAŁK\_BD} = (1 + NGW - NLW)*t_{BD}$$

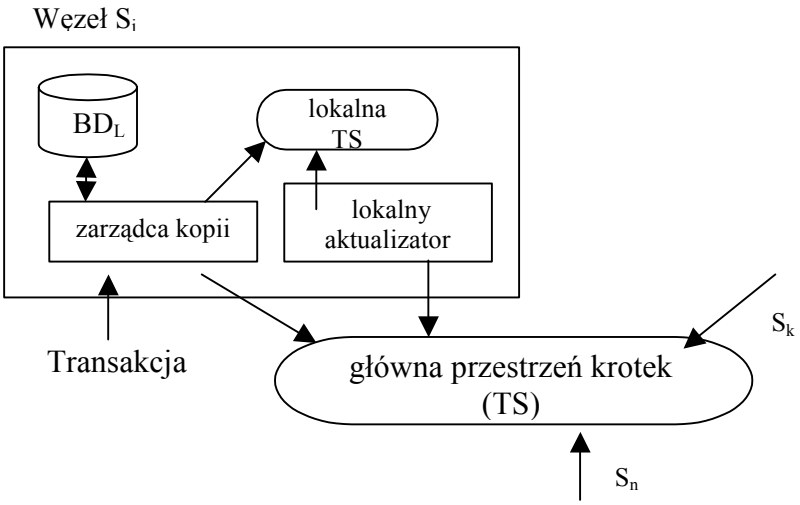
W koszcie tym nie został uwzględniony czas oczekiwania w kolejce, ponieważ jest on uzależniony od czasu trwania współbieżnie biegnących transakcji, a nie cech metody wykorzystywanej do realizacji transakcji rozproszonej. Czas ten nie może przekroczyć wartości granicznej, jaką stanowi *timeout*. Nastąpiłoby bowiem wtedy wycofanie oczekującej transakcji.

5. Optymalizacja rozwiązania

W kolejnym kroku analizy rozpatrzymy czynniki wpływające na poszczególne składniki otrzymanego w poprzednim punkcie kosztu wykonania transakcji (wzór 1). Należy przypuszczać, że czas t_{TS} zdeterminowany jest odległością węzła od wirtualnie współdzielonej pamięci, natomiast na czasy wstawiania i wyszukiwania informacji w przestrzeni krotek (t_{OUT} i t_{LOOK}) może rzutować znajdująca się tam liczba krotek. Ponieważ czas dostępu do lokalnej bazy danych (t_{BD}) nie podlega optymalizacji z punktu widzenia prezentowanego rozwiązania, więc przedmiotem dalszej analizy będą dwa pozostałe elementy.

Odległość węzła od wirtualnie współdzielonej pamięci

Wpływ odległości węzła od wirtualnie współdzielonej pamięci na czas realizacji transakcji może zostać ograniczony przez wykorzystanie lokalnej przestrzeni krotek (rys. 3). Do przestrzeni tej, za pomocą specjalnego modułu lokalnego aktualizatora, przenoszone są zapisy z pamięci głównej, dotyczące historii transakcji i historii wersji danych. W ten sposób kontakt zarządcy z globalną przestrzenią ogranicza się tylko do realizacji funkcji związanych z blokowaniem jednostek danych oraz ze sprawdzeniem ich aktualnego stanu.



Rys. 3. Architektura węzła zawierającego lokalną przestrzeń krotek
Fig. 3. The architecture of node containing local tuple space

Wpływ liczby krotek znajdujących się w przestrzeni krotek na czas realizacji transakcji

Podczas pracy systemu w przestrzeni krotek znajdują się następujące informacje:

- elementy historii transakcji,
- elementy historii wersji,
- krotki globalnych wersji danych,
- ostatni numer w kolejce.

Ponadto okresowo, w zależności od wykorzystania systemu, mogą pojawiać się elementy kolejki.

Operowanie na obiektach wirtualnie współdzielonej pamięci opiera się na dopasowaniu tych obiektów do wzorca. Wzorzec, podobnie jak krotka, zawiera sekwencje pól. Wśród nich mogą być takie, które posiadają wartości i wyrażenia (parametry aktualne) lub nazwy zmiennych, do których kopiowane są wartości z pól dopasowywanej krotki (parametry formalne). Takie pozycje powinny być poprzedzone znakiem „?”. O dopasowaniu krotki do wzorca mówimy, jeżeli spełnione są następujące warunki:

- wzorzec i krotka zawierają tę samą liczbę pól,
- typy, wartości i długości pól z parametrami aktualnymi są takie same jak w badanej krotce,
- typy parametrów formalnych wzorca są zgodne z typami odpowiadających im pól w dopasowywanej krotce.

Reguły dostępu do danych w przestrzeni krotek sugerują, że czas odnalezienia krotki w przestrzeni krotek będzie uzależniony od liczby znajdujących się tam elementów. Poczynione zostały zatem kroki, aby:

a) zadbać o usuwanie z przestrzeni zbędnych informacji tak szybko, jak tylko jest to możliwe,

b) oszacować rozmiar danych znajdujący się w TS, który może spowodować, że realizacja transakcji w systemie eksperymentalnym staje się nieefektywna.

Odpowiedzią na zadanie postawione w punkcie a) było wyposażenie zarządców kopii w mechanizm wykrywania i usuwania niepotrzebnych informacji z wirtualnie współdzielonej pamięci. Uwagę skupiono na krotkach, które są stałym elementem wypełnienia przestrzeni krotek i znajdują się w niej niezależnie od aktualnie biegnących transakcji. Należą do nich krotki historii transakcji, historii wersji, numer globalnej wersji danych oraz krotka z numerem ostatniego elementu w kolejce. Dwa ostatnie rodzaje krotek niosą ze sobą informację, która w wirtualnej pamięci musi być zawsze dostępna, więc oszczędności należało szukać wśród krotek reprezentujących historie transakcji i wersji danych.

Transakcje należące do historii są sukcesywnie wykonywane przez zainteresowane węzły. Istnieje zatem moment, kiedy określona transakcja dociera do wszystkich kopii i dalsze przechowywanie o niej informacji nie jest konieczne. Mechanizmem pozwalającym wykryć taką sytuację jest pole *liczba serwerów mających zrealizować transakcję* krotki historii transakcji. Pole to, w momencie wstawiania transakcji do historii, ma wartość równą liczbie kopii jednostki *x* pomniejszonej o jeden. Każdy z zarządców uaktualniających swoje dane dekrementuje to pole, co w efekcie prowadzi do jego zerowej wartości. Jest to sygnał dla wszystkich zarządców, że krotka może zostać usunięta. Razem z nią usuwane są wszystkie krotki historii wersji, które utworzone zostały po jej wykonaniu. Transakcja ta będzie dostępna jeszcze przez jakiś czas w plikach log zarządców kopii.

Oszacowania, o których mowa jest w punkcie b), związane są z liczba porównań, których należy dokonać, by uzyskać interesującą informację lub by móc stwierdzić, że takiej informacji nie ma. Mają one na celu ustalenie progu zajętości przestrzeni krotek, poza którym system nie pracuje efektywnie. Oszacowań tych dokonano empirycznie.

6. Przeprowadzone eksperymenty

Dla opracowanej metody wykonano szereg eksperymentów, mających na celu określenie wpływu omówionych czynników na czas realizacji transakcji. Przeprowadzone testy potwierdziły spodziewany efekt zwiększenia czasu przesyłu informacji wraz ze wzrostem odległości i zmniejszeniem przepustowości sieci komunikacyjnej. Wpływ tego czasu (w szerokim przedziale badań) na pracę testowanego systemu był jednak znikomy ze względu na rozmiar przesyłanych informacji (pojedyncza krotka mieści się w jednym pakiecie).

Zadaniem kolejnych eksperymentów było określenie wpływu liczby krotek znajdujących się w wirtualnie współdzielonej pamięci na czas wstawiania i poszukiwania pojedynczej

krotki. Testy polegały na wprowadzeniu określonej liczby krotek do przestrzeni krotek, a następnie wstawieniu lub odszukaniu wśród nich zadanego wzorca. Eksperymenty te modelują sytuację, do której dochodzi np. podczas zakładania blokady. Należy wtedy odnaleźć właściwą krotkę z numerem globalnej wersji danych, a następnie wprowadzić ją ze zmienionymi wartościami pól z powrotem do przestrzeni krotek. Testy te wykazały poprawność i efektywność funkcjonowania metody w obecności bardzo dużej liczby krotek w przestrzeni krotek, znacznie większej niż wartości przewidywane w rzeczywistym systemie.

Jednak głównym celem przeprowadzonych badań było określenie udziału czynności związanych z operacjami na przestrzeni krotek w ogólnym czasie realizacji transakcji. Dla uzupełnienia poprzednich badań wykonano eksperymenty, mające na celu wyznaczenie czasu wykonania dowolnych operacji na bazie danych. Po podstawieniu uzyskanych przykładowych rezultatów do wzoru (1) okazało się, że nakłady czasu na operacje organizacyjne wprowadzane przez opracowaną metodę są pomijalne w stosunku do całkowitego czasu przeprowadzenia transakcji.

7. Podsumowanie

Przedstawiona w artykule metoda umożliwia przeprowadzenie transakcji, dotyczących kopii danych z wykorzystaniem tylko jednej z nich, nie dopuszczając równocześnie do powstawania kolizji w realizacji modyfikacji tych samych zestawów rekordów w różnych kopiach danych. Właściwości te uzyskano stosując takie mechanizmy współbieżnego dostępu do danych, jak: blokady, wersjonowanie danych oraz historie przeprowadzonych transakcji. Wpływ na sposób realizacji tych mechanizmów miało wykorzystanie współdzielonej pamięci zarówno do komunikacji pomiędzy poszczególnymi elementami systemu, jak i do przechowywania niezbędnych informacji.

Analiza efektywności metody wraz z przeprowadzonymi eksperymentami wykazała, że efektywność realizacji transakcji zależy głównie od czasu operacji na bazie danych, a nakłady czasowe, wynikające z kontaktów z przestrzenią krotek, nie wprowadzają znaczących opóźnień.

LITERATURA

1. Acosta–Elias J., Navarro–Moldes L.: A Demand based Algorithm for Rapid Updating of Replicas. 22nd International Conference on Distributed Computing Systems. Vienna, Austria, 2002.
2. Agrawal D., Abbadi A. El. Steinke C.: Epidemic Algorithms in Replicated databases. Proceedings of the Symposium on Principles of the 16^{Th} ACM SIGACT–SIGMOD–SIGART Database Systems, Tuscon, Arizona 1997.
3. Ceri S., Pelagatti G.: Distributed Databases. Principles and Systems. McGraw–Hill Book Company, 1984.
4. Ceri S., Houtsma M.A.W., Keller A.M., Samarati P.: A Classification of Update Methods for Replicated Databases. Computer Science Technical Report STAN-CS-91-1392, October 1991.
5. Chłopek M., Harężlak K., Josiński H.: Implementacja podstawowych mechanizmów zarządzania rozproszoną bazą danych w eksperymentalnym systemie wykorzystującym model współdzielonej pamięci. IV Seminarium "Sieci Komputerowe". Zeszyty Naukowe Pol. Śl. s. Informatyka, z.32, Gliwice 1997.
6. Chłopek M., Harężlak K., Josiński H.: Sterowanie współbieżnym dostępem do danych podczas realizacji transakcji rozproszonych – rozwój eksperymentalnego systemu zarządzania rozproszoną bazą danych. V Seminarium "Sieci Komputerowe". Zeszyty Naukowe Pol. Śl., s. Informatyka, z. 34, Gliwice 1998.
7. Elmasri R. Navathe S. B.: Fundamentals of Database Systems. Addison–Wesley 2000.
8. Gracia–Molina H., Ullman J. D., Widom J.: Implementacja systemów baz danych. Wydawnictwa Naukowo-Techniczne, Warszawa 2003.
9. Harężlak K.: Transaction Management on Replicated Data. EDBT Konstanz, Niemcy, 2000.
10. Harężlak K.: Realizacja transakcji na kopiach danych w środowisku współdzielonej pamięci. VIII Seminarium "Sieci Komputerowe". Studia Informatica vol. 22, Krynica 2001.
11. Praca zbiorowa pod redakcją S. Kozielskiego: Systemy umożliwiające realizację algorytmów równoległych w sieciach komputerowych. Skrypt Politechniki Śl. nr 1975, Gliwice 1996.
12. Praca zbiorowa pod redakcją K. Zielińskiego: Środowiska programowania rozproszonego w sieciach komputerowych. Wydawnictwo Księgarnia Akademicka, Kraków 1994.
13. Staniszkis W.: Projektowanie rozproszonych systemów informatycznych. Informatyka nr 1, 1995.

14. Virtual Shared Memory and the Paradise system for Distributed Computing – A technical White Paper. SCIENTIFIC Computing Associates, Inc. Via Internet. http://www. lindaspaces. com/downloads/vsm.pdf, 1999.

Recenzent: Dr hab. inż. Stanisław Wołek Prof. Pol. Rzeszowskiej

Wpłynęło do Redakcji 5 lipca 2004 r.

Abstract

The paper presents the new, asynchronous update method of replicated data in virtual shared memory environment. The method advantage is possibility of utilizing only one data copy for distributed transaction realization contemporarily guarding against replicated data update conflicts. This is possible owning to usage versioning replicated data. The virtual shred memory was used as a store to keep tuple with global version number and tuples with transaction and version histories. These histories used for synchronizing all of data copies let the data copy manager to bring its copy up to date without necessary to communicate with any other manager.

The problem of distributed database transaction performing efficiency was also discussed. The time cost model for distributed transaction realization was proposed. Two elements were take into consideration: the distance between client and virtual shared memory and the number of tuples in virtual shared memory.

Adres

Katarzyna HARĘŻLAK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, kharezlak@zti.ps.edu.pl.