ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 32 Nr kol. 1356

Henryk KRAWCZYK, Sławomir BOKINIEC, Marcin TATARATA
Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

INTEGRACJA HETEROGENICZNYCH BAZ DANYCH W ŚRODOWISKU SIECIOWYM PVM

Streszczenie. W pracy przeanalizowano budowę uniwersalnego interfejsu łączącego heterogeniczne bazy danych w środowisku rozproszonym. Zaprezentowano trójpoziomową architekturę logiczną niezależną od fizycznej platformy. W oparciu o model logiczny zrealizowano prototyp systemu łączącego dwa klasyczne SZBD: Postgres i Informix, pracujące pod kontrolą systemów operacyjnych: MS Windows95, Linux i SunOs. Zarządzanie rozproszonymi procesami i ich komunikacja odbywały się w środowisku PVM/WPVM. Przedstawiona została również ocena efektywności zaimplementowanych algorytmów równoległego sortowania i łączenia tablic.

INTEGRATION OF HETEROGENEOUS DATABASES IN PVM ENVIRONMENT

Summary. This paper presents the structure of an universal interface between different databases in distributed environment. We present the logical architecture that is independent from the physical platform. The logical model is the base for the prototype of a system integrating Informix and Postgres databases, working under MS Windows 95, Linux and SunOs operating systems. The distributed processing is supported by PVM/WPVM. Data distribution was exploited for parallel execution of two algorithms: sorting and joining tables.

1. Wprowadzenie

Teoria rozproszonych baz danych jest stosunkowo młodą gałęzią informatyki. Świadczy o tym mała ilość powszechnie używanych systemów, ilość prowadzonych prac badawczych, rozbieżne klasyfikacje oraz ciągle nie rozwiązane problemy. Rozproszone bazy danych łączą w sobie wszystkie problemy popularnych scentralizowanych systemów zarządzania bazą

danych (SZBD) z problemami systemów rozproszonych. Część z nich stawia sobie dodatkowo zadanie transformacji różnych modeli danych i języków zapytań pomiędzy rozproszonymi bazami lokalnymi. Definicje rozproszonej bazy danych oraz rozproszonego systemu zarządzającego bazą danych są proste i klarowne ([6]):

***Rozproszona Baza Danych** (RBD) jest zbiorem wielu logicznie powiązanych baz danych rozproszonych pomiędzy węzły sieci komputerowej.*

***Rozproszony System Zarządzania Bazą Danych** (RSZBD) to oprogramowanie, które wspomaga zarządzanie RBD i sprawia, że jej rozproszenie jest niewidoczne dla użytkownika.*

Częste występowanie logicznie powiązanych rozproszonych baz danych jest następstwem rozmieszczenia w pewnej odległości poszczególnych działów tej samej organizacji. Dane są przechowywane i pielęgnowane w miejscu, w którym powstają. Obecność lokalnych sieci komputerowych w wielu przedsiębiorstwach, jak również fakt podłączania ich do sieci miejskich, czy też metropolitalnych umożliwia każde fizyczne połączenie między komputerami, na których znajdują się logicznie powiązane bazy danych. Komputery te są często zupełnie z sobą niekompatybilne. W parze z różnorodnością sprzętową idą różne systemy operacyjne i systemy baz danych z innymi modelami danych, odmiennymi językami zapytań do baz danych, różnymi się metodami dostępu do danych, itd. RSZBD powinien obsługiwać te wszystkie rodzaje heterogeniczności w systemach RBD.

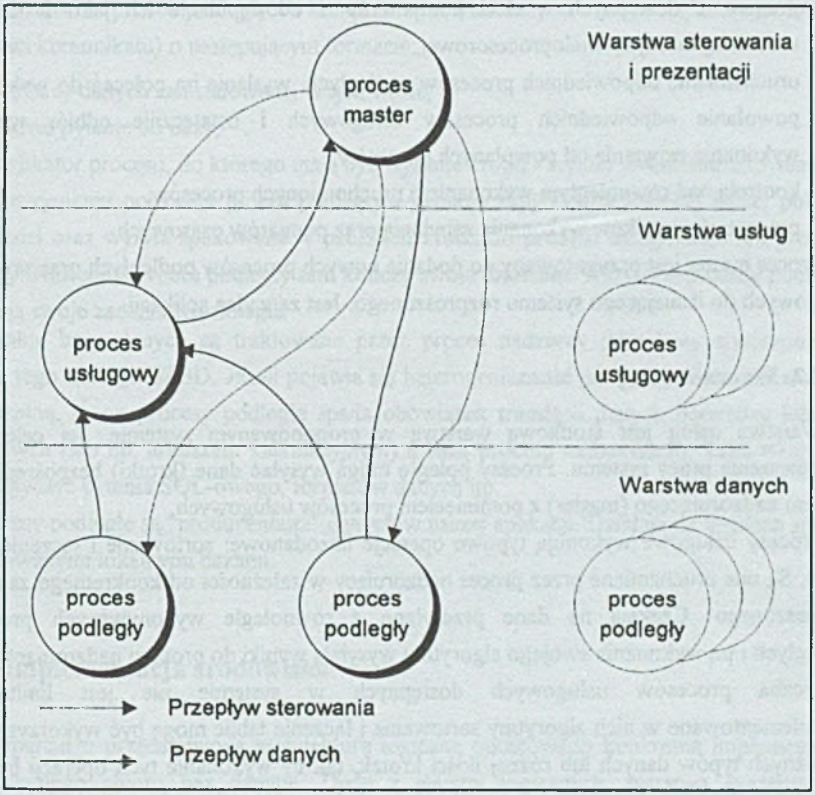
Przy konstruowaniu *od fundamentów* RSZBD, istotnym problemem jest dobór środowiska komunikacji pomiędzy poszczególnymi komputerami pracującymi w sieci. Wybór takiego środowiska powinien zapewnić: prostotę obsługi, dostępność dla kilku systemów operacyjnych i wielu platform sprzętowych. Poza tym powinno ono kompensować różnice w sprzęcie i oprogramowaniu systemowym porzez automatyczną translację różnych formatów danych, protokołów itp., a również umożliwiać dynamiczne zarządzanie rozproszonymi zasobami z jednego wybranego węzła.

Wszystkie wyżej wymienione kryteria spełnia PVM [4]. PVM (Parallel Virtual Machine) jest prosty w obsłudze i, co jest istotne, funkcjonuje na komputerach z systemem operacyjnym typu Unix oraz pod systemami MS Windows. Dołączenie bibliotek PVM-a do aplikacji powoduje, że różnorodne komputery połączone w sieci traktowane są jako jedna wirtualna maszyna wieloprocesorowa. Rozproszenie i heterogeniczność systemu są więc ukryte przed programistą, który posługuje się jedynie operacjami zarządzania zbiorem zadań wykonywanym na procesorach tej maszyny i operacjami komunikacji między nimi na odpowiednio wysokim poziomie abstrakcji.

Celem pracy jest połączenie kilku rozproszonych baz danych za pomocą PVM-a. Ma to zapewnić przeglądanie ich zawartości z jednego z węzłów, wykorzystanie rozproszenia danych, a także równoległe wykonywania operacji bazodanowych. GUI utworzonego RSZBD umożliwia nie tylko sformułowanie zapytania do rozproszonej bazy danych, ale również

pomiar efektywności wykonania rozproszonych algorytmów sortowania i łączenia tablic. Proponowany prototyp jest przygotowany do dodawania heterogenicznych, lokalnych baz danych i wykorzystywania nowych algorytmów przetwarzania równoległego.

2. Koncepcja logicznego interfejsu do bazy danych



Rys. 1. Trójwarstwowa architektura logiczna systemu
Fig. 1. Layers in the logical architecture

Na proponowaną architekturę logiczną składają się trzy warstwy. Procesy w każdej warstwie wykonują zadane funkcje w systemie oraz mają określone interfejsy z innymi warstwami. Rys. 1 przedstawia zestaw czterech podstawowych procesów systemu, ich przydział do warstw i wzajemne powiązania. W dalszej części rozdziału opisano poszczególne warstwy.

2.1. Warstwa sterowania i prezentacji

Jest to najważniejsza warstwa w systemie. W niej znajduje się proces master (proces nadzorujący), którego zadaniem jest:

- przechowywanie informacji o zasobach systemu i uaktualnianie jej na życzenie użytkownika,
- pobieranie zapytań do rozproszonej bazy danych od użytkownika,
- dołączanie dostępnych w sieci komputerów i udostępnianie ich jako procesorów wirtualnej maszyny wieloprocesorowej,
- uruchamianie odpowiednich procesów podległych, wysłanie im poleceń do wykonania, powołanie odpowiednich procesów usługowych i ostatecznie odbiór wyników wykonania zapytania od powołanych procesów,
- kontrola nad równoległym wykonaniem uruchomionych procesów,
- prezentacja wyników wykonania zapytania oraz pomiarów czasowych.

Proces master jest przygotowany do dodania nowych procesów podległych oraz procesów usługowych do istniejącego systemu rozproszonego. Jest zarządcą aplikacji.

2.2. Warstwa usług

Warstwa usług jest środkową warstwą w proponowanym systemie. Jej celem jest przyspieszenia pracy systemu. Procesy poległe mogą wysyłać dane (krotki) bezpośrednio do procesu nadzorującego (master) z pominięciem procesów usługowych.

Procesy usługowe wykonują typowe operacje bazodanowe: sortowanie i łączenie tablic (*join*). Są one uruchamiane przez proces nadzorujący w zależności od konkretnego zapytania rozproszonego. Czekają na dane przesyłane z równolegle wykonywanych procesów podległych i po wykonaniu swojego algorytmu wysyłają wyniki do procesu nadzorującego.

Liczba procesów usługowych dostępnych w systemie nie jest limitowana. Zaimplementowane w nich algorytmy sortowania i łączenia tablic mogą być wykorzystywane do różnych typów danych lub różnej ilości krotek, tak by wykonanie tych operacji było jak najszybsze. Proces usługowy może się składać z jednego lub wielu równolegle działających procesów. Master decyduje, jaki algorytm jest najbardziej odpowiedni dla konkretnej operacji.

Z poszczególnymi algorytmami związane są różne parametry wejściowe. Przykładowo, dla najprostszego sortowania przekazywane są trzy liczby: dwie pierwsze określają identyfikatory procesów podległych, które prześlą krotki do procesu, trzecia - numer kolumny, według której ma nastąpić sortowanie.

Proces usługowy może być wykonywany na dowolnym komputerze dołączonym do wirtualnej maszyny wieloprocesorowej. Master uruchamiając procesy usługowe na odpowiednim procesorze może przyspieszyć wykonanie algorytmu.

2.3. Warstwa danych

Jest to najniższa warstwa w proponowanej architekturze. Składa się z dowolnej liczby procesów podległych. Są one umieszczone w węzłach sieci, w których znajdują się lokalne bazy danych z danymi, które chcemy przeglądać w procesie nadzorującym (master). Ich aktualne rozmieszczenie jest zapamiętywane w bazie danych procesu nadzorującego (słowniku globalnym). Jeśli w zapytaniu rozproszonym znajduje się pytanie do lokalnej bazy, proces master uruchamia proces podległy na komputerze z tymi danymi oraz przesyła mu polecenie (w postaci komunikatu) o następującym formacie:

- nazwa bazy danych zainstalowanej w tym węźle,
- SQL-owe pytanie do bazy,
- identyfikator procesu, do którego mają być wysłane krotki - wyniki wykonania zapytania.

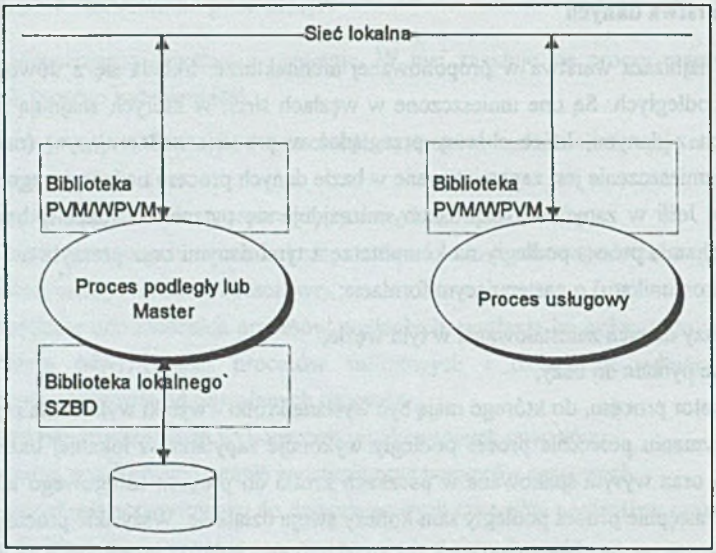
Po otrzymaniu polecenia proces podległy wykonuje zapytanie w lokalnej bazie, pobiera odpowiedzi oraz wysyła spakowane w paczkach krotki do procesu usługowego lub procesu nadzorcy. Następnie proces podległy sam kończy swoje działanie. Wszystkie procesy podległe wykonują swoje zadania równoległe.

Lokalne bazy danych są traktowane przez proces nadzorcy jako bazy stworzone za pomocą tego samego SZBD. Jeżeli pojawia się heterogeniczność pomiędzy bazą nadzorcy a bazą lokalną, to na procesy podległe spada obowiązek translacji danych pomiędzy lokalną bazą danych (lub np. arkuszem kalkulacyjnym) a bazą procesu nadzorczego. Taka translacja może dotyczyć pytania SQL-owego, formatów danych itp.

Procesy podległe są "producentami" danych w naszej aplikacji. Działają na węzłach sieci z zainstalowanymi lokalnymi bazami.

3. Implementacja środowiska

W oparciu o przedstawioną architekturę logiczną opracowano konkretną implementację dla dostępnego zbioru baz danych. Dwie z warstw logicznych, warstwa prezentacji i sterowania oraz warstwa danych, wymagają dostępu do zewnętrznych źródeł danych. Proces nadzorca musi więc mieć dostęp do globalnego słownika oraz repozytorium informacji statystycznych, procesy podrzędne danych zaś z definicji korzystają z zewnętrznych baz danych. Warstwa usług przetwarza dane dostarczone przez warstwę niższą, dlatego nie wymaga współpracy z SZBD. Wszystkie procesy muszą posiadać wspólną platformę komunikacji sieciowej, którą w tym przypadku jest środowisko PVM/WPVM [1]. W konsekwencji wyłoniły się dwie zasadnicze implementacje procesów w tworzonym prototypie. Rys. 2 przedstawia koncepcję tych dwu szablonów implementacyjnych.



Rys. 2. Koncepcja implementacji
Fig.2. The implementation idea

Prototyp systemu [3] zrealizowany został w oparciu o systemy komputerowe przedstawione w tabeli 1.

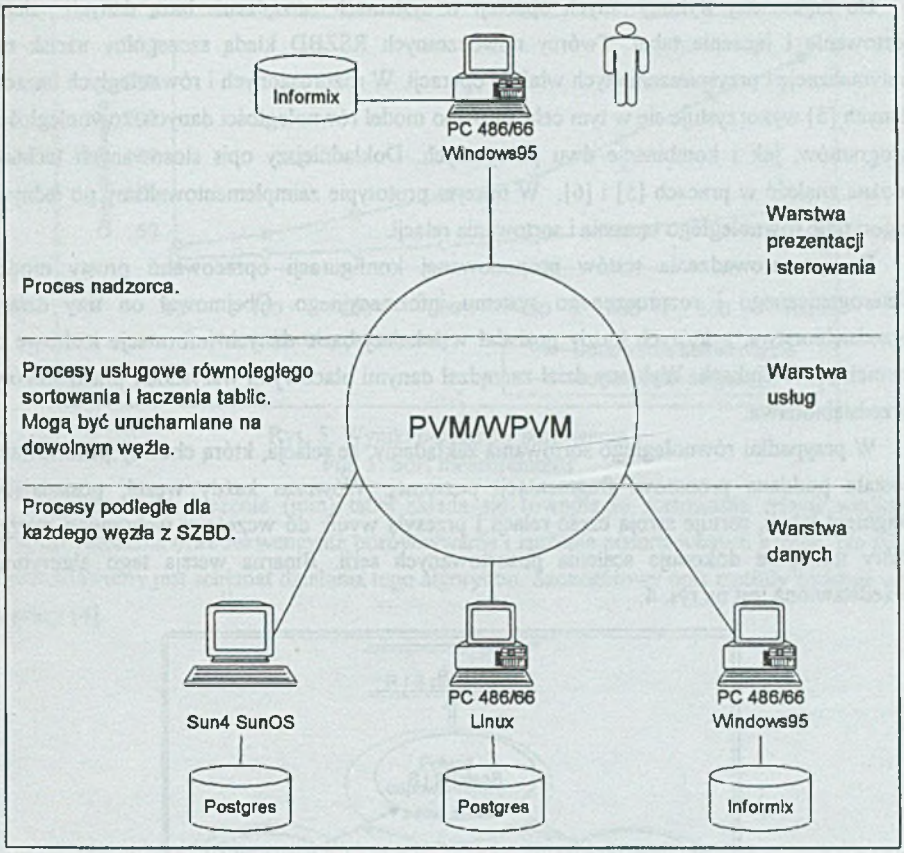
Tabela 1

Środowisko prototypu - platforma sprzętowa i programowa

Komputer	System operacyjny	SZBD	Środowisko komunikacji sieciowej
PC 486/66	MS Windows95	Informix	WPVM
PC 486/66	Linux	Postgres	PVM
Sun4	SunOS	Postgres	PVM

Proces master został zaimplementowany w środowisku MS Windows 95 przy użyciu kompilatora MSVC++ oraz bibliotek systemu Informix ver. 6 oraz WPVM ver. 2.0. Procesy należące do warstwy usług - realizujące równoległe łączenie oraz sortowanie relacji - zostały zrealizowane w każdym węźle maszyny wirtualnej, tak by była możliwa dynamiczna konfiguracja maszyny wirtualnej w zależności od obciążeń poszczególnych węzłów.

W każdym węźle umieszczone zostały również procesy podległe odwołujące się do lokalnych SZBD. Rys. 3 przedstawia fizyczną realizację przyjętej koncepcji.



Rys. 3. Architektura fizyczna systemu
Fig. 3. The physical architecture

Na rysunku przedstawiono elementy fizyczne prototypu na tle modelu logicznego. Komputery były włączone do sieci lokalnej Ethernet.

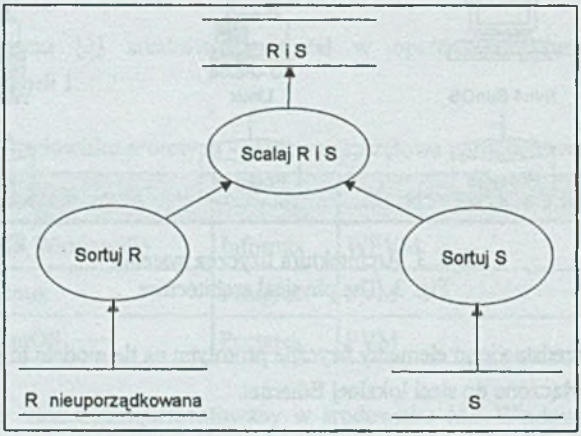
4. Eksperymenty testowe

Problem optymalizacji zapytań rozproszonych w RSZBD nie należy do łatwych [5]. Jednym z jego aspektów jest problem równoległości w realizacji zapytań do baz danych. W ogólności możemy mówić o paralelizmie wewnątrz zapytania rozproszonego (*interquery parallelism*) i paralelizmie pomiędzy zapytaniami (*intraquery parallelism*).

Do najczęściej wykonywanych operacji w systemach zarządzania bazą danych należą sortowanie i łączenie tabel. Twórcy nowoczesnych RSZBD kładą szczególny nacisk na optymalizację i przyspieszenie tych właśnie operacji. W rozproszonych i równoległych bazach danych [5] wykorzystuje się w tym celu zarówno model równoległości danych, równoległości programów, jak i kombinacje dwu powyższych. Dokładniejszy opis stosowanych technik można znaleźć w pracach [5] i [6]. W naszym prototypie zaimplementowaliśmy po jednym algorytmie równoległego łączenia i sortowania relacji.

Do przeprowadzenia testów proponowanej konfiguracji opracowano prosty model heterogenicznego i rozproszonego systemu informacyjnego. Obejmował on trzy działy przedsiębiorstwa, z których każdy posiadał w lokalnej bazie danych informacje kadrowe o swoich pracownikach. Wybrany dział zarządzał danymi płacowymi wszystkich pracowników przedsiębiorstwa.

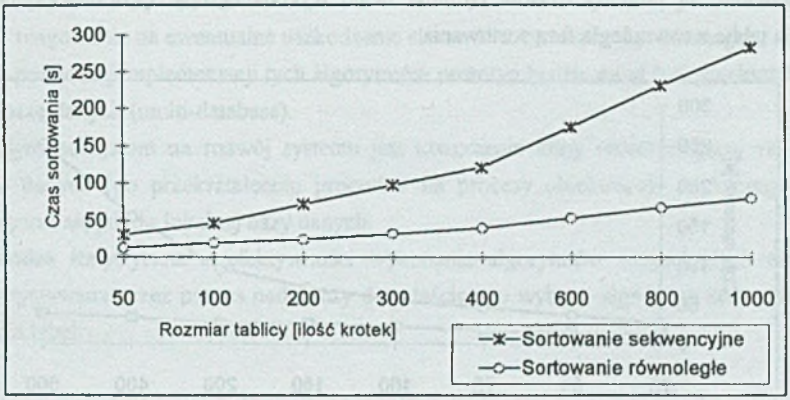
W przypadku równoległego sortowania zakładamy, że relacja, którą chcemy posortować, została poddana procesowi fragmentacji poziomej. Wówczas każdy węzeł, posiadający fragment tabeli, sortuje swoją część relacji i przesyła wynik do wcześniej wybranego węzła, który następnie dokonuje scalenia posortowanych serii. Binarna wersja tego algorytmu przedstawiona jest na rys. 4.



Rys. 4. Równoległe sortowanie
Fig. 4. Parallel sort

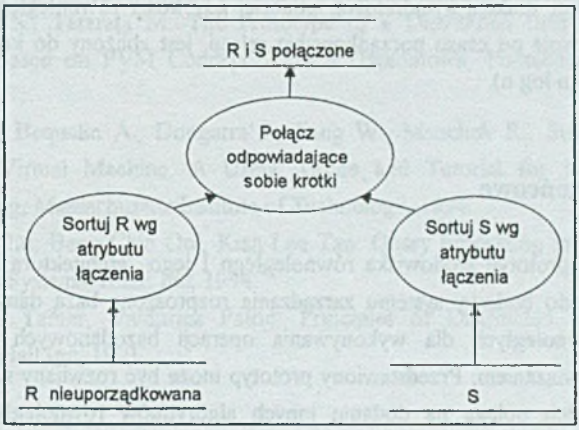
Powyższy algorytm porównany został z metodą sekwencyjną. W metodzie sekwencyjnej obie części relacji były najpierw pobierane z węzłów, umieszczane w jednej tablicy, a następnie sortowane za pomocą języka zapytań lokalnego SZBD.

Zgodnie z przewidywaniami uzyskaliśmy około dwukrotne przyspieszenie w sortowaniu relacji w stosunku do operacji sekwencyjnie porządkującej tę samą tabelę. Rys. 5 pokazuje uzyskane wyniki dla obu metod.



Rys. 5. Wyniki pomiarów sortowania
Fig. 5. Sort measurements

Na równoległe łączenie (join) tabel składa się równoległe sortowanie relacji według atrybutu złączenia oraz sekwencyjne porównywanie i łączenie posortowanych krotek. Na rys. 6 przedstawiony jest schemat działania tego algorytmu. Szczegółowy opis metody znajduje się w pracy [4].

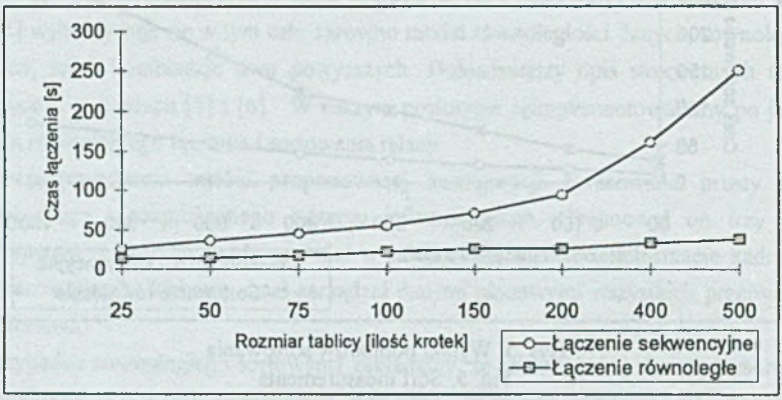


Rys. 6. Łączenie tablic z równoległą fazą sortowania
Fig. 6. Join with parallel sort phase

Przedstawiona metoda została porównana z algorytmem sekwencyjnym. Zakładała ona, że proces master pobiera kolejno relacje do złączenia od wybranych hostów i wykorzystuje lokalny SZBD do ich połączenia.

W przypadku równoległego łączenia tabel wyniki są o tyle interesujące, że zauważamy ponad dwukrotny wzrost wydajności. Może to być związane z niskim stopniem optymalizacji procesu łączenia tabel w stosunku do operacji sortowania w badanych SZBD. Na rysunku 7

przedstawione są pomiary czasów łączenia tablic wykonanego w sposób sekwencyjny i łączenia tablic z równoległą fazą sortowania.



Rys. 7. Wyniki pomiarów łączenia tablic
Fig. 7. Join measurements

Już z tego prostego wykresu można zauważyć, że złożoność obliczeniowa łączenia sekwencyjnego jest funkcją kwadratową, podczas gdy koszt równoległego łączenia tablic, który zależy głównie od czasu porządkowania relacji, jest zbliżony do kosztu optymalnego sortowania - O (n log n).

5. Uwagi końcowe

Opracowany prototyp środowiska równoległego I jego architektura logiczna stanowią dobrą podstawę do budowy systemu zarządzania rozproszoną bazą danych. Zastosowanie algorytmów równoległych dla wykonywania operacji bazodanowych okazało się być efektywnym rozwiązaniem. Przedstawiony prototyp może być rozwijany na kilka sposobów. Najprostszy z nich polega na dodaniu innych algorytmów równoległych jako nowych procesów usługowych lub też nowych procesów podległych, współpracujących z nowymi SZBD.

Tworząc prototyp skupiono się na mechanizmach komunikacji, zarządzania rozproszonymi zasobami i pomiarami efektywności architektury. Dlatego też rozpatrzono jedynie możliwość czytania z bazy. Konieczne jest również umożliwienie modyfikacji lokalnych baz danych (zapis, kasowanie). W konsekwencji następujące algorytmy powinny być opracowane i zaimplementowane:

- kontrolowanie wykonania równoległych transakcji (zapobieganie zakleszczeniom),

- optymalizacja zapytań,
- reagowanie na ewentualne uszkodzenie elementów systemu rozproszonego.

Po pomyślnej implementacji tych algorytmów prototyp będzie mógł być przekształcony w multibazę danych (multi-database).

Innym pomysłem na rozwój systemu jest utworzenie klasy reprezentującej zewnętrzne źródło danych (po przekształceniu procesów na procesy obiektowe), do którego można odwoływać się jak do lokalnej bazy danych.

Wiedza statystyczna o efektywności wykonania algorytmów równoległych może być wykorzystywana przez proces nadzorczy do właściwego wyboru algorytmu sortowania i/lub łączenia tabel.

LITERATURA

[1] Alves A., Silva L., Carreira J., Gabriel Silva J.: WPVM: Parallel Computing for the People.

[2] Bobak Angelo R.: Distributed and Multi-database Systems, ARTECH HOUSE Inc, 1996

[3] Bokiniec S., Tatarata M: The Prototype of a Distributed Information Processing System Based on PVM Concept - Praca Dyplomowa, Politechnika Gdańska, ETI 1996.

[4] Geist A., Bequelin A., Dongarra J., Jiang W., Manchek R., Sunderam V.: PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Paralell Computing, Massachusetts Institute of Technology, 1994.

[5] Hongjun Lu, Beng-Chin Ooi, Kian-Lee Tan: Query processing in Parallel Relational Database Systems, IEEE Inc, 1994.

[6] Ozsu M. Tamer, Valduriez Patric: Principles of Distributed Database Systems, Prentice-Hall Inc, 1991.

Recenzent: Dr hab.inż. Stanisław Kozielski
Prof. Politechniki Śląskiej

Wpłynęło do Redakcji 25 listopada 1996 r.

Abstract

In this paper we present the logical architecture of the system that integrates distributed heterogeneous databases and a brief description of the prototype implementing this architecture. We assume that local databases are already installed on network nodes and the problem is to enable access to them all from the application installed on one of the nodes. GUI of our application hides distribution and heterogeneity from user. Our architecture is independent from physical platform, the system (prototype) is flexible for extension.

PVM and WPVM with their process control and message passing are used for communication between nodes in our prototype. It also supports heterogeneity and it proves to be good choice.

Interfaces between the layers are defined and it makes creating new slaves easier. The interface between master and data slave: master sends to data slave (after spawning it) information that consists of:

- local database name,
- SQL query,
- identifier of process that should receive results of query execution (tuples).

Master process was created on PC with Windows 95 and in Informix NE; three data slaves work on PC with Linux and Postgres database, on PC with Windows 95 and Informix database, on Sun4 with SunOS with Postgres (fig. 3). Service slaves were compiled for all available architecture so they can run on the most suitable processor.

Fig. 5 and fig. 7 show measurements of the parallel algorithms' performance with comparison to serial execution. Parallel execution of database operations sort and join speeded twice their execution. The prototype and the architecture are good bases for developing our prototype into real distributed application - an efficient multi-database.