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

Krzysztof ZIÓŁKOWSKI
Politechnika Śląska, Instytut Informatyki

SIECIOWE METODY DOSTĘPU DO SERWERA BAZ DANYCH MICROSOFT SQL SERVER

Streszczenie. Artykuł opisuje architekturę i cechy bibliotek komunikacyjnych Microsoft SQL Server-a, ze szczególnym uwzględnieniem tych, które są dostępne w środowisku Windows, oraz metodę i wyniki ich porównania pod względem wydajności.

NETWORK ACCESS METHODS TO DATABASE SERVER MICROSOFT SQL SERVER

Summary. This article describes architecture and features of Microsoft SQL Server network libraries, especially this which are available in Windows environment, and discus methods and results of it performance comparison.

1. Wstęp

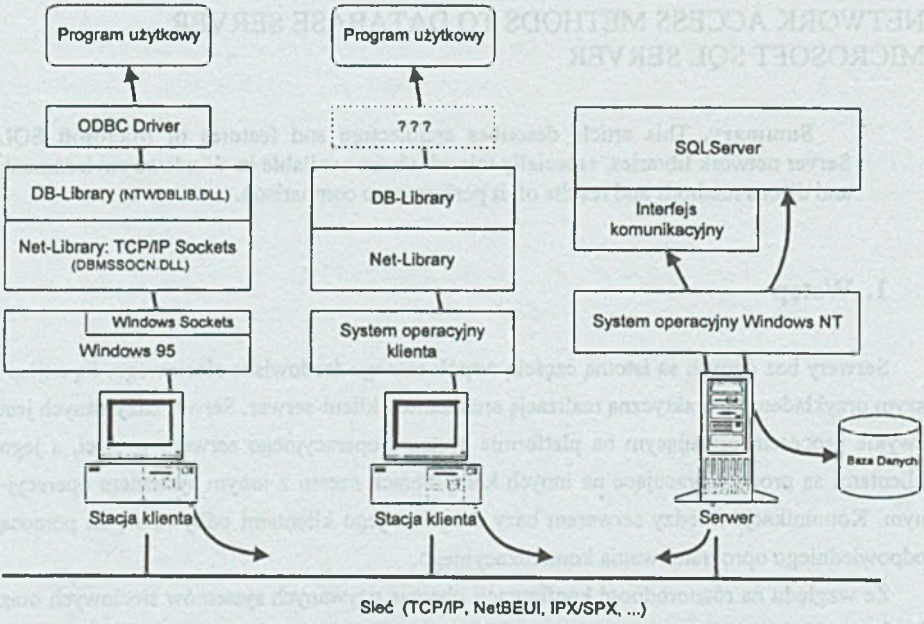
Serwery baz danych są istotną częścią współczesnego środowiska sieciowego. Są najlepszym przykładem na praktyczną realizację architektury klient-serwer. Serwer bazy danych jest zwykle procesem pracującym na platformie systemu operacyjnego serwera w sieci, a jego klientami są procesy pracujące na innych komputerach często z innym systemem operacyjnym. Komunikacja między serwerem bazy danych a jego klientami odbywa się za pomocą odpowiedniego oprogramowania komunikacyjnego.

Ze względu na różnorodność konfiguracji obecnie używanych systemów sieciowych oraz konieczność dopasowywania się do potrzeb użytkowników oprogramowanie służące do komunikacji między serwerem i klientem samo w sobie jest dosyć skomplikowane, a wybór odpowiedniego do konkretnych wymagań systemu może stanowić problem. Niniejsze opraco-

wanie stanowi próbę prezentacji metod oceny interfejsów sieciowych dostępnych dla Microsoft SQL Server 6.0.

2. Architektura oprogramowania Microsoft SQL Server

Microsoft SQL Server pracuje w środowisku systemu operacyjnego Windows NT i jest bardzo dobrze zintegrowany z tym środowiskiem. Oprogramowanie klienckie może pracować w jednym z następujących środowisk operacyjnych: Windows NT, Windows 95, Windows 3.x, MS-DOS. Interfejs komunikacyjny jest całkowicie przeźroczysty dla programu użytkowego współpracującego z SQL Serwerem. Program użytkowy komunikuje się z bazą danych za pomocą zestawu funkcji DB-Library. Funkcje wchodzące w skład DB-Library definiują interfejs programisty do SQL Serwera i są jedyną metodą, za pomocą której oprogramowanie może się z nim komunikować; wszystkie inne iterfejsy programisty *(ang. Application Programers Iterface)* - w tym ODBC - używają DB-Library. Strukturę oprogramowania klienta przedstawia rysunek 1.



Rys. 1. Architektura oprogramowania Microsoft SQL Server
Fig. 1. Microsoft SQL Server software architecture

Komunikacja poprzez sieć może się odbywać za pomocą interfejsów wymienionych w tabeli 1. Konfiguracja środowiska klienta odbywa się za pomocą specjalnego interakcyjnego programu. W przypadku Windows 95 i NT parametry konfiguracyjne są zapisywane w *registry*, w przypadku Windows 16-bitowych zbiorze WIN.INI.

Tabela 1

Interfejsy komunikacyjne klienta w różnych systemach operacyjnych

Net-Library	Windows NT lub Windows 95	Windows (16-bitowe)	MS-DOS
Named Pipes	DBNMPNTW.DLL	DBNMP3.DLL	DBNMPIPE.EXE
NWLink IPX/SPX	DBMSSPXN.DLL	DBMSSPX3.DLL	DBMSSPX.EXE
Banyan VINES	DBMSVINN.DLL	DBMSVIN3.DLL	DBMSVINE.EXE
TCP/IP Sockets	DBMSSOCN.DLL	DBMSSOC3.DLL	brak
Multi-Protocol	DBMSRPCN.DLL	DBMSRPC3.DLL	brak

3. Sieciowe interfejsy Microsoft SQL Server

Wszystkie biblioteki komunikacyjne SQL Server-a są oparte na jakimś dobrze zdefiniowanym interfejsie do tworzenia oprogramowania sieciowego. Wynikiem tego jest wyraźna warstwowa struktura oprogramowania i łatwa konfiguracja. Możliwe też jest stosowanie oprogramowania komunikacyjnego innych producentów. Jeden klient może korzystać z wielu różnych bibliotek przy odwoływaniu się do różnych serwerów. Poniżej przedstawiono w skrócie cechy wybranych bibliotek komunikacyjnych.

3.1. Interfejs Named Pipes

Interfejs nazwanych potoków *(ang. Named Pipes)* wywodzi się z LAN Manager-a i OS/2, aczkolwiek sam mechanizm potoku jest znany również w innych systemach operacyjnych (np. UNIX). Interfejs Named Pipes jest proponowany przez program instalacyjny SQL Server-a jako domyślny.

Jest to sposób komunikacji między procesami *(ang. Iterproces Communication)* polegający na emulowaniu zbioru dyskowego o specjalnych własnościach. Nazwany potok może być jedno- lub dwukierunkowy i łączyć proces serwera i jeden lub więcej procesów klientów. Komunikacja może odbywać się w obrębie jednego komputera lub poprzez sieć. Proces serwera tworzy jedno lub więcej wcieleń potoku za pomocą funkcji CreateNamePipe i nadaje mu unikalną nazwę. Wszystkie wcielenia potoku współdzielą jego nazwę, ale każde posiada własne bufory i służy do komunikacji z jednym klientem. Klienci mogą dołączyć się do tak utworzonego potoku poprzez jego nazwę za pomocą funkcji CreateFile lub CallNamedPipe.

Prawa dostępu do potoku są definiowane jak do każdego innego obiektu w systemie. Procesy serwera i klienta zapisują i odczytują dane poprzez potok, tak jakby był to zbiór na dysku, za pomocą funkcji ReadFile i WriteFile, kilka innych funkcji odpowiada za operacje związane z badaniem stanu potoku i ustawianiem jego parametrów [2].

Nazwa potoku ma format:

a) `\\nazwa_serwera\pipe\nazwa_potoku` - w przypadku klienta;

b) `\\.\pipe\nazwa_potoku` - w przypadku serwera.

Domyślnie SQL Server używa potoku \\.\pipe\sql\query, ale nazwa ta może być zmieniona w czasie konfiguracji [1].

3.2. Interfejs TCP/IP Sockets

Interfejs ten jest realizacją znanego z systemów UNIX sposobu komunikacji miedzy procesami - BSD Sockets, który powstał na Uniwersytecie Kalifornijskim w Berkeley. Jest to interfejs mocno związany z protokołem TCP/IP (aczkolwiek istnieją jego realizacje oparte na innych protokołach - np. IPX/SPX, Apple Talk). W wersji dla środowiska Windows jego interfejs programisty *(ang. Application Programmers Interface)* opisuje specyfikacja Microsoft WinSock.

Standard Berkeley Software Distribution Interprocess Communication (BSD IPC), a co za tym idzie, również Microsoft Winsock, pozwalają na tworzenie aplikacji wymieniających dane pomiędzy swoimi procesami. Co istotne, nie jest określone, czy procesy te muszą być uruchomione na tej samej maszynie, czy też komunikacja zachodzi przy wykorzystaniu sieci komputerowej. Standard pozwala na komunikowanie się procesów bez ingerowania w to, w jaki sposób komunikacja pomiędzy nimi jest rzeczywiście realizowana. Cały mechanizm wywoływania wielu funkcji usługowych niższych warstw sieciowych ukryty jest pod pojęciem **gniazda komunikacyjnego** (ang. *socket*).

Gniazdo oraz **deskryptor gniazda** (ang. *socket decriptor*) i **przyporządkowanie adresu** (ang. *binding*) to podstawowe pojęcia pozwalające na zrozumienie zasady działania modelu BSD IPC.

a) Gniazdo - jest to punkt końcowy procesu komunikacji. Para gniazd połączonych ze sobą stanowi kanał komunikacyjny podobny do potoku.

b) Deskryptor gniazda - stanowi odpowiednik deskryptora pliku i jest używany jako argument funkcji używanych w modelu BSD IPC.

c) Przyporządkowanie adresu - jest konieczne, by gniazdo mogło być udostępnione innym procesom poprzez sieć komputerową.

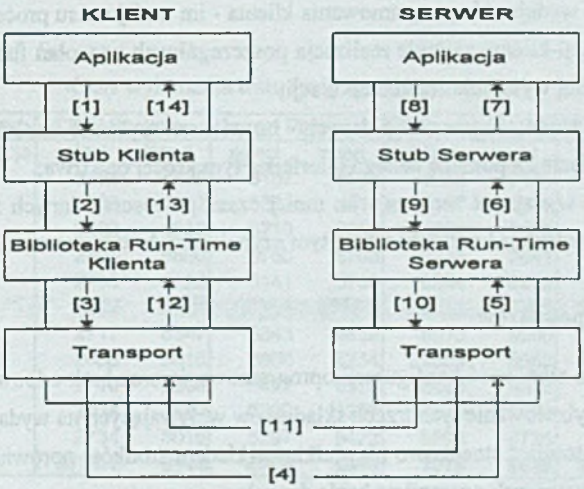
Większość aplikacji modelu BSD składa się z dwóch osobnych części. Jedna z nich to proces żądający połączenia, czyli **klient**, druga to proces udostępniający i akceptujący połączenie, czyli **serwer** [3].

Domyślnie SQL Server używa numeru gniazda *(ang. socket number)* 1433 (oficjalnie przyznany przez Internet Assigned Number Authority), ale można skonfigurować go na dowolny (nie kolidujący z innym oprogramowaniem) numer.

3.3. Interfejs Multi-Protocol (RPC)

Ta biblioteka komunikacyjna korzysta z udogodnień, jakie daje zdalne wywoływanie procedur (ang. Remote Procedure Call) dostępne w środowisku Windows NT/95, oraz daje możliwość równoczesnej komunikacji za pomocą więcej niż jednego interfejsu sieciowego (w chwili obecnej są to: TCP/IP Windows Sockets, NWLink IPX/SPX i Named Pipes).

RPC to narzędzie umożliwiające użytkownikom na pozornie bezpośrednie wywoływanie procedur, znajdujących się w zdalnym programie serwera. Zarówno aplikacja klienta, jak i serwera posiada w pamięci swój obszar adresowy, czyli miejsce, gdzie może przechowywać dane, wykorzystywane podczas wywoływania zdalnych procedur. Rysunek 2 ilustruje zasadę wywoływania takich procedur z wykorzystaniem RPC [3]:



Rys. 2. Zasada działania RPC [3]
Fig. 2. Principle of RPC work [3]

Biblioteka Muilti–Protocol jako jedyna daje możliwość kodowania transmisji pomiędzy klientem a serwerem. Kodowanie można wymusić dla wszystkich klientów korzystających

z danego serwera (opcja w programie instalacyjnym serwera) lub dla każdego klienta indywidualnie (wpis w *registry* na stacji klienta) [1].

3.4. Inne iterfejsy

Inne biblioteki komunikacyjne dostępne obecnie to:

a) NWLink IPX/SPX - dla klientów sieci Novell NetWare (DOS i Windows);
b) Banyan VINES SPP - dla klientów sieci Banyam VINES;
c) Apple Talk ADSP - dla klientów pracujących w środowisku Apple Macintosh OS;
d) DECnet sockets - dla klientów pracujących w środowisku VMS i Digital Pathworks;

wymagają one jednak dodatkowego oprogramowania odpowiednich firm.

4. Ocena wydajności

Interesującym zagadnieniem jest, w jaki sposób zmiana biblioteki komunikacyjnej może wpłynąć na szybkość działania oprogramowania.

Analiza zagadnienia prowadzi do wniosku, że biblioteka komunikacyjna może wpływać na wydajność systemu klient-serwer poprzez:

a) wpływ na wydajność oprogramowania klienta - im mniej czasu procesora i innych zasobów stacji klienta zajmuje realizacja poszczególnych wywołań funkcji DB-Library, tym szybciej wykonuje się kod aplikacji;
b) wpływ na szybkość transmisji danych - im większa szybkość transmisji jest możliwa do osiągnięcia za pomocą danego interfejsu, tym krócej ona trwa;
c) wpływ na wydajność serwera - im mniej czasu procesora i innych zasobów serwera absorbuje interfejs komunikacyjny, tym szybciej działa sam serwer.

4.1. Metoda pomiarów

Ze względu na złożoność konfiguracji oprogramowania oraz brak odpowiednich narzędzi programowych wyizolowanie tych trzech składników wpływających na wydajność jest bardzo trudne. Można natomiast stosukowo niewielkim nakładem środków porównać czas, jaki zajmuje wykonanie typowych operacji na bazie danych.

Do przeprowadzenia badań wydajności został napisany specjalny program testowy, który mierzył czas wykonania operacji wstawiania (INSERT, transmisja do serwera) i czytania (SELECT - FETCH, transmisja z serwera) rekordów z testowej bazy danych. Program napi-

sano z użyciem Microsoft Visual C++ ver. 4.0, komunikował się on z bazą danych poprzez interfejs ODBC.

Badania przeprowadzono w środowisku Microsoft Windows NT, serwerem był komputer ALR Evoultion z dwoma procesorami Pentium 90 MHz i 40 MB RAM, a klientem komputer Pentium 133 MHz 32 MB RAM. Komputery połączone były za pomocą sieci Ethernet. Badania przeprowadzono przy znikomym ruchu w sieci, użytkownik testowy był jedynym użytkownikiem w sieci lokalnej (nie wyklucza to niestety niewielkiego ruchu związanego np. z wymianą poczty, transmisjami organizacyjnymi routerów, itp.).

Badania przeprowadzono dla następujących bibliotek komunikacyjnych:

a) Named Pipes;

b) TCP/IP Sockets;

c) Multi-Protocol z kodowaniem;

d) Multi-Protocol bez kodowania.

Testy przeprowadzono dla 500 rekordów o rozmiarze 8 i 258 bajtów, tabela posiadała trzy pola: dwa pola typu całkowitego i jedno pole tekstowe.

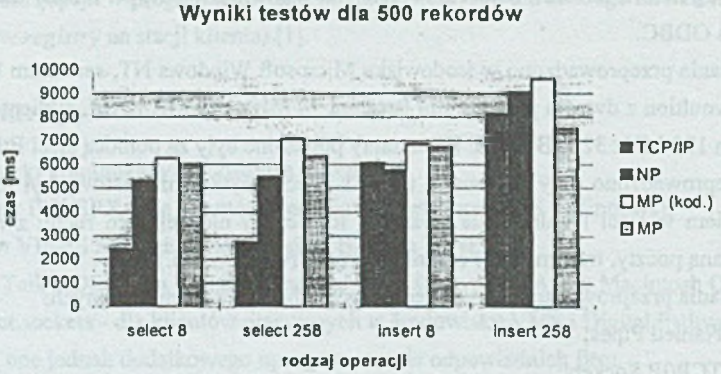
4.2. Wyniki testów

Wyniki testów przedstawia tabela 2 i wykres na rysunku 3.

Tabela 2

Czas wykonania testu [ms] dla 500 rekordów

Rozmiar rekordu	TCP/IP Sockets		Named Pipes		Multiprot. (kod.)		Multiprot.	
operacja	select	insert	select	insert	select	insert	select	insert
8	2438	5828	5187	5938	6266	7125	6015	6547
8	2406	5812	5250	5813	6187	6500	6704	7625
8	2422	5813	5219	5657	6282	7375	5734	6500
8	2406	6890	5750	5703	6234	6641	5610	6391
8	2438	5922	5141	5703	6344	6578	5984	6578
średnio	**2422**	**6053**	**5309.4**	**5762.8**	**6262.6**	**6843.8**	**6008.4**	**6728.2**
258	2717	8547	5343	8859	6516	9500	6250	1100
258	2688	8016	5968	8234	7859	8953	6187	9969
258	2719	8266	5422	8578	6469	9313	7031	8719
258	2734	7860	5390	11297	7640	11594	5969	8860
258	2734	8016	5297	8422	6906	8735	6187	8891
średnio	**2718.4**	**8141**	**5484**	**9078**	**7078**	**9619**	**6324.8**	**7607.8**



Rys. 3. Wykres wartości średnich czasów wykonania testów
Fig. 3. Chart of average test execution time

4.3. Dyskusja wyników

Przedstawione wyniki wskazują, że wybór biblioteki komunikacyjnej ma istotny wpływ na wydajność systemu współpracującego z SQL Server-em. Za najbardziej efektywny interfejs w badanym środowisku należy uznać bibliotekę TCP/IP Sockets; przewaga tego interfejsu jest szczególnie widoczna przy operacjach odczytu danych.

Przy stosowaniu biblioteki Milti-Protocol różnice w wydajności wynikające z zastosowania kodowania danych są zauważalne dla długich rekordów i w ekstremalnym przypadku osiągają 20%.

Podczas przeprowadzania testów zauważono, że na wyniki znacznie większy wpływ ma obciążenie serwera lub/i stacji klienta niż wybrany interfejs, dlatego testy były przeprowadzane w prawie laboratoryjnym środowisku. Jest to zrozumiałe, należy się bowiem spodziewać, że komunikacja poprzez szybką sieć lokalną stanowi mniejszy procent obciążenia systemu niż inne jego czynności, tym bardziej że maksymalna osiągnięta szybkość efektywnej transmisji danych wyniosła ok. 47 kB/s (TCP/IP Sockets, insert, rekordy po 258 B), co jest wartością niewielką w porównaniu do przepustowości sieci.

LITERATURA

[1] Microsoft SQL Server Books Online; 1992-1995 Microsoft ® Corporation.

[2] Microsoft Win32 Software Development Kit, 1985-1995 Microsoft ® Corporation.

[3] Kowalczyk A., Twardoń A., Ziółkowski K.: Interfejsy programowe, umożliwiające komunikację między procesami w środowisku sieciowym Microsoft Windows. ZN Pol. Śl. s. Informatyka, Gliwice 1996, z. 30.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 21 listopada 1996 r.

Abstract

This article describes features of network communication libraries provided with Microsoft SQL Server and network interfaces that lay under them, then provides performance comparison.

Chapter one is an introduction to database servers network communication problems and assumes that selection of network interface is not trivial.

Chapter two presents Microsoft SQL Server software architecture with accent on network subjects and accessibility from different operating system platforms (DOS, Windows 3.x, Windows NT, Windows 96).

Chapter three describes, with some more details, SQL Server network libraries: Named Pipes, TCP/IP sockets, Multi-Protocol; and network programming interfaces that lay under them.

The last chapter describes methods and results of performance measurement for described earlier libraries. The performance was measured in laboratory environment on Microsoft Windows NT operating system working both on Pentium server an workstation, linked with Ethernet. The best performance result in this environment was achieved by TCP/IP Sockets library.