Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

KONCEPCJA ZASTOSOWANIA SZTUCZNYCH SIECI NEURONOWYCH W DIAGNOSTYCE BAZ DANYCH ORACLE

Jarosław Gramacki

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: j.gramacki@iie.uz.zgora.pl

STRESZCZENIE

W artykule przedstawiono koncepcję zastosowania sztucznych sieci neuronowych w diagnostyce kluczowych parametrów wydajnościowych bazy danych ORACLE. Pokazano, że jest uzasadnione i możliwe zbudowanie układu neuronowego przewidującego ewentualne pogorszenie się wybranych parametrów bazy.

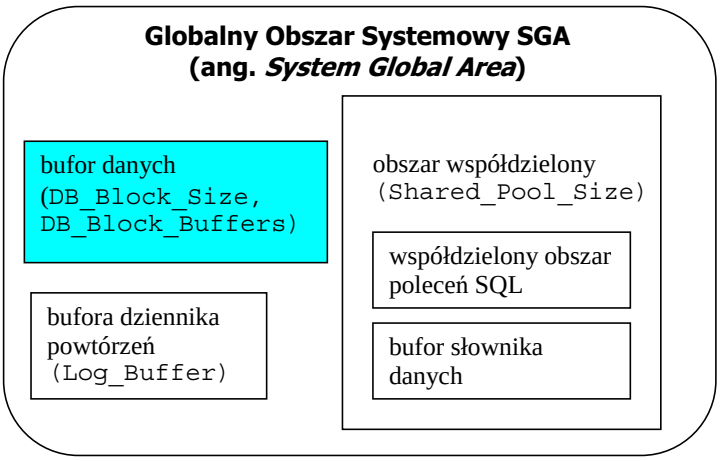
1. WPROWADZENIE

Wymienić można co najmniej parę czynników i związanych z nimi zadań wpływających na wydajność bazy danych w obszarze związanym z zarządzaniem pamięcią [2]. Wymienione poniżej oraz zilustrowane na rysunku 1 elementy tworzą tzw. globalny obszar systemowy SGA (ang. *System Global Area*), który jest niczym innym jak odpowiednio zarządzanym na potrzeby bazy danych Oracle obszarem pamięci operacyjnej komputera. Pamięć ta jest zawsze alokowana w momencie uruchamiania instancji bazy poleceniem `startup` (wydanym z poziomu programu *Server Manager* – do wersji 8i, lub *SQL*Plus* – w wersjach 9i oraz 10g). Jest ona jednocześnie współdzielona przez wszystkich użytkowników podłączonych w danym momencie do bazy.

Do głównych zadań administratora bazy należy [3, 4, 6]:

- strojenie obszaru współdzielonego (ang. *Shared Pool*),
- strojenie dziennika powtórzeń (ang. *Redo Log Buffer*),
- strojenie bufora danych (ang. *Buffer Cache*).

W pracy skupiono się nad zagadnieniem obsługi bufora danych. Zaproponowana metodologia nie będzie merytorycznie różnić się w pozostałych przypadkach a różnice będą jedynie ilościowe.



Rys. 1. Globalny obszar systemowy SGA. Czcionką nieproporcjonalną podano odpowiednie parametry pliku konfiguracyjnego instancji `init<SID>.ora`

Wielkość obszaru SGA określić można wydają na przykład poniższe polecenie

```
-- Parametry instancji

SELECT * FROM v$sga;

-- przykładowy wynik

NAME                 VALUE
---------------- ---------
Fixed Size           75804
Variable Size    106840064
Database Buffers  49594368     =  8192          * 6054
                                   DB_Block_Size * DB_Block_Buffers
Redo Buffers         77824
```

Wielkość bufora danych oblicza się jako iloczyn dwóch parametrów konfiguracyjnych instancji Oracle: `DB_Block_Size` oraz `DB_Block_Buffers`. Pierwszego z nich nie można zmieniać bez przebudowania bazy danych, gdyż jego wielkość ustawiana jest w momencie tworzenia nowej bazy danych. Podaje on w bajtach rozmiar najmniejszego elementu obszaru bufora danych – jednego bloku danych. Doborowi podlega parametr `DB_Block_Buffers`, który określa liczbę bloków, które tworzą cały bufor danych.

W obszarze pamięci jakim jest bufor danych przechowuje się dane użytkowe (tabele, indeksy), do których dostęp mają wszyscy użytkownicy systemu (oczywiście w zakresie posiadanych do nich uprawnień). Po wystartowaniu bazy danych bufor danych jest pusty i zapełnia się stopniowo wraz z pobieraniem przez użytkowników kolejnych danych z plików dyskowych. Gdy w jakimkolwiek momencie pracy któryś z użytkowników zażąda

jakichkolwiek danych, system w pierwszej kolejności stara się odnaleźć je w pamięci operacyjnej czyli w buforze danych. Ponieważ czas dostępu do pamięci jest nieporównywalnie krótszy niż czas dostępu do pliku (plików) dyskowego, więc optymalny dobór parametrów bufora danych ma bezpośredni wpływ na średni czas dostępu do danych.

W pracy pominięto oczywisty, choć nie zawsze uświadamiany sobie przez administratorów, aspekt pierwszeństwa strojenia modelu danych, na których działa aplikacja, strojenia (optymalizacji) samej aplikacji oraz ściśle związane z tym ostatnim strojenie zapytań SQL. Wydaje się, że w wielu przypadkach problem (pozornie słabej) wydajności systemu leży nie w niewłaściwych parametrach instancji bazy, ale w aplikacjach, które w niej działają.

Innym problemem jest często duże przeszacowanie zaalokowanych zasobów systemowych (głównie pamięci) na potrzeby instancji. Pomijając oczywiste w takim przypadku marnotrawienie zasobów sprzętowych komputera, problemem w takim przypadku może być duży narzut obliczeniowy związany z koniecznością obsługi np. nieracjonalnie dużego bufora danych.

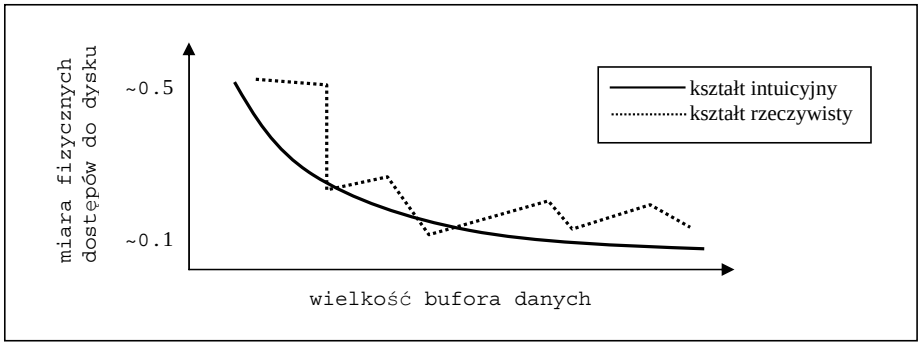
Należy również wspomnieć, że w buforze danych mogą również znajdować się kopie tych samych danych, jeśli na przykład więcej niż jedna transakcja operuje na tych samych danych i zmiany nie zostały jeszcze zatwierdzone (ang. *committed*). Fakt ten nie zmienia jednak istoty buforowania w pamięci zawartości bazy danych.

Parametrem charakteryzującym "jakość" bufora danych jest tzw. współczynnik trafień w bufor (ang. *Cache Hit Ratio*). Wyliczany on jest na podstawie analizy całkowitej liczby fizycznych operacji odczytu z dysku (ang. *physial reads*) oraz całkowitej liczby operacji odczytu danych, które odnaleziono w buforze. Dane do jego wyliczenia (tzw. statystyki) znajdują się odpowiednich tzw. tabelach (widokach) słownikowych systemu Oracle i uaktualniane są na bieżąco w trakcie pracy systemu. Zwykle użyteczną ich wartość można obliczyć dopiero po kilku / kilkunastu godzinach pracy systemu, licząc od momentu uruchomienia instancji.

```
-- Wyliczenie współczynnika trafień w bufor

SELECT (1 - (PR.value / (DBG.value + CG.value)))*100 "Cache Hit Ratio"
FROM
  v$sysstat PR, v$sysstat DBG, v$sysstat CG
WHERE
  PR.name   = 'physical reads' AND
  DBG.name  = 'db block gets' AND
  CG.name   = 'consistent gets';

-- przykładowy wynik

Cache Hit Ratio
---------------
     95,5167612
```

Zależność między wielkością bufora a współczynnikiem trafień nie jest liniowa. Oznacza to, że po osiągnięciu pewnej granicznej wielkości bufora danych, dalsze zwiększanie jego rozmiaru daje znikome efekty, a dodatkowy narzut obliczeniowy związany z zarządzaniem większą ilością pamięci może skutkować nawet spowolnieniem pracy całego systemu.

Wspomnianą nieliniowość łatwo pokazać eksperymentalnie, wykonując wykres zależności współczynnika trafień od wielkości bufora danych dla zadanego scenariusza dostępu do danych. Nieliniowość ta szczególnie uzasadnia zastosowanie sieci neuronowych do analizy (aproksymacji) tej zależności [7, 8]. Na rysunku 2 pokazano hipotetyczną zależność pomiędzy wielkością fizycznych dostępów do danych dyskowych a wielkością bufora. W rzeczywistości liczba analizowanych parametrów będzie dużo większa.



Rys. 2. Ilustracja rzeczywistego i intuicyjnego przebiegu zależności ilości dostępów do fizycznych plików z danymi od wielkości bufora danych

Uwaga. Przy analizie współczynnika trafień pamiętać należy, że wyniki tzw. pełnego odczytu tabeli (ang. *full table scan*) nie są domyślnie kierowane w "normalny" sposób do bufora, ale w sposób zapewniający możliwie szybkie "pozbycie" się wyników takich odczytów z bufora. W czasie testów wygodnie jest jednak zapełniać bufor takimi właśnie odczytami (wtedy łatwo o duże porcje danych). Domyślne działanie procesu obsługującego bufor łatwo jednak zmienić poprzez używanie w zapytaniach SQL wskazówki `CACHE` (ang. *hint*). Opcja ta jest również często wykorzystywana do zapewnienia sobie szybkiego dostępu do tzw. małych słowników (ang. *small lookup tables*), które w normalnym trybie (`NOCHACHE`) byłyby szybko usunięte z bufora danych.

Wielkość parametru `DB_Block_Buffers` dobiera się eksperymentalnie, korzystając z wiedzy i doświadczenia administratora systemu. Jest to jednak zawsze proces długotrwały a często i uciążliwy, gdyż wymaga restartu instancji i następnie odpowiednio długiej pracy w normalnych warunkach w celu wypełnienia się bufora danych i zaistnienia odpowiednio dużej ilości dostępów do zgromadzonych w nim danych.

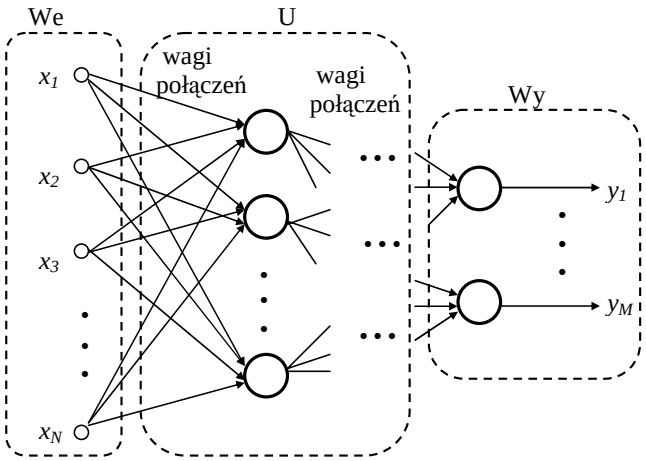
Odpowiednie procedury, opisane w [4, 6], pozwalają na symulację wpływu zwiększenia i / lub zmniejszenia bufora danych na wynikowy współczynnik trafień. Stosowane tam podejście jest jednak uciążliwe – podobnie jak dla przypadku opisanego wyżej.

2. SIECI NEURONOWE

Najogólniej stwierdzić można [7, 8], że dobrze skonstruowana sieć neuronowa (SN) jest w stanie "nauczyć się" aproksymować dowolną funkcję wielu zmiennych *f(x1, x2,. x3,...,xN).* Zmiennymi mogą być dowolne wartości reprezentujące wybrany obiekt sterowania (w naszym przypadku: bazy danych).

Cechą podstawową SN jest więc możliwość uczenia się odwzorowywania dowolnej funkcji, najczęściej zależnej od wielu zmiennych i silnie nieliniowej. Ponieważ jest to aproksymacja, a nie interpolacja, to sieć jest w stanie **uogólniać** wiedzę, wykrywać istniejące pomiędzy zmiennymi zależności. W procesie uczenia sieć tworzy więc rodzaj statystycznego modelu uczonej zależności (funkcji).

SN jest zbudowana z tzw. neuronów, ułożonych warstwami. Szczegóły budowy pojedynczego neuronu [7, 8] nie są w tych dosyć ogólnych rozważaniach istotne. Neurony tej samej warstwy nie są połączone między sobą, ale neurony dwóch sąsiednich warstw są połączone "każdy z każdym". Wyróżnia się warstwę wejściową (We), warstwy ukryte (U) (zazwyczaj 1 lub 2) i warstwę wyjściową (Wy). Sygnał przechodzi od warstwy We poprzez U aż do Wy, skąd jest odbierany i interpretowany. Sygnał przechodząc przez dane połączenie wewnątrz SN zostaje pomnożony przez liczbę przypisaną temu połączeniu, zwaną wagą połączenia. Wagi mogą mieć dowolne wartości rzeczywiste. Siec taka zwana jest również perceptronem wielowarstwowym.



Rys. 3. Sieć jednokierunkowa jako perceptron wielowarstwowy. Tworzą go neurony ułożone w warstwach o jednym kierunku przepływy sygnałów, o połączeniach międzywarstwowych jedynie pomiędzy sąsiednimi warstwami

W pierwszej kolejności należy ustalić, jak sieć będzie zbudowana. Ilość neuronów wejściowych i wyjściowych jest określona z góry dla danego problemu, natomiast ilość neuronów w warstwach ukrytych (oraz ilość samych warstw, choć zazwyczaj stosuje się jedną, góra dwie) można regulować.

Po wyborze struktury sieci (wstępnej, gdyż w kolejnych krokach może zajść potrzeba jej modyfikacji) przechodzi się do zasadniczej czynności – tzn. di jej uczenia. Uczenie to po prostu dobieranie wag połączeń między neuronami w taki sposób, aby po podaniu na wejście sieci jakichś wartości, na jej wyjściu uzyskać odpowiedni wynik. Uczenie odbywa się w następujący sposób:

Przygotowuje się zestaw danych wejściowych wraz z odpowiadającymi im wynikami, jakie powinna uzyskać sieć. Wagi sieci inicjalizuje się w sposób przypadkowy. Następnie podaje się kolejno, wiele razy, wszystkie zestawy danych wejściowych i patrzy, na ile odpowiedź sieci różni się od poprawnego wyniku. Oblicza się różnicę i następnie w odpowiedni sposób koryguje się wagi połączeń wewnątrz sieci (poczynając od ostatniej warstwy, aż do warstwy wejściowej). W wyniku otrzymujemy sieć nauczoną, tj. mającą odpowiednie wartości wag, aby efektywnie rozwiązać postawione jej zadanie.

Po nauczeniu sieci trzeba ją następnie przetestować – do tego wykorzystuje się kolejny zestaw danych wejściowych i wyjściowych, ale takich, które nie były wykorzystane podczas uczenia. Jeśli na tym zestawie odpowiedzi sieci na zadane wymuszenia będą poprawne, to można się spodziewać, że w sytuacjach przyszłych (dla jakichś nieznanych danych) też tak będzie. Jeśli nie – trzeba powtórzyć uczenie, ewentualnie zmieniając budowę sieci albo zestaw danych uczących.

Generalnie rzecz biorąc, tylko jeden układ przestrzenny sieci jest optymalny. Jeśli warstwy ukryte mają zbyt wiele neuronów, sieć nadmiernie dopasowuje się do nauczanych danych (jej wynik staje się bliższy interpolacji) i przez to traci zdolność uogólniania. O takiej sieci mówi się, że jest "przeuczona". Jeśli natomiast neuronów w warstwach ukrytych jest za mało, sieć nie jest w stanie poprawnie odtworzyć uczonej funkcji, jest "niedouczona".

3. PRZEWIDYWANIE PARAMETRÓW BUFORA DANYCH

W naszym przypadku aproksymować będziemy opisany w rozdziale 1 współczynnik trafień (wyjście SN) w funkcji takich zmiennych jak: wielkość bufora danych, wielkość obiektów (tabel) w bazie danych, sposób i częstotliwość dostępu do danych, wykorzystanie indeksów, itp. (wejścia SN) [1]. W rzeczywistości analiza tych parametrów (zmiennych) w kontekście strojenia wydajności bufora danych jest klasycznym zagadnieniem diagnostycznym, w którym wykorzystywana jest wiedza eksperta (tu: administratora systemu Oracle). Oczywiście model matematyczny (funkcja wielu zmiennych) nie jest znana. W rzeczywistości nawet trudno wyobrazić sobie zbudowanie takiego, nawet przybliżonego, modelu. W

praktyce funkcja ta podana będzie w postaci tabelarycznych danych będących wynikiem np. wielu przeprowadzonych eksperymentów na bazie. Zebrane dane zostaną zinterpretowane przez sieć neuronową, a zbudowany w ten sposób neuronowy model wybranego aspektu działania bazy danych zostanie użyty do predykcji wartości współczynnika trafień dla innych wartości zmiennych wejściowych.

4. ZADANIE TESTOWE

Korzystano z klasycznego w systemie Oracle schematu `SUMMIT2`, (skrypt o nazwie *summit2.sql* instaluje się automatycznie w systemie) do którego wprowadzono bardzo dużą ilość danych do tabel biorących udział w złączeniach N:N. Przykładowo zarejestrowano ok. 200 tysięcy faktur sprzedaży, ok. 100 tysięcy zamówień, podobną liczbę stanów magazynowych produktów oraz kilkanaście tysięcy danych o klientach firmy. Dane udostępniono trzem użytkownikom.

W losowy sposób, losowy użytkownik wybiera po kilkanaście tysięcy wierszy z losowo wybranego układu odpowiednich (możliwych do sensownego złączenia) tabel. Wymuszono przy tym, aby wszystkie dane zostały normalnie przesłane do bufora (hint `CACHE`). Ilość i częstotliwość pobrania określonego zbioru danych tworzą tzw. miarę charakteru wykorzystania zgromadzonych danych [1]. Wielkości te (odpowiednio przeskalowane) podawane są na wejścia sieci neuronowej. Kolejną daną wejściową jest wielkość bufora danych. Wyjściem jest wyliczony dla takiego układu danych współczynnik trafień w bufor.

Dla powyższych danych przeprowadzono proces uczenia wielowarstwowej sieci neuronowej z jedną i dwoma warstwami ukrytymi metodą wstecznej propagacji błędów.

Praktyczne wykorzystanie nauczonej sieci będzie polegało np. na otrzymaniu informacji o charakterze wykorzystania zgromadzonych danych na podstawie aktualnej wielkości bufora czy też o (przewidywanej) wielkości współczynnika trafień w bufor na podstawie zmiany charakteru wykorzystania danych przez użytkowników bazy. Podobnie dla oczekiwanego współczynnika trafień można by określać konieczne zmiany w wielkości bufora danych.

5. PODSUMOWANIE

W pracy zaprezentowano koncepcję zastosowania sieci neuronowych w analizie danych istotnych dla optymalnego zarządzania pamięcią w systemie Oracle. Uzasadniono powód ich zastosowania oraz przedstawiono szkic rzeczywistych testów w środowisku Oracle dostarczających bazę wiedzy wykorzystywaną w uczeniu wybranej struktury sieci neuronowej.

Aby przekonać się o praktycznej celowości zaprezentowanego w pracy podejścia należałoby wykonać większą liczbę testów na rzeczywistej bazie, zawierającej dużą liczbę danych i

dodatkowo bardzo intensywnie eksploatowanej. Autor pracy prowadzi obecnie takie eksperymenty, wyniki tych prac będą sukcesywnie publikowane.

LITERATURA

[1] U. Harder, T. MacLeod: *Predicting buffer hit ratios with neural networks*, UK Performance Engineering Workshop 1999, Bristol University (ISBN 0952402785)

[2] *Oracle Concepts:* Dokumentacja systemu Oracle

[3] *Oracle Administrator's Guide:* Dokumentacja systemu Oracle

[4] *Oracle Designing and Tuning for Performance:* Dokumentacja systemu Oracle

[5] *Schemat SUMMIT2 (skrypt summit2.sql)*: dostępny na każdej płycie instalacyjnej systemu Oracle

[6] J. Wrembel, M. Jezierski, M. Zakrzewicz: *System zarządzania bazą danych Oracle 7 i Oracle 8*, Wydawnictwo Nakom, Poznań, 2000, ISBN 83-86969-34-2

[7] J. Korbicz, A. Obuchowicz, D. Uciński: *Sztuczne sieci neuronowe*, *Podstawy i zastosowania*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994

[8] R. Tadeusiewicz: *Sieci neuronowe*, Warszawska Akademicka Oficyna Wydawnicza RM, Warszawa 1993