Ukryte modele Markowa jako mechanizm automatycznej predykcji żądań użytkowników w systemach z materializacją hierarchiczną

Witold Andrzejewski[1], **Zbyszko Królikowski**[1], **Mariusz Masewicz**[1], **Robert Wrembel**[1]

Streszczenie: Artykuł omawia szczegóły eksperymentu, którego celem była ocena mechanizmu automatycznego podejmowania decyzji o ewentualnej rematerializacji obiektów będących zmaterializowanymi wynikami działania metod w obiektowej bazie danych. Badanym mechanizmem predykcji były ukryte modele Markowa, a środowisko, w których badano ich przydatność to obiektowa baza danych z zaimplementowanym mechanizmem hierarchicznej materializacji wyników działania metod.

Słowa kluczowe: materializacja, obiektowe, bazy, danych, predykcja, żądań.

1. Wprowadzenie

Idea utrwalania wyników działania różnych elementów systemu informatycznego (ang. *method materialization, caching*) jest stosowana od samego początku istnienia systemów informatycznych. Mechanizmy takie pojawiają się w każdym aspekcie działania tych systemów, począwszy od wspierania działania systemów wejścia wyjścia (bufory dyskowe), komunikacji pomiędzy pamięcią a procesorem (pamięć cache), a skończywszy na zaawansowanych technikach przyspieszania pracy protokołów sieci komputerowych (cache przeglądarek, DNS), czy wewnątrz zaawansowanych baz danych (bufory danych w bazie danych, perspektywy zmaterializowane).

Popularność opisanego mechanizmu, nie oznacza, że jest on pozbawiony wad. Do najważniejszych należą:

- konieczność składowania zmaterializowanych wyników - często nie jest znana z góry ich ilość,
- konieczność zaimplementowania mechanizmu zapewniającego dostarczenie użytkownikowi aktualnego wyniku - jeżeli to możliwe to z bufora materializacji, a jeżeli nie jest to możliwe to wykonując właściwe działanie,
- konieczność pielęgnowania bufora materializacji - nowo wyliczone wyniki należy składować (ten proces będziemy dalej nazywać *materializowaniem wyników obliczeń*), wyniki o których z różnych źródeł dowiedzieliśmy się, że są nieaktualne należy usuwać (ten proces będziemy nazywali *inwalidowaniem zmaterializowanych wyników obliczeń*),

[1] Wydział Informatyki i Zarządzania, Politechnika Poznańska, Piotrowo 2, 60-965 Poznań
e-mail: {wandrzejewski,zkrolikowski,mmasewicz,rwrembel}@cs.put.poznan.pl

- brak możliwości przewidywania czasu odpowiedzi systemu na zapytanie użytkownika - Wynik może być w buforze i est zwracany natychmiast, lub też należy go obliczyć.

Spróbujemy tu przedstawić propozycję rozwiązania ostatniego z wymienionych powyżej problemów na przykładzie materializacji wyników działania metod w obiektowej bazie danych.

Wybór obiektowego systemu zarządzania bazą danych jako obiektu badań nie jest przypadkowy. Znajdują one coraz więcej zastosowań w systemach wymagających skomplikowanego modelu danych. Do takich systemów należą systemy typu CAD, CAM, czy też GIS. Wobec takich systemów stawia się bardzo wygórowane wymagania dotyczące ich wydajności i wówczas krytycznym parametrem staje się zarówno czas dostępu do pól składowanych obiektów i czas wykonywania metod zaimplementowanych dla tych obiektów. Aby zwiększyć wydajność dostępu do wartości przechowywanych w polach obiektów można stosować różnego rodzaju indeksy.

Największym problemem obiektowych baz danych jest więc czas dostępu do wyników działania metod, które są podstawowym sposobem wzajemnej komunikacji pomiędzy obiektami a także pomiędzy światem zewnętrznym a obiektem. To właśnie czas wykonania metody decyduje o czasie potrzebnym systemowi na wysłanie ostatecznej odpowiedzi do użytkownika. Niestety możliwości optymalizowania czasu wykonania metody są ograniczone - w dużej mierze czas ten determinowany jest stopniem skomplikowania metody.

Pewnym pomysłem na skrócenie czasu oczekiwania użytkownika na wynik działania metody jest materializacja wyników działania metod. Mechanizm ten można w skrócie scharakteryzować następującym pseudoalgorytmem: „jeżeli coś obliczyłeś (wykonałeś metodę) to zwróć wynik użytkownikowi, i przy okazji zapamiętaj go - jeżeli wkrótce ktoś także będzie potrzebował tego wyniku, wówczas zamiast ponownie wykonywać metodę zwrócisz wynik z bufora materializacji".

Zmaterializowany wynik działania metody zostanie zwrócony tylko wówczas, gdy system będzie miał pewność, że jest on ciągle poprawny, czyli: (1) została wywołana ta sama metoda, (2) na rzecz tego samego obiektu, (3) przesłano te same wartości parametrów wywołania metody i (4) w międzyczasie nikt nie modyfikował tych pól obiektu, których wartość ma wpływ na wynik działania metody. Jeżeli nie są spełnione warunki 1-3 to system jest zmuszony wykonać wywoływaną metodę (pola wrażliwe dla wyniku działania danej metody), gdyż nie posiada w swoim buforze odpowiedniej materializacji. Jeżeli natomiast nie jest spełniony warunek 4 to system jest zmuszony do ponownego wykonania danej metody (i ponownej materializacji jej wyniku). Aby zminimalizować ryzyko wystąpienia takiej sytuacji w niniejszym artykule prezentujemy mechanizm pozwalający na przewidywanie, przyszłego zapotrzebowania na inwalidowany właśnie wynik działania metody. Predykcja taka powinna odbywać się w warstwie bazy danych odpowiedzialnej za obsługę materializacji wyników działania metod.

Dziedziny zastosowania mechanizmów materializowania metod obejmują: obiektowe bazy danych, obiektowo-relacyjne hurtownie danych (Gopalkrishnan, 2000 i Huynh, 2000), zmaterializowane perspektywy obiektowe (Motschnig, 1996 i

Wrembel, 1998), obiektowe systemy przetwarzania rozproszonego, np. CORBA.

Przedstawiona zostanie tu koncepcja materializacji hierarchicznej, poszerzona o mechanizmy predykcji odpowiadającej na pytanie „czy dokonać automatycznej rematerializacji właśnie inwalidowanej materializacji" oraz struktury danych niezbędne do jej implementacji.

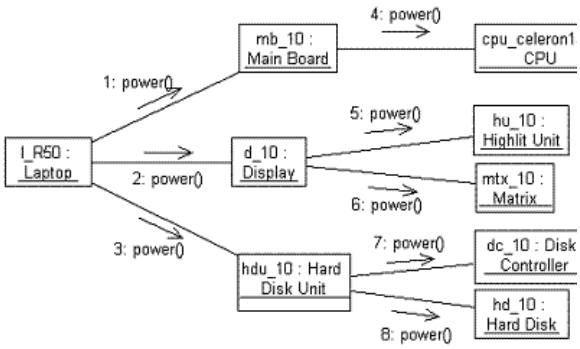
2. Prace badawcze

Mechanizmy materializacji wyników działania metod w obiektowych bazach danych zostały przedstawione w wielu publikacjach naukowych (Jhingran, 1991, Bertino, 1991, Kemper, 1991, 1994, Pugh, 1998, Eder, 1994, Gardarin, 1996). Zaprezentowano w nich różne podejścia do problemu materializowania wyników i utrzymywania uzyskanego w ten sposób bufora materializacji. W pracy (Jhingran, 1991) omówiono wyniki teoretycznej analizy kosztów przechowywania danych wyliczanych przez metody. Nie uwzględniono jednak możliwości wzajemnego wywoływania się metod. Nie zostały też uwzględnione koszty związane z pielęgnowaniem zbioru zmaterializowanych wyników - na przykład automatyczna inwalidacja zmaterializowanego wyniku, konieczność powtórnego wykonania metody w celu rematerializacji jej wyniku po inwalidacji.

Inne koncepceje, to na przykład wykorzystanie B-drzewa do przechowywania zmaterializowanych wyników działania metod (Bertino, 199). W tym przypadku materializowane są wyniki metod wywoływanych bez argumentów wejściowych.

W pracach (Kemper, 1991, 1994) przedstawiono koncepcję materializowania metod posiadających argumenty wywołania. Prace te zawierają opis proponowanego rozszerzenia słownika bazy danych o struktury niezbędne do przechowywania zmaterializowanych wyników. Zmaterializowane wyniki działania metod są przechowywane w strukturze o nazwie Generalised Materialisation Relation. W celu pielęgnowania zmaterializowanych wartości, każdy obiekt posiada zbiór tych metod, które korzystały z danego obiektu. Zbiór ten jest wykorzystany do odszukania i unieważnienia zmaterializowanych wyników na skutek zmodyfikowania obiektów wykorzystanych w metodach. Również ta koncepcja nie uwzględnia zależności między metodami. Ponadto, projektant systemu jest odpowiedzialny za zdefiniowanie struktur danych, które dla każdego obiektu będą przechowywały zbiór metod korzystających z tego obiektu.

W pracy (Pugh, 1998) i (Eder, 1994) zaprezentowano koncepcję tzw. temporalnej materializacji, w której wyniki działania metod są przechowywane w pamięci operacyjnej serwera przez określony czas. W (Eder, 1994) zaproponowano tzw. inverse methods, których wartości są przechowywane w systemie tylko w czasie wykonywania zapytania odwołującego się do tych metod. W pracy (Pugh, 1998) przedstawiono koncepcję, w której metody złożone są dekomponowane do metod prostszych. Wyniki tych ostatnich są przechowywane w systemie, pod warunkiem, że argumentami wywołania są stałe.



Rysunek 1. Przykładowy diagram współpracy pomiędzy obiektami bazy danych projektów komputerów mobilnych

3. Materializacja hierarchiczna - koncepcja

W artykule (Jezierski, 2003) opisano technikę hierarchicznej materializacji wyników działania metod. Hierarchiczna oznacza, że materializowany jest nie tylko ostateczny wynik działania metody wywołanej z określonymi parametrami na rzecz konkretnego obiektu, ale także wyniki metod wywoływanych przez daną metodę. Przechowywana dodatkowo informacja o powiązaniach pomiędzy obiektami pozwala na zapewnienie dostarczenia użytkownikowi poprawnych wyników nawet wówczas, gdy ulegną zmianie atrybuty wrażliwe dla metody znajdującej się daleko w hierarchii wzajemnych wywołań materializowanych metod. W ogólnym przypadku, metody mogą posiadać wiele argumentów wywołania różnych typów.

Przykład. W celu zilustrowania koncepcji materializacji hierarchicznej rozważmy uproszczoną bazę danych. Każda klasa reprezentuje element komputera. Komputer mobilny (klasa *Laptop*) składa się z płyty głównej (*Main_Board*), ekranu (*Display*) i jednostki dyskowej (*Hard_Disk_Unit*). Płyta główna jest zbudowana z jednostki centralnej (*CPU*). Ekran jest zbudowany z matrycy (*Matrix*) i jednostki podświetlającej (*Highlit_Unit*). Jednostka dyskowa składa się z kontrolera (*Disk_Controller*) i samego dysku (*Hard_Disk*).

Każda z powyższych klas posiada metodę *power()* obliczającą zużycie mocy danego obiektu. Diagram współpracy dla instancji omawianej bazy przedstawiono na Rys.1. Wartość metody *power()* dla obiektu *l_R50* (instancja klasy *Laptop*) jest sumą mocy pobieranej przez obiekt *mb_10* (*Main_Board*), *d_10* (*Display*) i *hdu_10* (*Hard_Disk_Unit*).

Załóżmy, że metoda *power()*, wywołana na rzecz obiektu *l_R50*, została zmaterializowana hierarchicznie. W naszym przykładzie, zostaną dodatkowo materializowane następujące wyniki: *mb_10.power()*, *d_10.power()* i *hdu_10.power()*.

Przyjmijmy następnie, że obiekt składowy *cpu_celeron14* został zastąpiony przez *cpu_pentiumM17*. Zmiana ta skutkuje zwiększonym zużyciem mocy przez instancję klasy *Main_Board* (obiekt *mb_10*), co z kolei wpływa na zwiększenie poboru mocy całego komputera *l_R50*. Należy więc unieważnienić wyniki wcze-

śniej zmaterializowanych metod *l_ R50.power()* i *mb_ 10.power()* i ponownenie je wyliczyć przy następnym wywołaniu. Zauważmy jednak, że ponowne wyliczenie *l_ R50.power()* może wykorzystać nadal aktualne wartości *d_ 10.power()* i *hdu_ - 10.power()*.

Należy zauważyć, że na wynik wywołanej metody użytkownik ocekuje albo bardzo krótko (wynik, był zmaterializowany) albo zdecydowanie dłużej. Ten dłuższy czas oczekiwania jest czasem wyliczania wyniku metody (materializacja hierarchiczna nie wydłuża czasu wykonywania metody (Jezerski, 2003,2004, Masewicz, 2005, 2006)).

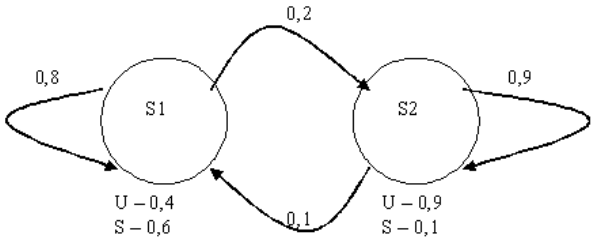
Proponujemy tu wykonanie predykcji żądań użytkowników już w momencie wykonywania operacji modyfikacji danych, czyli już podczas inwalidowania materializacji podejmujemy decyzję, czy od razu wyliczyć nową wartość metody i ją zmaterializować (automatyczna rematerializacja), czy czekać aż użytkownik sam wywoła metodę i wówczas materializować wyliczony wynik. W obu przypadkach nakład pracy systemu jest dokładnie taki sam, gdyż za każdym razem metoda jest wyliczana tylko raz i jej wynik jest materializowany. Problemem są tylko sytuacje, kiedy nie uda nam się podjąć poprawnej decyzji o automatycznej rematerializacji. Rozróżniamy tutaj dwa rodzaje błędów, które można w takim przypadku popełnić:

- Podjęliśmy decyzję o nie wykonywaniu automatycznej rematerializacji, a za chwilę użytkownik zażądał wykonania tej metody - konsekwencją takiego błędu jest oczekiwanie użytkownika na wynik metody.
- Podjęliśmy decyzje o wykonaniu automatycznej rematerializacji, a następną operacją aktualizacja, która od razu wykona inwalidację stworzonej materializacji - oznacza to, że system wykonał pracę w celu automatycznego rematerializowania wyniku metody, która to materializacja i tak nigdy nie została wykorzystana.

Skuteczna predykcja w tym wypadku może znacząco wpłynąć na zadowolenie użytkowników bez jednoczesnego zwiększania obciążenia systemu. W tym artykule zaproponujemy algorytm, który z jednej strony będzie dążył do zwiększania zadowolenia użytkowników, starając się zapewnić im zmaterializowane wyniki nawet wtedy, kiedy przed chwilą była wykonywana operacje inwalidacji, a z drugiej strony nie będzie powodował obciążenia systemu większego niż w przypadku braku materializacji, czy też standardowej materializacji hierarchicznej (w której materializacja wyniku następuje niejako przy okazji normalnego wyliczania metody).

4. Ukryte modele Markowa

Ukryte modele Markowa (ang. Hidden Markov Model, określane dalej: HMM) to mechanizm, który na podstawie obserwacji dotychczasowego zachowania obiektu (zanotowanej historii wyemitowanych symboli - w naszym przypadku operacji na danym obiekcie) pozwoli przewidzieć, jak zachowa się ten obiekt (a dokładniej użytkownik wykorzystujący dany obiekt) w przyszłości. U postaw tego mechanizmu leży założenie procesu charakteryzującego się brakiem pamięci dotyczącej poprzedniego zachowania tego procesu - własność Markowa. W naszym przypadku procesem



Rysunek 2. Przykładowy automat reprezentujący stany, prawdopodobieństwa przejścia pomiędzy stanami i prawdopodobieństwa wyemitowania określonego symbolu

będzie aplikacja użytkownika. O procesie tym wiemy tylko tyle, że jest opisany jakimś automatem wielostanowym, gdzie dla dowolnej chwili automat ten znajduje się w jakimś stanie (nieznanym dla nas) i z określonym prawdopodobieństwem (zależnym od aktualnego stanu) generuje symbol wyjściowy, po czym przechodzi do kolejnego stanu.

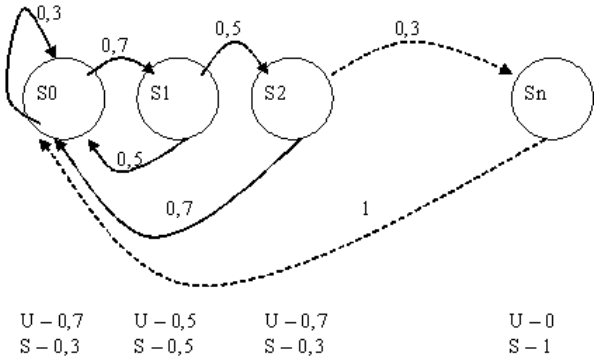
W naszym przypadku zbiór symboli wyjściowych zawiera dwa symbole: U - wykonano operację modyfikacji i należy dokonać inwalidacji zmaterializowanego wyniku i S - uruchomiono metodę.

Wokół tak zdefiniowanego HMM powstał rozbudowany aparat matematyczny, pozwalający wyznaczyć na podstawie historii operacji:

- prawdopodobieństwo wyemitowania kolejnego symbolu - jeżeli znamy historię operacji, oraz prawdopodobieństwa wyemitowania symbolu w każdym ze stanów, a także prawdopodobieństwa przechodzenia między stanami automatu reprezentującego dany proces - na przykład algorytmy prefiksowi i sufiksowy
- najbardziej prawdopodobny ciąg stanów, który doprowadził do wyemitowania danej historii operacji - czyli innymi słowy stan, w jakim aktualnie znajduje się proces - na przykład algorytm Viterbiego
- prawdopodobieństwa wyemitowania poszczególnych symboli w każdym ze stanów oraz prawdopodobieństwa przechodzenia pomiędzy stanami automatu reprezentującego proces - na przykład algorytm Bauma-Welcha

W eksperymencie będziemy starali się przewidywać, jaka operacja zostanie wykonana przez użytkownika aplikacji, jeżeli aktualnie wykonywaną operacją jest operacja U. Jeżeli będziemy sądzić, że następną operacją będzie operacja U - wówczas nie będziemy wykonywali automatycznej rematerializacji. Jeżeli przewidzimy, że następną operacją będzie odczyt, to wówczas automatycznie - wykonamy metodę, aby zmaterializować nowy wynik jej działania. Proces automatycznej rematerializacji może wykonać się natychmiast, bądź też zostać zaplanowany na później. Za automatyczną rematerializację wyniku powinna odpowiadać sama baza danych.

W przypadku podjęcia właściwej decyzji dotyczącej automatycznej rematerializacji - koszty ponoszone przez system (czas procesora, operacje I/O) powinien być



Rysunek 3. Automat procesu Markowa dla PMAP o maksymalnej sekwencji operacji U równej n

porównywalny do kosztu systemu bez predykcji - należy uwzględnić niewielki koszt działania systemu predykcji. Dodatkowe koszty pojawiają się jedynie w przypadku nietrafnych decyzji:

- W przypadku podjęcia decyzji o automatycznej rematerializacji, kiedy jednak okaże się, że następną operacją będzie kolejna operacja modyfikująca - kosztem błędu jest czas na uruchomienie metody i przygotowanie materializacji

- W przypadku podjęcia decyzji o nierematerializowaniu wyniku działania metody - kosztem błędnej decyzji, jest czas oczekiwania użytkownika na wynik.

Jako szczególny rodzaj HMM możemy też uznać rozwiązanie przedstawione w (Masewicz, 2006). W rozwiązaniu tym, nazywanym predykcją PMAP, predykcja następuje na podstawie częstości występowania ciągów operacji U o określonej długości. Częstości te są wyznaczane na podstawie historii operacji na danym obiekcie. Wówczas automat opisujący proces Markowa posiada stany symbolizujące ciągi operacji U o określonej długości - prawdopodobieństwo przejścia do kolejnego stanu (ciągu operacji U o długości o jeden większej) równe prawdopodobieństwu wystąpienia dłuższego ciągu operacji U (w przeciwnym wypadku - jeżeli wygenerowana jest operacja S - automat wraca do stanu zerowego - czyli ciąg U ma długość zero). Prawdopodobieństwa wygenerowania określonych symboli (U i S) dla każdego stanu odpowiadają prawdopodobieństwom przejścia do kolejnego stanu.

Opisany tu mechanizm predykcji żądań użytkowników PMAP jest o tyle cenny, że jak wykazano w (Masewicz, 2006) pomimo swej prostoty, a więc i małego narzutu obliczeniowego - daje bardzo dobre wyniki. Z tego też względu będzie on wykorzystywany do oceny jakości predykcji dokonywanej przy użyciu HMM

5. Opis przeprowadzonego eksperymentu

Aby udowodnić poprawność naszych rozważań dotyczących poprawy pracy systemu z zastosowaniem automatycznej rematerializacji sterowanej algorytmem predykcji opartym o aparat matematyczny związany z HMM, przeprowadziliśmy eksperyment, w którym porównaliśmy zachowanie mechanizmu materializacji z predykcją z trzema innymi rozwiązaniami:

- Materializacja hierarchiczna bez predykcji - jest to mechanizm najkorzystniejszy z punktu widzenia wykorzystania zasobów serwera. Materializacja następuje tutaj w chwili pierwszego odczytu wyniku metody (metoda jest wyliczana i przy okazji materializowany jest wynik).
- Materializacja, w której automatyczne rematerializacje następują zawsze po operacji modyfikującej obiekt. To rozwiązanie jest najlepsze z punktu widzenia użytkownika. Natomiast z punktu widzenia systemu oznacza to wiele rematerializacji, które nie zostaną ani razu wykorzystane (seria następujących po sobie operacji modyfikacji danych).
- Materializacja z prostym modelem predykcji - PMAP.

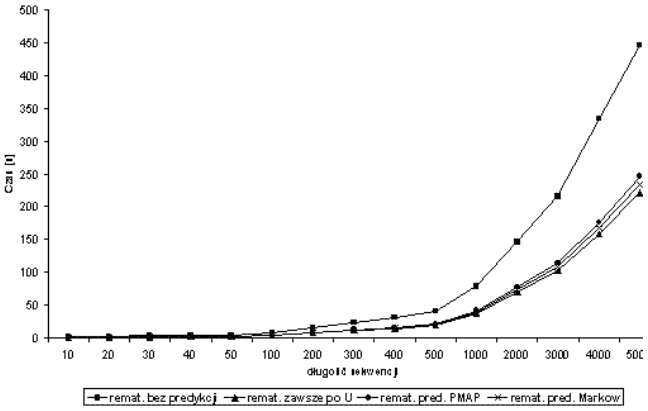
Celem eksperymentu jest wykazanie, że materializacja hierarchiczna wspierana algorytmami predykcji podejmującymi decyzje o dokonywaniu automatycznej rematerializacji jest podejściem, które z jednej strony pozwoli na osiągnąć całkowite obciążenie systemu, które będzie porównywalne lub niewiele gorsze od obciążenia systemu z materializacją hierarchiczną a z drugiej strony postaramy się zminimalizować czasy oczekiwania użytkowników na wyliczenie wyniku metody dążąc w tym przypadku do sytuacji zbliżonej do tej, jaka ma miejsce w systemach, które po każdej modyfikacji dokonują rematerializacji metody.

Jako parametr informujący o wykorzystaniu zasobów systemu przyjęliśmy czas procesora wykorzystany podczas trwania poszczególnych fragmentów programu testującego.

Obciążenie systemu zdefiniowaliśmy jako wszystkie działania podejmowane przez program (docelowo funkcjonalność ta powinna być zapewniona przez mechanizmy wbudowane w bazę danych) w celu obsługi materializacji, czyli sprawdzenie, czy materializacja jest dostępna, utworzenie materializacji, inwalidacja materializacji i wyliczenie wartości metody wtedy, kiedy było ono inicjowane przez algorytmy predykcji.

Jako czas użytkownika zdefiniowaliśmy czas oczekiwania użytkownika od momentu wysłania żądania pobrania wartości metody do chwili jej otrzymania - pomniejszony o czas trwania czynności systemowych. Czas ten był bliski 0 wtedy kiedy system znalazł zmaterializowaną wartość metody i większy od zera wtedy, kiedy należało wyliczać wartość metody.

W celu przeprowadzenia eksperymentu przygotowaliśmy komputer wyposażony w procesor AMD Athlon 2GHz i 512 MB pamięci operacyjnej. Zainstalowaliśmy na nim system operacyjny Windows XP. Testową bazą danych była baza FastObjects t7 9.0. Testowe dane miały łączny rozmiar około 2GB. Wszystkie metody zaimplementowane w testowych klasach wykonywały się przez ponad 1s.



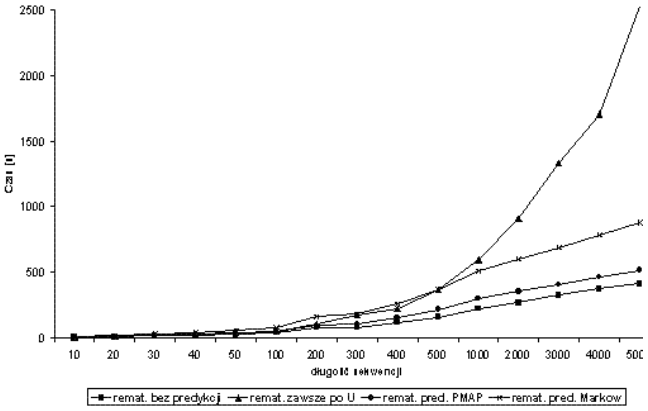
Rysunek 4. Czas oczekiwania użytkownika na zakończenie sekwencji operacji w pierwszym eksperymencie - liczba aplikacji 5, 25% operacji U

Eksperyment polegał na uruchamianiu przygotowanych sekwencji operacji i mierzeniu czasów wykonywania poszczególnych fragmentów programu. Przygotowane sekwencje operacji miały długość od 1 do 5000 operacji, przy czym jednym z parametrów napisanego przez nas generatora sekwencji był stosunek ilości operacji odczytu do ilości operacji zapisu (w kolejnych testach 10%, 25%, 50%, 75%, 90% operacji stanowiły operacje zapisu). Innym parametrem generatora ciągów operacji był stopień podobieństwa wygenerowanego ciągu do najbardziej prawdopodobnego sposobu działania zadanego automatu.

W pierwszym podejściu założyliśmy, że stanami automatu wykorzystywanego przez HMM będą wykorzystywane przez użytkowników aplikacje, które z pewnymi prawdopodobieństwami generują operacje U i S. Wygenerowane zbiory testowe symulowały działanie od 1 do 10 równoległych aplikacji. Dla potrzeb tego eksperymentu założyliśmy też, że znamy powiązania pomiędzy tymi aplikacjami, czyli prawdopodobieństwa zmiany stanów naszego automatu. Dzięki takiemu podejściu mogliśmy porównać jakość samej predykcji wykonywanej przy pomocy metody PMAP i algorytmów sufiksowych lub prefiksowych stosowanych gdy w HMM znamy wszystkie prawdopodobieństwa - zarówno dla stanów automatu, jak i dla generowanych symboli. Wynik tego eksperymentu przedstawiają rysunki 4 i 5. Na rysunku 4 przestawiony jest czas oczekiwania użytkownika na zakończenie wykonywania sekwencji operacji dla różnych strategii rematerializowania wyników działania metod.

Dla porównania - dla tego samego eksperymentu na rysunku 5 zaprezentuje czas pracy warstwy systemu zarządzania bazą danych, odpowiedzialnej za zarządzanie materializacją.

Podsumowując tą część eksperymentu można stwierdzić, że strategie z predykcją żądań użytkowników zawsze okazywały się lepsze z punktu widzenia użytkownika od strategii polegającej na materializacji hierarchicznej bez predykcji i gorsze



Rysunek 5. Czas pracy warstwy zarządzającej materializacja podczas wykonywania sekwencji operacji w pierwszym eksperymencie - liczba aplikacji 5, 25% operacji U

od strategii rematerializowania zawsze po operacji U. Jednocześnie strategia materializowania zawsze po operacji U okazała się najmniej efektywna z punktu widzenia systemu zarządzania bazą danych (zwłaszcza przy wzrastającym udziale operacji U w sekwencji wykonywanych operacji).

Ciekawie wygląda też porównanie jakości predykcji dokonywanej przy pomocy PMAP i HMM. PMAP miała przewagę dla małej liczby aplikacji (w naszym teście 1 do 4 aplikacji) i wtedy, kiedy stosunek liczby operacji U do operacji S był duży (w zależności od liczby aplikacji 25% lub 50% operacji U wystarczyło, aby przewagę uzyskiwał algorytm PMAP). Oznacza to, że dla jakości predykcji decydujące znaczenie ma ilość stanów automatu opisującego sekwencję operacji.

W kolejnych eksperymentach sprawdziliśmy zachowanie się algorytmów Viterbiego i Bauma-Welcha. Tutaj wyniki okazały się podobne do wyników eksperymentu pierwszego, przy czym nieznacznie wzrastało obciążenie warstwy systemowej przez predykcję dokonywaną przy pomocy HMM.

W kolejnej serii eksperymentów powróciliśmy do warunków z eksperymentu pierwszego - czyli wyznaczyliśmy automat i wszystkie prawdopodobieństwa dla procesu. Eksperymenty wykonaliśmy dla 10 współbieżnie działających aplikacji, przy czym w celu utrudnienia predykcji - wprowadzaliśmy aplikacje z losowym rozkładem operacji (od 1 do 10 aplikacji). Przy takich założeniach okazało się, że algorytm PMAP zdecydowanie lepiej radzi sobie z predykcją.

Wreszcie w ostatnich eksperymentach podobnie jak w poprzedniej serii wprowadzaliśmy kolejne aplikacje o losowym rozkładzie operacji, przy czym predykcja przy pomocy HMM opierała się tu podobnie jak w drugiej serii eksperymentów o algorytmy wyznaczające prawdopodobieństwa dla automatu opisującego proces. W tym wypadku dało się zauważyć przewagę predykcji opartej o HMM nad predykcją partą o mechanizm PMAP.

6. Podsumowanie

Wyniki przeprowadzonych eksperymentów potwierdziły, że ukryte modele Markowa mogą być bardzo dobrym mechanizmem predykcji żądań użytkowników a tym samym mogą służyć jako mechanizm automatycznego podejmowania decyzji, co do ewentualnego rematerializowania inwalidowanych właśnie materializacji. Eksperymenty pokazały też, że oba zaprezentowane mechanizmy predykcji żądań użytkowników wzajemnie się uzupełniają. Jednocześnie zaobserwowaliśmy, że każdorazowo mechanizm materializacji z predykcją żądań użytkowników (opartą o jeden lub drugi mechanizm) okazywał się być niewiele gorszym z punktu widzenia użytkownika od mechanizmu rematerializacji zawsze po inwalidowania zmaterializaowanego wyniku (mechanizm najgorszy z punktu widzenia systemu zarządzania bazą danych) i jednocześnie predykcja okazała się być zawsze niewiele gorsza z punktu widzenia obciążenia systemu od materializacji z rematerializacją dopiero po żądaniu użytkownika (najgorsza metoda z punktu widzenia użytkownika).

Literatura

Bertino, E. (1991) Method precomputation in object-oriented databases. *SIGOS Bulletin, 12 (2, 3), 1991*, 199-212.

Bębel, B. i Wrembel, R. (2002) Method Materialization Using the Hierarchical Technique: Experimental Evaluation *Proc. of Joint Conference on Knowledge-Based Software Engineering (JCKBSE), Slovenia*

Gopalkrishnan, V., Li, Q. i Karlapalem, K. (2000) Efficient Query Processing with Associated Horizontal Class Partitioning in an Object Relational Data Warehousing Environment *Proc. of Design and Management of Data Warehouses (DMDW), Sweden, 2000*

Huynh, T.N., Mangisengi, O. i Tjoa, A.M. (2000) Metadata for Object-Relational Data Warehouse *Proc. of Design and Management of Data Warehouses (DMDW), Sweden*

Jezierski, J., Masewicz, M. i Wrembel, R. (2004) On Optimising Data Access via Materialised Methods in Object-Oriented Systems *Proc. of Advances in Information Systems (ADVIS), Turkey*

Jezierski, J., Masewicz, M., Wrembel, R. i Czejdo, B.(2003) Designing Storage Structures for Management of Materialised Methods in Object-Oriented Databases *Proc. of Object-Oriented Information Systems (OOIS), Switzerland*

Jhingran, A. (1991) Precomputation in a Complex Object Environment *Proc of the IEEE Data Engineering Conference, Japan, pp. 652-659*

Kemper, A., Kilger, C. i Moerkotte, G. (1991) Function Materialization in Object Bases *Proc. of SIGMOD*

Kemper, A., Kilger, C. i Moerkotte, G. (1994) Function Materialization in Object Bases: Design, Realization, and Evaluation *IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 4*

Masewicz, M., Nyczkowski, B., Staniszewski, R. i Wrembel, R. (2005) Materializacja metod w obiektowych i obiektowo-relacyjnych bazach danych: implementacja prototypowego systemu *Proc. of KKNTPD 2005, Poznań Poland*

Masewicz, M., Wrembel, R. i Jezierski, J. (2005) Optimising Performance of Object-Oriented and Object-Relational Systems by Dynamic Method Materialisation *Proc. of ADBIS 2005, Tallin, Estonia*

Masewicz, M., Wrembel, R., Stabno, M. i Staniszewski, R. (2006) PMAP: Framework to Predicting Method Access Patterns for Materialized Methods *Proc. of International Conference on Advances in Information Systems (ADVIS), Turkey*

Morzy, T., Wrembel, R. i Koszlajda, T. (2000) Hierarchical materialisation of method results in object-oriented views *Proc. of ADBIS-DASFAA, Czech Republic*

Motschnig-Pitrik, R. (1996) Requirements and Comparison of View Mechanisms for Object-Oriented Databases *Information Systems, 21(3)*

(1995) Object Management Group. The Common Object Request Broker: Architecture and Specification.

Pugh, W. i Teitelbaum, T. (1989) Incremental Computation via Function Caching *Proc. of Principles of Programming Languages, USA*

Wrembel, R.(1998) Object-Oriented Views: Virtues and Limitations *Proc. of Intern. Symposium on Computer and Informatoin Sciences (ISCIS), Turkey*

HMM as prediction mechanism for object oriented database systems with hierarchical materialisation

In this paper we present a new prediction mechanism for hierarchical method materialisation in object oriented databases, based on Hidden Markov Models. Our contributions are twofold. First we present our modification of the stateoftheart PMAP prediction mechanism, and second, we present experimental evaluation of our algorithm, which proves its feasibility and benefit for query processing.