STUDIA INFORMATICA 2005
Volume 26 Number 1 (62)

Marcin GORAWSKI, Robert CHECHELSKI
Politechnika Śląska, Instytut Informatyki

ALGORYTM BALANSOWANIA PRZESTRZENNEJ TELEMETRYCZNEJ HURTOWNI DANYCH

Streszczenie. Równoważenie obciążenia systemów równoległych zapewnia minimalny czas realizacji zadań wysłanych do przetworzenia. W niniejszym artykule przedstawiono algorytm balansowania obciążeniem rozproszonej telemetrycznej hurtowni danych. Balansowanie jest wykonywane poprzez dobór rozmiaru zbioru danych ładowanych w każdym z węzłów.

Słowa kluczowe: balansowanie danymi, alokacja danych, fragmentacja danych, systemy równoległych baz danych, aR-drzewo, proces ETL

BALANCING ALGORITHM OF SPATIAL TELEMETRIC DATA WAREHOUSE

Summary. Balancing of parallel systems workload is very essential to ensure minimal response time of jobs submitted to process. In this paper an workload balancing algorithm of spatial telemetric data warehouse is presented. Balancing is performed by selection of data set size loaded into each node.

Keywords: data balancing, data allocation, data fragmentation, parallel databases systems, aR-tree, ETL process

1. Wprowadzenie

System przestrzennej telemetrycznej hurtowni danych (ang. *Spatial Telemetric Data Warehouse /STDW/)* zbiera dane telemetryczne o pomiarach zużycia mediów (gazu, prądu, wody, ciepła) [4].

Rozwiązaniem jest dwuwarstwowa infrastruktura telemetryczna, którą tworzy:

a) telemetryczny system zintegrowanego odczytu liczników,
b) system przestrzennej hurtowni danych telemetrycznych.

Telemetryczny system zintegrowanego odczytu liczników bazuje na technologii AMR (ang. *Automated Meter Reading*) oraz GSM/GPRS. System ten pozwala przesyłać dane z liczników zużycia mediów zlokalizowanych na dużym obszarze geograficznym do serwera telemetrycznego z wykorzystaniem sieci telefonii komórkowej GSM w technologii GPRS.

System przestrzennej telemetrycznej hurtowni danych jest systemem wspomagania podejmowania decyzji taktycznych o wielkości produkcji medium na podstawie krótkoterminowych prognoz ich zużycia. Prognozy te sporządzane są na podstawie analiz danych zgromadzonych w hurtowni danych zasilanej z serwera telemetrycznego. Serwer telemetryczny jest źródłem danych pomiarowych, które w procesie ekstrakcji są ładowane do bazy STDW.

1.1. Architektura przestrzennej telemetrycznej hurtowni danych

System STDW opisany jest modelem gwiazdy kaskadowej indeksowanej aR-drzewem na platformie Oracle9i/Java.

Architektura STDW jest strukturą rozproszoną z jednym klientem i kilkoma serwerami, umożliwiającą podniesienie wydajności poprzez równoległe wykonywanie zadań. W rozproszonej hurtowni STDW dane są przechowywane w wielu węzłach, z których każdy jest zarządzany przez osobny RDBMS Oracle 9i zdolny do pracy niezależnie od pozostałych.

STDW ma architekturę rozproszoną typu *shared nothing(SN)*, w której każdy węzeł posiada własny procesor i obszar pamięci oraz własny dysk, a komunikacja pomiędzy procesorami odbywa się wyłącznie poprzez połączenia sieciowe (przekazywanie komunikatów) w technologii RMI (ang. *Remote Method Invocation*). Technologia RMI pozwala obiektom znajdującym się na różnych komputerach w sieci komunikować się między sobą. Każdy obiekt znajdujący się na serwerze implementuje metody, które będą mogły być wywoływane przez aplikacje utworzone po stronie klienta. RMI przesyła przez sieć obiekty (łącznie z metodami i wartościami pól) za pośrednictwem protokołu TCP/IP.

Architektura *shared nothing* wymaga bardziej zaawansowanego zarządzania przy zapewnieniu praktycznie liniowego przyspieszenia wykonywanych operacji wraz ze wzrostem liczby procesorów i dysków [7].

Dane źródłowe dla systemu STDW są tak dzielone pomiędzy N węzłów, że każdy węzeł posiada ten sam schemat rozmieszczenia danych. Tabela faktów, zajmująca zwykle ponad 90% rozmiaru wszystkich tabel występujących w hurtowni, jest dzielona na N fragmentów o określonych rozmiarach. Tabele wymiarów są replikowane do wszystkich węzłów w takiej samej formie [2].

Pracę systemu STDW wykonanego architekturze *shared nothing* można podzielić na następujące fazy:

- faza rozproszonego ładowania – tabele wymiarów są replikowane do wszystkich węzłów, a wiersze tabeli faktów są rozprowadzane według przyjętego algorytmu podziału do wszystkich węzłów,
- faza dystrybuowania zapytania – zapytanie jest transformowane w N zapytań częściowych, które są wykonywane niezależnie w każdym z N węzłów; faza ta rozprowadza zapytanie w tej samej formie do wszystkich serwerów, ale niektóre typy zapytań wymagają zmiany formy dla poszczególnych węzłów,
- faza przetwarzania zapytania – zapytania częściowe są wykonywane równolegle we wszystkich węzłach; jest to jedna z głównych przyczyn przyspieszenia i lepszej skalowalności tej architektury,
- faza łączenia wyników – w fazie tej wyniki częściowe są zbierane ze wszystkich węzłów, łączone i dostarczane do użytkownika.

Rozproszone przetwarzanie wykonywane jest w każdym z węzłów wyłącznie na fragmencie danych w nim przechowywanych.

1.2. Potrzeba balansowania systemu STDW

Głównym celem rozproszenia STDW jest równoległe wykonywanie takich operacji, jak: ładowanie danych, tworzenie indeksów, przetwarzanie zapytań, odtwarzanie. Dane zgromadzone w hurtowni rozproszonej są przechowywane w kilku węzłach zarządzanych przez niezależny system zarządzania bazą danych.

Istnieje kilka metod uzyskiwania dobrej równoległości pracy systemów rozproszonych, np: kawałkowanie (fragmentacja), pliki gridowe (a także pliki produktu kartezjańskiego) lub drzewowo bazujące struktury danych. Drzewowo bazujące struktury są używane do deklasteryzacji zbioru tworzącego kostkę danych i podziału na kilka partycji przydzielanych do różnych węzłów. Pliki gridowe lub pliki produktu kartezjańskiego są silnymi i istotnymi technikami poprawiania wydajności systemu równoległego. Stopień równoległości przetwarzania jest zdeterminowany przez rozkład danych pomiędzy węzłami, nazywany **strategią alokacji danych** (ang. *Data Allocation Strategy*).

Celem balansowania obciążenia systemu rozproszonego jest minimalizacja średniego czasu odpowiedzi na zadania przezeń realizowane [12]. Innymi słowy, oznacza to zapewnienie odpowiedniego wkładu zasobów każdego węzła w równoległym wykonaniu tych zadań.

Dotychczasowe rozwiązania procesu balansowania obciążeniem systemu STDW koncentrowały się na przypadku, w którym system składał się z węzłów o tych samych parametrach bądź charakterystykach wydajnościowych. Istotą problemu, który jest poruszany w tym artykule, jest balansowanie obciążeniem systemu STDW (zwanego dalej w skrócie **B-**

STDW) zbudowanego w oparciu o komputery o różnych charakterystykach wydajnościowych. Różnice te dotyczą mocy obliczeniowej, rozmiaru pamięci operacyjnej oraz przepustowości podsystemu we/wy.

Dalsza część artykułu jest opracowana: sekcja 2 opisuje system STDW, sekcja 3 omawia tematykę balansowania obciążeniem w systemach rozproszonych i metody alokacji danych w kontekście wykorzystania HCAM, w sekcji 4 omówiony jest algorytm balansowania systemu B_STDW, sekcja 5 i 6 przedstawia wybrane aspekty implementacji systemu B-STDW i analizę wyników uzyskanych metodą eksperymentalną.

2. System przestrzennej telemetrycznej hurtowni danych

System STDW bazuje na rozproszonej przestrzennej hurtowni danych (DSDW) rozwijanej w Pracowni Hurtowni Danych i Systemów Inteligentnych Zakładu Teorii Informatyki Instytutu Informatyki Politechniki Śląskiej [5].

Zasadnicze cechy DSDW to:

- architektura SN – bazy danych Oracle 9i/Oracle 10g są zainstalowane w poszczególnych komputerach klasy PC,
- modelowanie danych w schemacie gwiazdy kaskadowej,
- obsługa danych przestrzennych (wymiary przestrzenne i nieprzestrzenne, miary tylko numeryczne)[3],
- indeksowanie przestrzenne metodą aR-drzewa,
- środowisko rozwojowe i aplikacyjne na platformie Java.

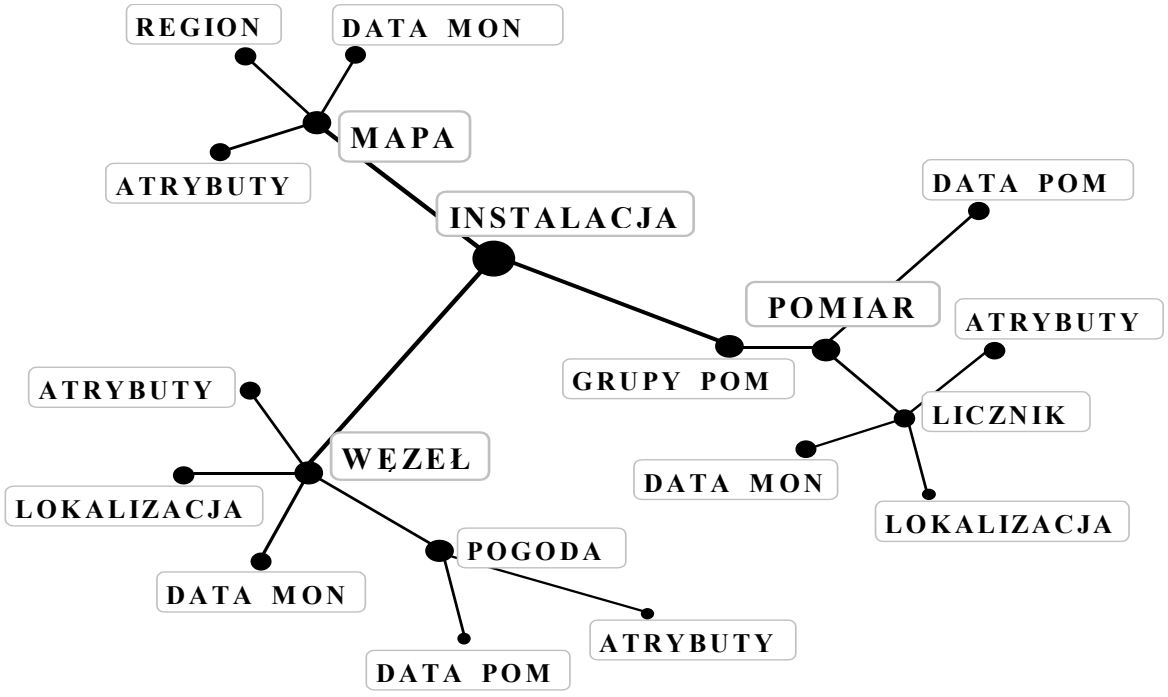
DSDW przechowuje dane dotyczące pomiarów zużycia różnych mediów: wody, gazu, energii elektrycznej, zamodelowanych gwiazdą kaskadową o schemacie jak na rys. 1. Główną tabelą faktów jest tabela INSTALACJA, która łączy ze sobą główne wymiary – WĘZEŁ, MAPA i POMIAR. Wymiar WĘZEŁ to informacje o węzłach zbiorczych instalacji, MAPA zawiera informacje o mapach obszaru obejmowanego przez węzeł, a POMIAR zawiera dane pomiarowe z liczników.

Analizy, które można obecnie wykonać z użyciem DSDW, skupiają się na wymiarze POMIAR. Fakty w nim przechowywane posiadają wymiar przestrzenny określony jako LICZNIK/LOKALIZACJA oraz czasowy – DATA POM. Wymiar przestrzenny składa się z trzech atrybutów przestrzeni: X, Y, Z i opisuje lokalizację licznika w terenie. Wymiar czasowy składa się z daty wykonania pomiaru.

DSDW obsługuje kilka typów liczników. Każdy typ posiada inną granulację faktów w wymiarze czasowym, rozciągającą się od kilkunastu minut do kilku godzin, co sprawia, że rozmiary pomiarów różnych liczników różnią się w swej liczności.

STDW obsługuje zapytania przestrzenne o wartości zużycia mediów w zadanym regionie. Zapytanie dotyczy jednego lub kilku obszarów w kształcie prostokątów, w których poszukiwana jest suma wartości zużycia mediów dla ostatniego okresu pomiaru. Dodatkowo możliwe jest zapytanie o wartości pomiarów dla określonego przedziału czasu.

Aby móc wydajnie i szybko odpowiadać na zadane przez użytkownika zapytania, zastosowano metodę przestrzennego indeksowania znaną jako agregacyjne R-drzewo (aR--drzewo). Drzewo agregatów jest dynamiczną strukturą danych przechowywaną w pamięci operacyjnej i stanowi przestrzenny indeks danych. Zawiera kilka poziomów agregacji, przez co znacznie usprawnia proces wyznaczania odpowiedzi na zapytania. Drzewo składa się z węzłów zawierających zagregowane wartości pochodzące z fragmentów regionu. Węzły są obiektami wartości zagregowanych wraz ze współrzędnymi obejmowanego regionu. Najniższy poziom drzewa jest tworzony bezpośrednio na podstawie danych szczegółowych czerpanych z bazy hurtowni danych. Kolejne poziomy są budowane w wyniku zbierania informacji z jednostek agregacji na niższym poziomie i agregowane w obiektach bieżącego poziomu. Regiony jednostek umieszczonych na wyższym poziomie obejmują regiony jednostek agregacji poziomu niższego. W ten sposób tworzony jest każdy poziom drzewa aż do korzenia zbierającego wszystkie agregaty i obejmującego cały region.



Rys. 1. Schemat modelu danych zastosowanego w STDW
Fig. 1. A data model schema used in STDW

Oprogramowanie drzewa agregatów w STDW obsługuje mechanizm pamięci wirtualnej. Umożliwia on budowę drzew o dowolnie dużych rozmiarach, ograniczonych jedynie pojemnością dostępnej pamięci masowej. Idea rozwiązania polega na zapisywaniu nieużywanych

węzłów drzewa do bazy danych i ewentualne późniejsze ich wykorzystanie, jeśli zajdzie taka potrzeba. W tym celu wykorzystywana jest specjalna tabela MATNODE, zawierająca zagregowane węzły. Zapis do tej tabeli odbywa się w momencie potrzeby zwolnienia pamięci operacyjnej dla nowo tworzonych węzłów lub gdy następuje kończenie pracy hurtowni danych [4].

3. Istota balansowania systemu STDW

W celu zwiększenia całkowitej wydajności systemu STDW w procesie balansowania minimalizuje się średni czas odpowiedzi na zapytania testowe wysłane do przetworzenia. Średni czas odpowiedzi jest definiowany jako średni interwał pomiędzy momentem, gdy zapytanie zostało wysłane do hurtowni, a momentem, w którym otrzymano odpowiedź (po przetworzeniu).

W systemie STDW każdy węzeł realizuje odpowiedź na to samo zapytanie na swojej kopii danych tak, aby czas pracy każdego z nich był podobny. W przypadku systemów hurtowni danych opartych na komputerach o tych samych parametrach balansowanie przekłada się na zapewnienie równego obciążenia każdego z dysków (tutaj wydajność dysku jest „wąskim gardłem").

W przeciwnym razie w procesie balansowania STDW należy użyć odpowiedniego schematu alokacji danych**,** którego głównym zadaniem jest zapewnienie właściwego udziału każdego węzła w realizacji odpowiedzi na różne typy zapytań.

3.1. Schematy alokacji danych

Aby podzielić zbiór danych na partycje ładowane do poszczególnych węzłów systemu należy najpierw wybrać podzbiór wymiarów tworzących kostkę danych (ang. datacube), który utworzy tzw. schemat partycjonowania danych. Schemat ten określa sposób podziału zbioru według wybranych atrybutów.

Pierwszym krokiem tworzenia schematu jest utworzenie bloków gridowych. W tym celu dziedzina każdego z wymiarów schematu jest dzielona na N_i przedziałów. Dziedzina bloków gridowych jest produktem złożenia wszystkich przedziałów.

Utworzone w ten sposób bloki (będące w istocie hiperprostokątami określającymi przedziały wymiarów tworzących schemat partycjonowania) określają sposób podziału oryginalnego zbioru danych. Bloki te są następnie przydzielane do węzłów przetwarzających zależnie od przyjętego *schematu alokacji danych*. Każdy blok zawierający podzbiór danych kostki o określonych wartościach poszczególnych atrybutów schematu partycjonowania tworzy tzw. plik produktu kartezjańskiego. Obok plików produktu kartezjańskiego istnieją

pliki gridowe, które ogólnie różnią się tym, że jednen plik gridowy może zawierać kilka plików kartezjańskich. Zatem jeden plik gridowy obejmuje kilka wartości atrybutów schematu partycjonowania.

Istnieje kilka schematów alokacji danych, zaprojektowanych do różnych zastosowań. Część z nich posiada restrykcje, które uniemożliwiają bądź poważnie ograniczają ich stosowanie w środowiskach hurtowni danych. Poniżej zaprezentowano kilka najpopularniejszych schematów wraz z krótkim opisem oraz krytyką:

- Disk modulo – schemat stosowany do zapytań zadanych zbiorem predykatów dla części wymiarów (tzw. partial match queries),
- Field-wise Exclusive-or, Error Correcting Codes – schematy wykorzystujące operacje bitowe; są bardzo restrykcyjne, liczba węzłów musi być potęgą liczby dwa. W przypadku drugiego schematu dodatkowo liczność atrybutów również musi być potęgą dwójki,
- Hilbert Curves Allocation Method (HCAM) – schemat wykorzystujący krzywe Hilberta; osiąga dobre wyniki z zapytaniami przedziałowymi (ang. *range queries*),
- Vector Based Declustering – schemat zaprojektowany do przetwarzania obrazów, ponadto stosowany w kartograficznych bazach danych. Jego wadą jest wymóg, aby dziedzina każdego z atrybutów była dzielona na tę samą liczbę przedziałów. Ograniczony do dwu wymiarów,
- Cycling Allocation – ogólna klasa schematów przydziału cyklicznego, wszystkie ograniczone do dwu wymiarów,
- General Multidimensional Data Allocation (GeMDA) – metoda zbliżona do Disk Modulo oraz HCAM, nie faworyzuje jednak żadnego typu zapytań.

Więcej szczegółów można znaleźć w przeglądzie schematów zawartym w [9].

3.2. Alokacja danych w STDW

W systemie STDW najistotniejsze są dane podsumowujące. W tym celu do systemu dodano aR-drzewo wraz z obsługą mechanizmu pamięci wirtualnej. Powoduje ono, że agregaty są wyznaczane na bieżąco podczas wykonywania zapytania (jeśli nie zostały wyznaczone w odpowiedzi na wcześniejsze zapytania). Zatem to zapytanie określa, które dane szczegółowe zostaną pobrane do agregacji. Kolejną zaletą tej struktury indeksowej jest to, że czas pobierania danych już zagregowanych jest bardzo krótki, pomijalnie mały w stosunku do czasu agregacji, gdy pobieranych jest bardzo dużo danych szczegółowych z bazy hurtowni [5].

Istotny jest również fakt różnej liczności pomiarów liczników zgromadzonych w bazie hurtowni, który wynika z różnego stopnia granulacji w wymiarze czasowym.

Zważając na przytoczone cechy systemu STDW oraz jego balansowania przy-jęto użyć schemat alokacji danych tabeli faktów, dający dobrą równoległość odpowiedzi na obecnie obsługiwane zapytania zakresowe i dobrać tak rozmiar danych składowanych w węźle, aby zapewnić zbliżony średni czas pracy każdego węzła. Ponadto tabela faktów ma być partycjonowana w taki sposób, aby zbiór pomiarów dla każdego licznika miał taki sam rozmiar.

Jako metodę alokacji wybrano HCAM ze względu na jej odpowiedniość dla zapytań zakresowych. Metoda ta bazuje na idei krzywych wypełniania przestrzeni. Cechą charakterystyczną tej krzywej jest to, że przechodzi ona przez dany punkt przestrzeni dokładnie raz i nigdy się nie krzyżuje [11]. Może ona zostać użyta do uzyskania liniowego uporządkowania bloków gridowych schematu partycjonowania danych. W tym schemacie alokacji krzywa Hilberta jest używana do wybierania kolejnych bloków gridowych, które są następnie przydzielane do węzłów sposobem round-robin. Pomysł polega na tym, że dwa bloki umieszczone blisko siebie w przestrzeni liniowej powinny być również blisko siebie w k-wymiarowej przestrzeni bloków gridowych, a tym samym powinny być rozmieszczone w innych węzłach, by zapewnić większą równoległość przetwarzania zapytania.

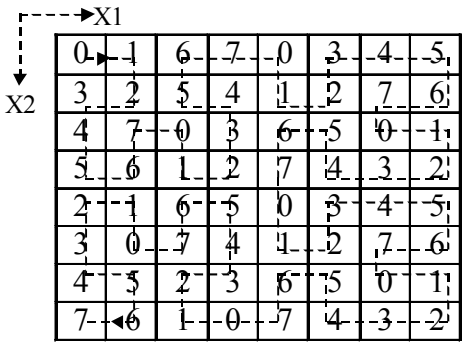
Ogólna funkcja wykonująca odwzorowanie pomiędzy węzłami i blokami d-wymiarowej przestrzeni o współrzędnych (X_1, ..., X_d) ma postać:

$$HCAM(X_1, X_2, ..., X_d) = H(X_1, X_2, ..., X_d) \bmod N, \quad (1)$$

gdzie: *H( )* jest funkcją wyznaczenia h-wartości odwzorowujących przestrzeń d-wymiarową w przestrzeń liniową z użyciem krzywej Hilberta, natomiast *N* oznacza liczbę dostępnych węzłów.

Rysunek 2 przedstawia przykład schematu HCAM dla bloków gridowych przestrzeni dwuwymiarowej, utworzonych przez podział każdego z wymiarów na 8 przedziałów; bloki przydzielane są do 8 węzłów. Numery zawarte w kratkach określają numer węzła, do którego zostanie przydzielony dany blok. Dodatkowo na rysunku zaznaczono krzywą Hilberta przebiegającą od lewego górnego do lewego dolnego rogu zaprezentowanej przestrzeni.

Dla zapytań o małym rozmiarze i przy dużej liczbie węzłów metoda HCAM pracuje podobnie lub lepiej w stosunku do metod Error Correcting Codes, Field-wise Exclusive-or, Disk Modulo. Dla dużych zapytań lub mniejszej liczby węzłów wszystkie wymienione metody osiągają zbliżone wyniki, co jest spowodowane podobnym rozłożeniem danych na mniejszej liczbie dysków, tym samym dając zbliżone czasy odpowiedzi na zapytania [8].



Rys. 2. Przykład schematu HCAM
Fig. 2. An HCAM example

4. Algorytm balansowania systemu STDW

Wybór metody alokacji HCAM był podyktowany zarówno ze względu na jej odpowiedniość dla zapytań zakresowych, jak i wykorzystaniem krzywych Hilberta [1] jako efektywnego narzędzia porządkowania. Zaprezentowane w [1] podejście to paskowanie danych uporządkowanych krzywymi Hilberta na styl Round-Robin, tak aby następujące po sobie rekordy były przesyłane do kolejnych węzłów. Uzasadnieniem dla tego wzorca paskowania było duże zwiększenie prawdopodobieństwa, iż przestrzeń ograniczana przez hiperprostopadłościan zapytania użytkownika będzie równomiernie rozproszona pomiędzy N węzłów.

Aby dobrać rozmiary zbiorów ładowanych do poszczególnych węzłów, należy wyznaczyć ich wydajność. Dokonuje się tego poprzez realizację tej samej agregacji na zbiorze testowym przez wszystkie węzły. Miarą wydajności jest uzyskany czas agregacji. Zbiór testowy (używany do balansowania) jest podzbiorem tabeli faktów. Jest on definiowany przez użytkownika jako zakres numerów liczników, których pomiary będą ładowane.

Algorytm używa pojęcia *współczynnika podziału tabeli faktów* - p_i. Jest to wartość określająca procentowy udział rozmiaru podzbioru tabeli faktów przydzielonej do węzła w stosunku do rozmiaru całej tabeli faktów. Wykorzystując uzyskane czasy agregacji zbioru testowego przez każdy węzeł, algorytm iteracyjnie dobiera wartości współczynników p_i, tak by uzyskać zbliżony średni czas pracy węzła.

Szkic algorytmu balansowania

1. Ładowanie tabel wymiarów do wszystkich węzłów
2. Współczynniki podziału tabeli faktów $p_i = 1/N$

3. Ładowanie *testowego zbioru* tabeli faktów podzielonego wg współczynników podziału p_i do wszystkich węzłów
4. Agregacja wszystkich danych zbioru testowego
5. Maksymalne niezbalansowanie mniejsze od zakładanego? Tak – idź do **7**, Nie – idź do **6**
6. Modyfikacja współczynników podziału p_i na podstawie czasów agregacji, idź do **3**
7. Ładowanie *zbioru roboczego* tabeli faktów podzielonej wg współczynników podziału p_i do wszystkich węzłów

Pierwszym krokiem algorytmu jest załadowanie tabel wymiarów do wszystkich komputerów – każdy węzeł dostaje tę samą kopię tabel wymiarów. Następnie współczynniki p_i dla każdego węzła są ustawiane na wartość 1/N, gdzie N oznacza liczbę węzłów. Kolejnym krokiem jest wyznaczenie planu podziału tabeli faktów (pierwsza część kroku 3 algorytmu balansowania).

Tworzenie planu podziału tabeli faktów

1. Dla lokalizacji każdego licznika obliczane są jej h-wartość oraz liczba pomiarów.
2. Liczniki są sortowane narastająco według h-wartości.
3. Pomiary każdego z liczników są dzielone na przedziały o rozmiarze zadanym przez użytkownika (rozmiar fragmentu) i układane w ciąg postaci: <fragment pomiarów licznika1>, <fragmenty pomiarów licznika2> (kolejność liczników według pkt. 2).
4. Przedziały są rozmieszczane w kolejnych węzłach metodą Round-Robin z przeskokami, aby zachować wartość współczynnika podziału dla *i*-tego węzła:
   4.1. Jeśli rozmiar obecnie przydzielonego podzbioru tabeli faktów dla węzła i w stosunku do sumy rozmiarów podzbiorów tabeli faktów we wszystkich węzłach jest mniejszy od współczynnika p_i, to fragment zostaje przydzielony do węzła i.
   4.2. W przeciwnym przypadku fragment jest sprawdzany dla węzła i+1.
   4.3. Jeśli fragment nie zostanie przydzielony do żadnego z węzłów, wyznaczany jest węzeł, dla którego przydzielenie spowoduje najmniejsze przekroczenie jego współczynnika p_i.
   4.4. Kolejny fragment jest sprawdzany dla węzła o numerze 0.

Algorytm balansowania rozsyła tabelę faktów do wszystkich węzłów, wykorzystując plan podziału stworzony według podanego schematu. Krok ten realizuje podział całej tabeli faktów na partycje ładowane do węzłów, wykorzystując metodę HCAM, dodatkowo uwzględniając wyznaczone rozmiary zbiorów dla każdego z węzłów oraz fragmentowanie.

Następnie na załadowanych danych zostaje wykonana agregacja zapytaniem podanym przez użytkownika. Zapytanie jest takie samo dla wszystkich iteracji algorytmu, przed jego wykonaniem aR-drzewa są usuwane z serwerów, zatem mierzony jest czas budowy drzewa.

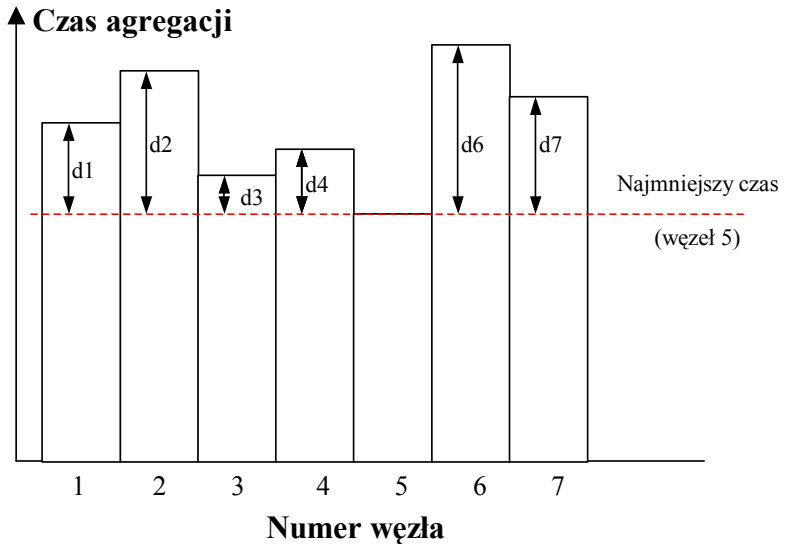
Na podstawie uzyskanych czasów zostają wyznaczone niezbalansowania każdego węzła w systemie w stosunku do najszybszego, według następującego wzoru:

$$imbalance_i = \frac{T_i - T_{fast}}{T_{fast}} = \frac{d_i}{T_{fast}}, \tag{2}$$

gdzie: T_i i T_{fast} to odpowiednio czasy agregacji węzła i oraz węzła najszybszego.

Na rysunku 3. zilustrowano sposób określania niezbalansowania każdego z węzłów. Węzłem najszybszym jest węzeł 5, a każdy z czynników d_i wynosi T_i - T_5.

Jeśli maksymalne niezbalansowanie jest większe od zadanego dopuszczalnego, następuje korekcja współczynników podziału tabeli faktów – p_i.



Rys. 3. Metoda wyznaczania niezbalansowania węzłów systemu B-STDW
Fig. 3. Calculation method of B-STDW system nodes imbalance

Algorytm korekcji, poza czasami agregacji, przyjmuje dwa współczynniki corrP (zwiększanie wartości współczynnika podziału) oraz corrN (zmniejszanie wartości współczynnika).

Algorytm korekcji współczynników p_i

1. Wyznaczony zostaje średni czas agregacji wszystkich węzłów.
2. Dla każdego węzła obliczane jest niezbalansowanie czasu agregacji do średniej, które następnie jest wykorzystane do korekcji współczynnika p_i. Niezbalansowanie jest wyznaczane według wzoru:

$$imb = \frac{T_i - avg}{avg} \tag{3}$$

2.1. Jeśli $imb > 0$, wtedy korekcja wygląda następująco:

$$p_i = p_i \cdot (1 - corrN \cdot imb) \tag{4}$$

2.2. Jeśli $imb < 0$, wtedy wykonywana jest korekcja:

$$p_i = p_i \cdot (1 - corrP \cdot imb) \tag{5}$$

2.3. Jeśli $imb = 0$, wtedy współczynnik nie jest korygowany.

3. Wszystkie współczynniki są korygowane do 100%.

W przypadku, gdy maksymalne niezbalansowanie jest mniejsze od zadanego, następuje ładowanie pełnej tabeli faktów (zbiór roboczy) dla wyznaczonych podczas balansowania współczynników p_i (analogicznie wykonując planowanie zaprezentowanym wcześniej algorytmem i rozsyłając zbiór).

5. Implementacja B-STDW

5.1. Specyfikacja wewnętrzna

System B-STDW bazuje na dwóch wcześniej utworzonych systemach: STDW oraz ETL[6]. Aby zaimplementować zaprezentowany algorytm balansowania, projekt podzielono na trzy środowiska: *Interfejs do systemu STDW*, *Zbiór komponentów ETL*, *Zarządca*. Do wyznaczania wartości funkcji odwzorowującej przestrzeń n-wymiarową w liniową z użyciem krzywych Hilberta wykorzystano kod z [10].

Operacje wspomagane przez *interfejs STDW* to: podłączenie do wszystkich serwerów określonych w liście serwerów, rozłączenie z wszystkimi serwerami, wykonanie operacji agregacji danych w obszarze objętym przez listę okien zapisanych w pliku, utworzenie aR-drzew na serwerach, usunięcie aR-drzew z serwerów, pobranie czasów agregacji, wykonanie polecenia DDL.

Środowisko ETL ma budowę komponentową, gdzie kod każdego komponentu jest wykonywany jako osobny wątek, stąd problemy ze sterowaniem pracą środowiska. Aby umożliwić synchronizację oraz wymianę informacji pomiędzy poszczególnymi komponentami systemu, utworzono klasę Container, zawierającą wyłącznie statyczne pola. Każde pole może być dostępne bezpośrednio (bez synchronizacji), gdyż tylko jeden komponent może dokonywać zapisu składowej i tylko jeden wykonuje odczyt.

Zarządca obok STDW i ETL jest trzecim środowiskiem systemu B-STDW odpowiedzialnym za nadzorowanie pracy obydwu. Jego zadaniem jest wykonywanie operacji

mających na celu przygotowanie systemu do pracy (inicjacja obydwu środowisk) oraz sterowanie ich działaniem, tak aby zrealizować operacje zadane przez użytkownika. System może pracować w dwu trybach: testowym i nietestowym. Tryb testowy jest używany w przypadku, gdy system B-STDW jest zbalansowany. W trybie tym nie są uruchamiane procesy ETL, serwery STDW pracują na już załadowanych danych, dopuszczalne operacje to *Blok Testów*, *Test*, *Rekonfiguracja*. W trybie nietestowym możliwe są wszystkie operacje.

Operacja Balansowanie ma celu zrównoważenie wszystkich węzłów tworzących B-STDW. Implementuje ona przytoczony algorytm balansowania. Operacja ma postać *n* iteracji składających się z ładowania danych (zbiór testowy) o rozmiarach proporcjonalnych do wartości współczynników podziału tabeli faktów, wykonania agregacji na całym załadowanym zbiorze, korekcji współczynników podziału odpowiednio do niezbalansowania poszczególnych węzłów.

Operacja Test polega na wykonaniu agregacji danych w obszarze objętym przez listę okien zapisanych w pliku (wykorzystano kod standardowego klienta STDW). Agregacja może być powtórzona określoną liczbę razy.

Rekonfiguracja jest operacją mającą zastosowanie wyłącznie w testach eksperymentalnych. Umożliwia wykorzystanie podzbioru węzłów spośród wszystkich dostępnych w B-STDW. Wybrane węzły będą brały udział w dalszej części testów aż do wystąpienia kolejnej operacji rekonfiguracji ustalającej ich nowy zbiór.

Blok Testu wykonuje operacje objęte blokiem zadaną ilość razy.

5.2. Specyfikacja zewnętrzna

System ma formę aplikacji konsolowej, interakcja z użytkownikiem to praktycznie oczekiwanie na naciśnięcie klawisza *„Enter”* po włączeniu serwerów.

Konfiguracja i obsługa zadań odbywa się poprzez pliki tekstowe. Wyniki są prezentowane w postaci 3 strumieni, które mogą być przekierowywane np. do konsol lub plików. Pliki listy serwerów oraz pliki konfiguracji klienta systemu STDW pozostały niezmienione. Podsystem ETL jest uruchamiany w dwu konfiguracjach – ładującej tabele wymiarów (poza ładowaniem tabeli wymiarów zostają zebrane informacje wykorzystywane przy podziale tabeli faktów) oraz ładującej tabelę faktów – stąd dwa pliki konfiguracyjne.

Plik konfiguracyjny systemu raportowania określa między innymi sposób formatowania komunikatów w strumieniach oraz sposób ich prezentacji (plik, konsola).

Plik konfiguracyjny Zarządcy ma formę dokumentu XML, zawierającego program do wykonania.

Nowo utworzonymi plikami są: pliki konfiguracyjne dla ETL (w których wykorzystano zaimplementowane komponenty), plik konfiguracyjny dla zarządcy oraz systemu raportowania. Pliki konfiguracyjne dla systemu DSDW pozostały w niezmienionej formie.

Wyniki pracy systemu B-STDW, dzięki zastosowanej technologii raportowania, są kierowane do strumieni, które z kolei mogą być przekierowane do wybranych przez użytkownika miejsc docelowych.

5.2.1. Komunikaty systemowe

System posiada trzy typy komunikatów: komunikaty Zarządcy, komunikaty klienta systemu STDW oraz komunikaty systemu ETL. Wszystkie komunikaty pojawiają się na komputerze pełniącym rolę zarządcy/klienta.

Zarządca posiada trzy niezależne strumienie komunikatów: *measurements*, *system*, *counters*. Strumień *measurements* jest związany z wynikami (pomiarami) działania systemu, zawiera informacje nt. czasów agregacji, niezbalansowania, współczynników p_i, itd. Strumień *system* zawiera informacje związane bezpośrednio z działaniem systemu o wykonywanych operacjach, jest to strumień diagnostyczny. Strumień *counters* niesie informacje nt. planowania i rozmieszczania wyników pomiarów oraz zawiera plany rozmieszczenia pomiarów wszystkich liczników dla każdej iteracji balansowania. Każde zdarzenie zawarte w strumieniu jest opatrzone czasem wystąpienia zawierającym dzień i godzinę z dokładnością do milisekund.

Użyta technologia raportowania umożliwia łatwe przekierowanie komunikatów zarówno do konsoli, jak i do pliku. Domyślnie informacje ze strumieni *measures* i *system* są kierowane do konsoli oraz do pliku, natomiast *counters* tylko do pliku. Pliki tych strumieni to odpowiednio: log/measurements.log, log/system.log, log/counters.log.

Poza informacjami ze strumieni Zarządcy do konsoli kierowane są standardowe komunikaty systemu ETL oraz komunikaty klienta systemu DSDW. W dalszej części zostaną opisane tylko komunikaty Zarządcy.

5.2.2. Strumień measures

Strumień niesie informacje o pomiarach pracy systemu. Zawiera opis każdej wykonywanej operacji programu oraz wszystkie pomiary z nią związane. Operacja balansowania jest raportowana następująco:

```
2004-07-21 23:37:30,765 Balansowanie systemu - parametry: iteracji: 15 max. niezb.: 0.1
zapytanie: bal_3wf.win liczniki: 2000 - 2133 typ wspolczyn.: 4 rozmiar fragm.: 5000 timeout:
4000
2004-07-21 23:37:30,765 balance: przygotowanie do balansowania
2004-07-21 23:37:30,781 Rzeczywiste wspolcz.: 0.1663368  0.16673942         0.16665997
         0.166704  0.16666311         0.1668967
2004-07-21 23:37:30,781 Odchylenie: 4.1045828835469753E-4
2004-07-21 23:37:30,781 Interwencji: 0
2004-07-22 00:11:45,125 balance: czasy pracy - 923422    1396562   1405500   957094    1637218
         1565766
2004-07-22 00:11:45,125 Niezbalansowania:  0.0  0.5123768 0.522056  0.036464367
         0.77299005         0.6956126
2004-07-22 00:11:45,125 balance: max. niezbalansowanie - 0.77299005
2004-07-22 00:11:45,125 balance: iteracja 1
2004-07-22 00:11:45,125 Wyznaczone wspolcz.: 0.20098083  0.16400838         0.16281846
         0.19873948         0.13197029         0.14148256
```

```
2004-07-22 00:11:45,140 Rzeczywiste wspolcz.: 0.20102146 0.16423613         0.1642897
         0.19781609         0.13079835         0.14183825
2004-07-22 00:11:45,140 Odchylenie: 0.0021379083192081282
2004-07-22 00:11:45,140 Interwencji: 0
2004-07-22 00:23:10,328 balance: czasy pracy - 298844    411703    408688    318125    464719
         455094
2004-07-22 00:23:10,328 Niezbalansowania:  0.0  0.3776519 0.367563  0.06451861
         0.5550555 0.52284807
2004-07-22 00:23:10,328 balance: max. niezbalansowanie - 0.5550555
2004-07-22 00:23:10,328 balance: iteracja 2
                  ...
                  ...
2004-07-22 01:35:07,250 balance: niezbalansowanie - 0.051491294
2004-07-22 01:35:07,250 Rzeczywiste wspolcz.: 0.2959062  0.14660741         0.14388722
         0.27503946         0.06444452         0.07411518
2004-07-22 01:35:07,250 Odchylenie: 4.740963486940657E-4
2004-07-22 01:35:07,250 Interwencji: 0
2004-07-22 02:29:57,000 balance: ------ balansowanie zakonczone ------
```

Na początku raportu występuje nagłówek określający operację balansowania. Następnie raportowane są:

- rzeczywiste współczynniki podziału (wynikające z fragmentacji zbioru),
- ich odchylenie od wyznaczonych przez algorytm współczynników p_i,
- liczba interwencyjnych przydziałów licznika do węzła.

Po wykonaniu agregacji system raportuje:

- czasy pracy wszystkich węzłów w kolejności wpisów z pliku listy serwerów STDW,
- niezbalansowania wszystkich węzłów w odniesieniu do najszybszego (najszybszy węzeł posiada niezbalansowanie równe zero),
- maksymalne niezbalansowanie.

Jeśli niezbalansowanie jest powyżej dopuszczalnego i nie przekroczono maksymalnej liczby iteracji, następuje kolejna iteracja, której numer jest raportowany do strumienia.

Operacja testu jest raportowana następująco:

```
2004-07-22 02:29:57,031 test: ------ test "balans.1 test.1" rozpoczety ------
2004-07-22 02:29:57,031 test - "balans.1 test.1" - iteracja 1
2004-07-22 02:42:55,062 Czasy pracy: 683016    688922    720797    713906    743594    652453

2004-07-22 02:42:55,062 Niezbalansowania:  0.04684322    0.055895213        0.10474931
         0.094187625        0.13968976         0.0
2004-07-22 02:42:55,062 test: ------ test "balans.1 test.1" zakonczony ------
```

Po nagłówku operacji podawane są: numer iteracji, czasy pracy wszystkich węzłów, niezbalansowania (identycznie jak przy balansowaniu).

Operacja rekonfiguracji podaje numery węzłów biorących udział w dalszej pracy systemu:

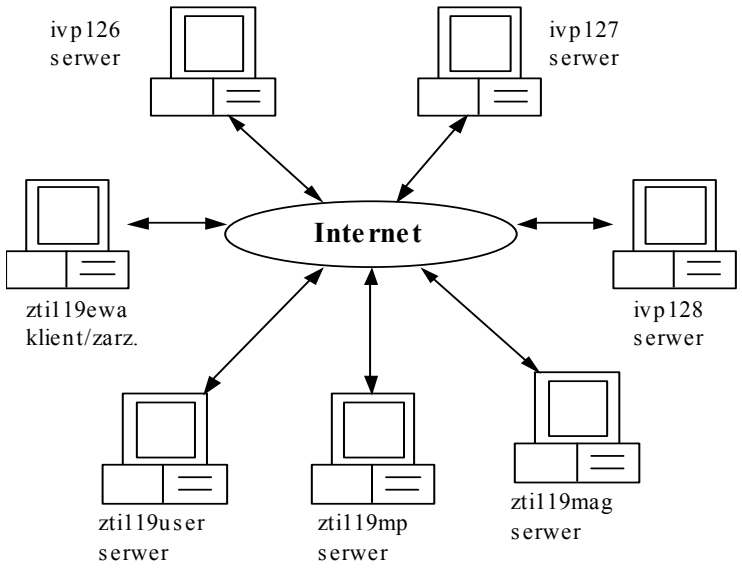
```
2004-07-25 08:22:28,828 ------Rekonfiguracja maszyn - biezace maszyny: 0 1 2 3 4
```

6. Testy wydajnościowe

Testowany system B-STDW charakteryzował się tym, że:

- składał się z 7 komputerów, z czego 6 pracowało jako serwery B-STDW, a jeden jako klient/Zarządca,
- dane źródłowe były przechowywane na komputerze-Zarządcy, który realizował operacje programu testów, w tym: balansowanie (ładowanie tabel wymiarów, podział i ładowanie tabeli faktów, wyznaczanie niezbalansowania), rozproszone wykonywanie agregacji danych w obszarze wyznaczonym przez listę okien zapisanych w pliku, rekonfiguracja systemu i inne,
- komputery serwery wykonywały operacje zlecone przez Zarządcę na ich kopii danych.

Architekturę testowanego systemu B-STDW przedstawiono na rys. 4.



Rys. 4. Architektura testowanego systemu STDW
Fig. 4. Tested system STDW architecture

Charakterystyki użytkowe komputerów z rys. 4.:

- komputery zti119mag, zti119ewa, ivp126: Intel Pentium IV 2.8 GHz, 512 MB RAM, dysk 120 GB,
- komputery ivp127, ivp128: Intel Pentium IV 1.8 GHz, 256 MB RAM, dysk 80 GB,
- komputery zti119mp, zti119user: Intel Pentium IV 1.7 GHz, 256 MB RAM, dysk 80 GB.

Na każdym z komputerów zainstalowane były Oracle 9i oraz Java Sun 1.4.2. Wszystkie komputery pracowały pod kontrolą systemu operacyjnego Windows XP za wyjątkiem zti119mp oraz zti119user, na których był zainstalowany Windows 2000.

W skład systemu B-STDW wchodziły wszystkie komputery wyszczególnione na rys. 4, z tym że komputer zti119ewa pracował jako klient/Zarządca B-STDW, pozostałe komputery jako serwery. Zbiór danych przechowywanych w bazach hurtowni obejmował pomiary dla 863 liczników.

Jako warunki końca każdej operacji balansowania przyjęto: obniżenie niezbalansowania poniżej 10% lub osiągnięcie 15 iteracji.

6.1. Test wpływu rozmiaru obszaru podlegającego agregacji na czas balansowania

Celem testu było zbadanie wpływu różnego rozmiaru zbioru testowego na czas balansowania. Do zbalansowania systemu B-STDW użyto zbiorów testowych zawierających pomiary 40, 84, 133, 175 liczników (odpowiednio 1/20, 1/10, 3/20 oraz 1/5 część zbioru roboczego). Obszar podlegający agregacji był tak dobrany, aby podczas agregacji pobrać wszystkie pomiary zbioru testowego. Rozmiar fragmentu był stały i wynosił 5000 krotek, współczynniki corrP i corrN wynosiły odpowiednio 1 i 1 (zwiększanie i zmniejszanie p_i proporcjonalnie do niezbalansowania).

Wykresy niezbalansowania w kolejnych iteracjach przedstawiono na rys. 5.

Jak pokazują wyniki testu, stosując zbiór testowy o rozmiarze 1/20 zbioru roboczego nie udało się zbalansować systemu B-STDW w ciągu 15 iteracji. Dla pozostałych rozmiarów zbiorów testowych osiągnięto zbalansowanie w zbliżonej liczbie iteracji. Jakość balansowania jest podobna, co widać po pokrywających się początkach obu wykresów.

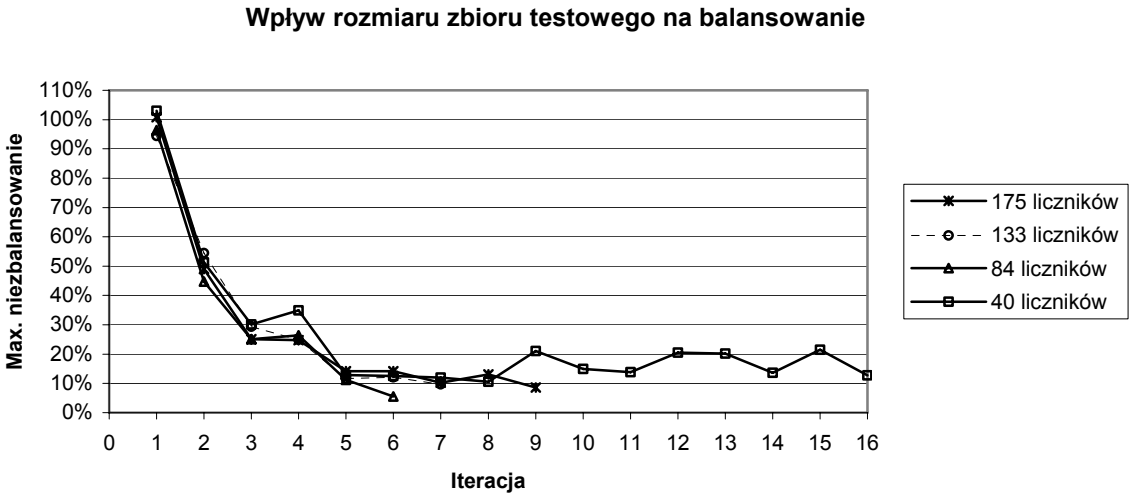
Płynie stąd wniosek, że dla testowanej konfiguracji rozmiar zbioru testowego powinien wynosić przynajmniej 10% rozmiaru zbioru roboczego. Dla zbiorów testowych o mniejszym rozmiarze (a tym samym zapytań o mniejszy obszar) może być wymagana większa liczba iteracji lub nie zapewnią one osiągnięcia odpowiednio małego niezbalansowania maksymalnego. Jak widać na wykresie 5, osiągnięcie niezbalansowania 15% dla obszaru agregacji obejmującego 40 liczników byłoby możliwe po czterech iteracjach.

6.2. Test wpływu rozmiaru fragmentu pomiarów na czas balansowania

Celem tego testu było zbadanie wpływu zmian rozmiaru fragmentu pomiarów na czas balansowania. Rozmiar fragmentu pomiarów określa maksymalną liczbę krotek pomiarów licznika przydzielaną do węzła w jednym kroku planowania. Sprawdzane wartości to: 30000 (brak fragmentacji), 5000, 2500, 500 krotek.

Jako zbiór testowy wybrano zbiór zawierający 133 liczniki (3/20 zbioru roboczego), a współczynniki corrP i corrN wynosiły odpowiednio 1 i 1.

Na rysunku 6 przedstawiono wykresy maksymalnego niezbalansowania dla różnych rozmiarów fragmentu. Pokazują one, że większy rozmiar fragmentu daje gładszy wykres niezbalansowania – brak lub mała liczba wzrostów niezbalansowania. Widać również, że zmniejszanie rozmiaru fragmentu wydłuża proces balansowania. W przypadku fragmentu o rozmiarze 500 krotek zakończenie balansowania było niemożliwe z powodu zmniejszenia udziału najsłabszych węzłów (zti119mp oraz zti119user) poniżej 1% rozmiaru całego zbioru przy dalszym wzroście niezbalansowania.



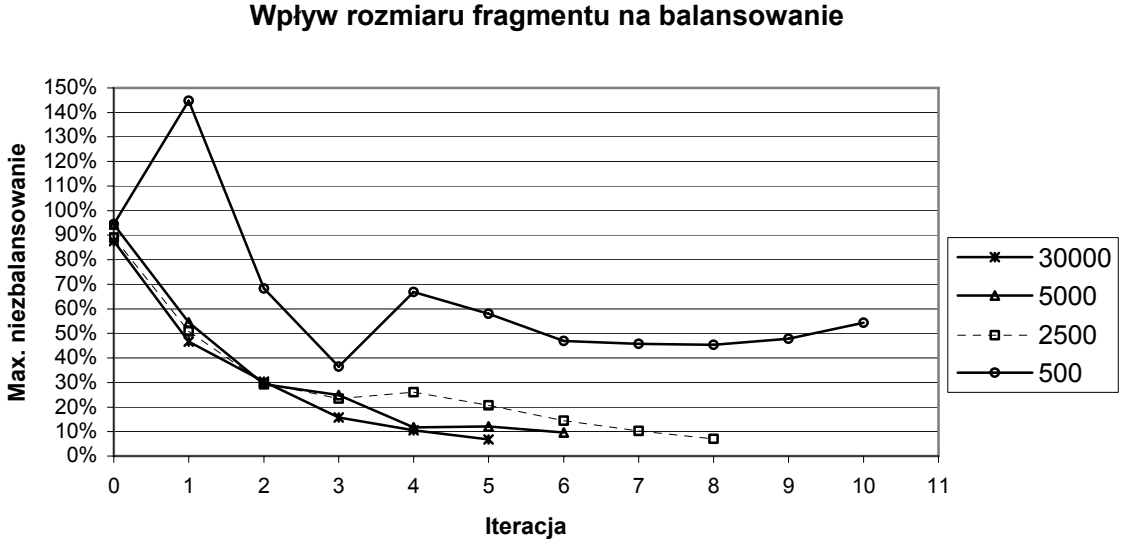
Rys. 5. Maksymalne niezbalansowanie węzłów B-STDW przy zmianach rozmiaru zbioru testowego
Fig. 5. Maximum imbalance of B-STDW nodes under changes of test set size

Wyniki takie są uzależnione od konstrukcji zapytania agregującego, które dla każdego licznika pobiera wszystkie pomiary, a następnie wykonuje na nich operację sortowania. Zmniejszenie rozmiaru fragmentu powoduje wzrost liczby fragmentów, a tym samym wzrost liczby operacji sortowania (w przypadku rozmiaru 500 krotek, przy odpowiednim rozmieszczeniu mogła zajść sytuacja, w której każdy komputer wykonywał sortowanie każdego licznika (fragmentu jego pomiarów), gdyż pomiary były dzielone, na co najmniej 10 fragmentów). To spowodowało nieustanne zmniejszanie rozmiarów przydzielanych zbiorów komputerom słabszym.

Istotne są wnioski wynikające z testów, a mianowicie:

1. Osiągnięcie małego niezbalansowania B-STDW stosując zbiór testowy o niewielkim rozmiarze może wymagać dużej liczby iteracji.

2. Stosowanie fragmentów o małym rozmiarze dla aktualnie wykonywanego przez STDW zapytania agregującego może skutkować trudnościami w zbalansowaniu oraz wydłużyć czas odpowiedzi na zapytania użytkownika.
3. Większe wartości współczynników corrP oraz corrN powodują szybsze zmniejszanie niezbalansowania, ale ostatecznie czas balansowania (liczba iteracji) jest dłuższy.



Rys. 6. Maksymalne niezbalansowanie B-STDW przy zmianach rozmiaru fragmentu
Fig. 6. Maksimum imbalance of B-STDW under changes of fragment size

7. Podsumowanie

Celem pracy był projekt oraz implementacja algorytmu balansowania systemem przestrzennej telemetrycznej hurtowni danych (STDW). Uzyskanym efektem jest wstępne stadium algorytmu balansowania, dające obiecujące wyniki oraz środowisko umożliwiające testy samego algorytmu jak i poszczególnych elementów systemu, co w przyszłości przełoży się na automatyzację testów nowych komponentów systemów STDW oraz ETL.

Do implementacji projektu użyto języka Java wraz z dużym zbiorem dostępnej z nim technologii użytkowej. Testowanie projektu zostało wykonane metodą mieszaną, która jest odpowiednia dla integracji złożonych systemów.

Wyniki testów wydajnościowych potwierdziły możliwość wykonania algorytmu balansującego rozproszoną przestrzenną hurtownię danych z wykorzystaniem metody dobierania rozmiaru zbioru roboczego dla każdego węzła. Wadą rozwiązania jest to, że w obecnym stadium zaprojektowany algorytm jest ściśle związany ze schematem danych telemetrycznych przechowywanych w STDW.

Projektowany system może być rozwijany w następujących kierunkach:

- uogólnienie algorytmu tak, aby pracował z dowolnym schematem danych przestrzennych,
- poprawa procesu balansowania w kierunku osiągania mniejszych niezbalansowań oraz skrócenia czasu trwania procesu poprzez dynamiczny dobór wartości współczynników corrN oraz corrP,
- obsługa balansowania hurtowni podczas kolejnych uaktualnień danych w niej zawartych,
- sprawdzenie innych schematów alokacji danych z pełnym wykorzystaniem techniki plików gridowych,
- zastosowanie komputerów wieloprocesorowych i wielodyskowych w B-STDW.

LITERATURA

1. Dehne F., Eavis T., Rau-Chaplin A.: Parallel Multi-Dimensional ROLAP Indexing. 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan 2003.
2. Bernardino J., Madeira H.: Data Warehousing and OLAP: Improving Query Performance Using Distributed Computing. CAiSE*00 Conference on Advanced Information Systems Engineering, Stockholm 2000.
3. Han, J., Stefanovic, N., Koperski, K. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. In Research and Development in Knowledge Discovery and Data Mining, Second Pacific-Asia Conference, PAKDD'98, Melbourne, Australia, 1998.
4. Gorawski M., Malczok, R.: Distributed Spatial Data Warehouse Indexed with Virtual Memory Aggregation Tree, 5th Workshop on Spatial-Temporal DataBase Management (STDBM_VLDB'04), Toronto, Canada 2004.
5. Gorawski M., Malczok, R.: Projekt i implementacja przestrzennej rozproszonej hurtowni danych klasy ROLAP, Rap. tech. RT/PHD&SI/1/2003, Gliwice, Polska.
6. Gorawski M., Wocław A.: Implementacja algorytmu odtwarzania Design-Resume w technologii JavaBeans. Studia Informatica, 1(52), 2003, Gliwice, Polska.
7. Papadias D., Kalnis P., Zhang J., Tao Y.: Efficient OLAP Operations in Spatial Data Warehouses. Spinger Verlag, LNCS 2001.
8. C. Faloutsos and P. Bhagwat: Declustering using fractals in Proc. of the Int'l Conf. on Parallel and Distributed Information Systems, San Diego, California, January 1993, pp. 18–25.

9. Hua K., Lo Y., Young H.: GeMDA: A Multidimensional Data Partitioning Technique for Multiprocessor Database Systems. Distributed and Parallel Databases, 9, 211-236, University of Florida, 2001.
10. D. Moore. Fast hilbert curve generation, sorting, and range queries http://www.caam.rice.edu/~dougm/twiddle/Hilbert.
11. Faloutsos C., Roseman S.: Fractals for Secondary Key Retrieval. Philadelphia, USA, 1989
12. Zeng Z., Bharadwaj V.: Design and analysis of a non-preemptive decentralized load balancing algorithm for multi-class jobs in distributed networks.

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

Wpłynęło do Redakcji 19 września 2004 r.

Abstract

Balancing of parallel systems workload is very essential to ensure minimal response time of jobs submitted to process. The complexity of data warehouse systems is very high with respect to system structure, data model and many mechanisms used in them, which have an strong influence on final performance. In this paper we present dynamic workload balance algorithm of the spatial telemetric data warehouse. We implement HCAM data partitioning scheme (fig. 2) which use Hilbert curves to order the space. The scheme was modified in way that makes possible setting of dataset size stored in each node. Presented algorithm iteratively calculates optimal size of partitions, which are loaded into each system node (fig. 3), by executing an aggregation series of a test data set. We investigate both situation in which data are and are not fragmented. Moreover we test a set of fragment sizes. Performed system tests (fig. 4-6) confirm possibility of realization of spatial telemetric data warehouse balancing algorithm by selection of dataset size stored in each node. Project was implemented in Java programming language with using of a set of available technology.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, M.Gorawski@zti.iinf.polsl.gliwice.pl.
Robert Chechelski: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, rochel@polsl.gliwice.pl.