Dominik Radziszowski

Skalowalny, komponentowy system zbierania i przechowywania danych pochodzacych z monitorowania systemów rozproszonych

Autoreferat rozprawy doktorskiej

Promotor
Prof. dr hab. inz. Krzysztof Zielinski AGH

Recenzenci
prof. dr hab. inz. Stanislaw Kozielski Politechnika Slaska
dr hab. inz. Grzegorz Dobrowolski AGH

AKADEMIA GÓRNICZO-HUTNICZA
im. Stanislawa Staszica w Krakowie
Wydzial Elektrotechniki, Automatyki, Informatyki i Elektroniki
Katedra Informatyki
Kraków 2007

Wstep

Wspólczesne systemy informatyczne generuja coraz wieksze ilosci informacji dotyczacych swojego dzialania, przebiegu kontrolowanych przez siebie procesów oraz zdarzen zachodzacych w otaczajacej je rzeczywistosci. Obszar dzialania systemów informatycznych powieksza sie nieustannie, obejmujac zastosowania, które jeszcze do niedawna zdawaly sie byc pod wylaczna kontrola czlowieka. Sytuacja ta stanowi wyzwanie dla wspólczesnych systemów zbierajacych i przechowujacych w bazie danych, dane pochodzace z monitorowania, glównie w dwóch plaszczyznach: uniwersalnosci oraz skalowalnosci. Uniwersalnosc systemów polega, w tym kontekscie, na zdolnosci do przechowywania dowolnych, podlegajacych monitorowaniu danych, adaptowalnosci do ich zmiennosci oraz wsparcia dla róznych trybów monitorowania. Skalowalnosc ma sluzyc zapewnieniu odpowiedniej wydajnosci, pomimo wzrostu wielkosci strumienia danych przyjmowanych przez system oraz objetosci danych juz zapisanych.

Fakt dynamicznego rozwoju technologii komponentowych oraz trend w zakresie ich powszechnego wykorzystywania, stawia systemy komponentowe w centrum zainteresowania oraz czyni badania nad ich skalowalnoscia i wydajnoscia szczególnie aktualnymi. Prowadzone w tym obszarze prace skupiaja sie glównie na tworzeniu rozwiazan ogólnych, wlasciwych dla dowolnych obiektowych struktur danych, optymalizowanych pod katem operacji odczytu poprzez wprowadzanie wielopoziomowych mechanizmów pamieci podrecznej. Stosunkowo malo jest prac dotyczacych porównania i optymalizacji architektur aplikacji komponentowych pod katem wydajnosci uzyskiwanej w procesie zapisu duzej ilosci danych. Powyzsze uwarunkowania spowodowaly wybranie technologii komponentowej jako bardzo atrakcyjnego obszaru dla realizacji systemu zbierania i przechowywania danych, który moze sprostac wymaganiom, zarówno w zakresie ogólnosci rozwiazania jak i jego skalowalnosci.

Cel i teza pracy

Autor stawia sobie za cel zbadanie, w jaki sposób, przy zachowaniu uniwersalnosci systemu zbierania i przechowywania danych pochodzacych z monitorowania, mozna zapewnic jego wydajnosc i skalowalnosc. W tym kontekscie zaproponowanych i zaimplementowanych zostalo kilka aplikacji komponentowych, opartych na wielu instancjach baz danych, mechanizmach buforowania i partycjonowania danych oraz komunikacji asynchronicznej; ich wplyw na wydajnosc i skalowalnosc zostal zweryfikowany doswiadczalnie. Celem pracy jest wykazanie, ze w technologiach komponentowych mozliwe jest stworzenie wydajnego i skalowalnego systemu zbierania i przechowywania danych pochodzacych z monitorowania reprezentowanych obiektowo zasobów oraz takiej reprezentacji tych danych, która zapewni elastycznosc ich przechowywania i wyszukiwania. Powyzsze rozwazania doprowadzily do sformulowania nastepujacej tezy:

> ***Mozliwa jest konstrukcja komponentowego systemu zbierania i przechowywania danych pochodzacych z monitorowania systemów rozproszonych, którego dzialanie jest adaptowalne do zmian zarówno rodzaju danych pochodzacych z konkretnego zasobu jak i trybu ich zbierania, zapewniajacego odpowiednia wydajnosc i skalowalnosc.***

Zasób jest rozumiany jako dowolne urzadzenie badz aplikacja udostepniajaca interfejs programistyczny, umozliwiajacy odczyt parametrów okreslajacych jego stan. Reprezentacja obiektowa jest wymaganiem umozliwiajacym ograniczenie badan do modelu obiektowego, w którym dziala wiekszosc wspólczesnych systemów monitorujacych. Nie wplywa ono na ogólnosc rozwiazania, poniewaz inne modele danych mozna zazwyczaj opisac modelem obiektowym. **Adaptowalnosc do zmian** oznacza dzialanie systemu w warunkach duzej zmiennosci monitorowanych zasobów. Monitorowany zasób moze bowiem w dowolnym momencie zmieniac (dodawac, usuwac) parametry, jakie udostepnia do monitorowania. Wartosci róznych parametrów moga byc udostepniane przez zasób w jednym z **trybów**:

raportowanie (ang. push), odpytywanie (ang. pull) oraz sledzenie zmian wartosci (ang. tracing); moga byc one zapisywane z rózna czestotliwoscia. Zaklada sie opracowanie uniwersalnych interfejsów programistycznych, pozwalajacych na jednolity zapis oraz odczyt danych, pochodzacych z dowolnego z monitorowanych zasobów. **Komponentowa** budowa ma na celu wykazanie, ze srodowiska komponentowe nadaja sie do implementacji tego typu systemów oraz zapewniaja odpowiednia **wydajnosc i skalowalnosc**. Konstrukcja systemu w technologii komponentowej jest równiez okazja do pokazania zalet oraz ograniczen tego podejscia.

Oryginalne osiagniecia rozprawy

Optymalizacja systemów komponentowych w zakresie wydajnego zbierania i przechowywania danych pochodzacych z monitorowania nie zostala dotychczas szerzej przedstawiona w literaturze przedmiotu. W celu zaprezentowania problematyki rozprawy przedstawione zostaly wybrane zagadnienia z zakresu monitorowania systemów rozproszonych, technologii komponentowych oraz mechanizmów zwiekszania wydajnosci systemów komponentowych. Na ich podstawie autor okreslil szczegółowa liste wymagan dla SZiPD (Systemu Zbierania i Przechowywania Danych). Kolejno dyskutowane sa mozliwe rozwiazania ogólnej architektury systemu, sposoby wspólpracy z agentami istniejacych systemów monitorujacych oraz mechanizmy zbierania i udostepniania danych. Zaproponowany i szczegółowo opisany zostal model informacyjny dla systemów monitorujacych oraz stworzony na jego podstawie ogólny obiektowy model danych oparty na koncepcji meta-danych; zdefiniowane uniwersalne interfejsy dostepu do danych. W celu wykazania realizowalnosci przyjetych wymagan, autor zbudowal w technologii komponentowej bazowy model systemu oraz pozytywnie zweryfikowal uzyskana przez model funkcjonalnosc.

Dalsze prace mialy na celu zapewnienie odpowiedniej wydajnosci i skalowalnosci systemu. Autor badal wplyw, jaki na te parametry maja zmiany wewnetrznej architektury systemu. Przedstawione i zaimplementowane zostalo piec róznych modeli SZiPD wykorzystujacych rózne mechanizmy optymalizacji systemów komponentowych: klasteryzacje serwerów aplikacji, zwielokrotnienie instancji baz danych, partycjonowanie danych oraz komunikacje asynchroniczna i kolejkowanie z wykorzystaniem brokera komunikatów. Zaprezentowana zostala równiez autorska koncepcja modelu hybrydowego, który posiada zdolnosc adaptacji mechanizmów zapisu danych do wielkosci strumienia danych. Przedstawione modele zostaly uruchomione na dedykowanej infrastrukturze sprzetowej oraz dokonano porównania ich wlasnosci. Zaproponowano metodologie testów, kryteria ewaluacji oraz srodowisko uruchomieniowe, w ramach którego przeprowadzono eksperymenty umozliwiajace praktyczna weryfikacje przedstawionych modeli w kontekscie ich wydajnosci oraz skalowalnosci. Autor wykazal przydatnosc architektury komponentowej do tworzenia skalowalnych systemów zbierania i przechowywania duzej ilosc danych pochodzacych z monitorowania systemów rozproszonych, a tym samym wykazal prawdziwosc postawionej tezy.

Model systemu

Z przeprowadzonej i zawartej w pracy analizy systemów monitorujacych wynika szczegółowa lista wlasnosci, jakie powinny charakteryzowac SZiPD (System Zbierajacy i Przechowujacy Dane pochodzace z monitorowania). Obejmuje ona nastepujace wymagania funkcjonalne:

- Róznorodnosc monitorowanych zasobów. System ma gromadzic dane pochodzace z monitorowania róznych zasobów. Powinien umozliwiac zapis i przechowywanie danych z róznych domen podlegajacych monitorowaniu, np. sieci komputerowych, jednostek obliczeniowych, pamieci masowych itp.
- Dynamiczne definiowanie zasobów. System musi umozliwiac rejestrowanie i usuwanie zasobów podlegajacych monitorowaniu w czasie swojego dzialania.

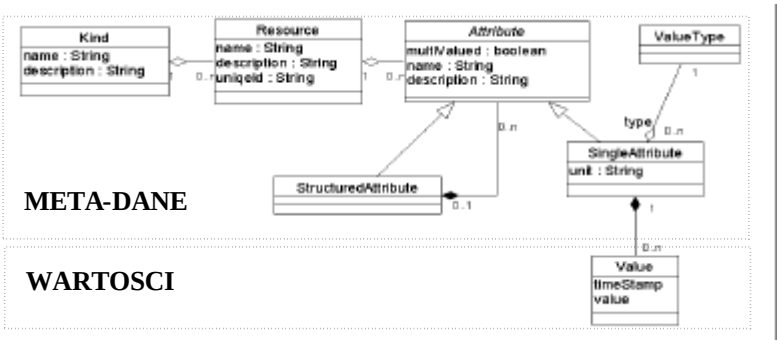
- Dynamiczne definiowanie atrybutów. System musi umozliwiac rejestrowanie atrybutów podlegajacych monitorowaniu na danym zasobie oraz latwe rozszerzanie tej listy o nowe atrybuty, które zasób udostepnil juz w czasie dzialania systemu.
- Obsluga atrybutów zlozonych. Analizowane systemy monitorujace wykorzystywaly logiczne grupowanie atrybutów w wieksze struktury (np. atrybut 'statystyka procesora' moze skladac sie z kilku atrybutów: czas bezczynnosci, srednie obciazenie itp.).
- Obsluga atrybutów wielowartosciowych. System musi obslugiwac sytuacje, w których wartosc atrybutu wyrazona jest kilkoma wartosciami (ilosc ta moze byc rózna w czasie). Przykladem takiego wielowartosciowego atrybutu moze byc atrybut „zalogowani uzytkownicy systemu operacyjnego", jego wartosc sklada sie z kilku wartosci bedacych nazwami aktualnie zalogowanych uzytkowników.
- Obsluga róznych typów danych. System powinien uwzgledniac mozliwie szerokie spektrum prostych typów danych wystepujacych w systemach monitorujacych.
- Rózna czestotliwosc zapisu. Konstrukcja systemu musi uwzgledniac mozliwa, rózna czestotliwosc zapisu danych pochodzacych z jednego zasobu. System powinien umozliwiac niezalezny zapis wartosci poszczególnych atrybutów oraz grupowanie danych.
- Jednolity interfejs zapisu danych. Pomimo mozliwej duzej zmiennosci atrybutów, udostepnianych przez zasób do monitorowania, system powinien udostepniac jednolity interfejs dla zapisu danych o zasobach, atrybutach i ich strukturze oraz samych wartosci.
- Jednolity interfejs dostepu do danych. Pomimo róznorodnosci zasobów podlegajacych monitorowaniu, system powinien udostepniac jednolity interfejs dostepu zarówno do informacji o zasobach, atrybutach i ich strukturze jak i do zgromadzonych wartosci. Interfejs powinien umozliwiac selekcje i filtrowanie zwracanych wartosci.
- Rózne tryby monitorowania. System powinien wspierac trzy podstawowe tryby monitorowania atrybutów: raportowanie (ang. push), odpytywanie (ang. pull) oraz sledzenie zmian wartosci (ang. tracing).

Najwiekszym wyzwaniem dla konstrukcji systemu opartego na powyzszych wymaganiach jest konstrukcja uniwersalnego modelu danych, uniwersalnych interfejsów zapisu i dostepu do danych oraz zapewnienie odpowiedniej wydajnosci i skalowalnosci.

Ogólny model danych

Ogólny obiektowy model danych sklada sie z dwóch grup danych: meta-danych, czyli danych opisujace dane oraz wartosci atrybutów, czyli konkretnych, zmiennych w czasie, wartosc monitorowanych atrybutów – Rys. 1. Meta-dane obejmuja:

- Przestrzen nazw (ang. kind) — grupuje podobne zasoby, czyli te opisywane przez podobne atrybuty (np. komputery, urzadzenia sieciowe). Jest opisywana przez nazwe i opis.
- Zasób (ang. resource) — reprezentuje monitorowany obiekt, jest opisywany unikalnym identyfikatorem zasobu, nazwa zasobu, opisem, posiada tez referencje do przestrzeni nazw, do której nalezy,

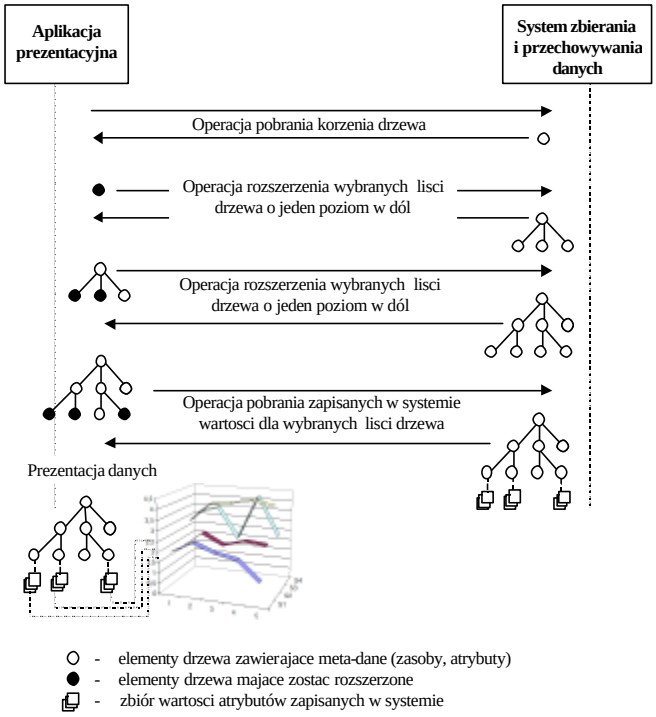


Rys. 1 Ogólny model danych pochodzacych z monitorowania.

- Atrybut (ang. attribute) — reprezentuje monitorowany parametr. Atrybuty moga sie zagniezdzac tworzac drzewiaste struktury. Atrybut jest, zatem klasa abstrakcyjna posiadajaca dwie podklasy: *StructuredAttribute* sluzaca do budowy tych struktur oraz *SimpleAttribute* reprezentujaca atrybut prosty. Atrybut moze opisywac zarówno dane skalarne jak i wielowartosciowe, wlasnosc ta moze byc regulowana przez odpowiednie ustawienie pola *multiValued*. Atrybut jest opisywany przez nazwe, opis oraz referencje zasobu, którego jest elementem. Atrybut prosty posiada dodatkowo jednostke oraz typ danych (klasa *ValueType*).

Uniwersalne interfejsy dostepu

Proces przeslania danych moze byc realizowany na dwa sposoby. Pierwszy sposób polega na dystrybucji konkretnej zmierzonej wartosci wraz z cala informacja potrzebna do jej interpretacji (meta-dane), a wiec cala struktura atrybutów. Drugie podejscie zaklada przeslanie w pierwszym kroku informacji opisujacej meta-dane, a nastepnie wykorzystanie identyfikatorów, nadanych przez system atrybutom prostym. Zmierzone wartosci sa dodatkowo wyposazane we wlasciwy identyfikator atrybutu, dzieki któremu po przeslaniu do systemu moga byc skojarzone i poprawnie przyporzadkowane odpowiednim atrybutom. SZiPD zostal oparty na koncepcji rozdzielnego przekazywania danych i meta-danych, procesu zbierania danych podzielony zostal na dwie skladowe:



Rys. 3 Dostep do danych – koncepcja uszczególawiania drzewa danych.

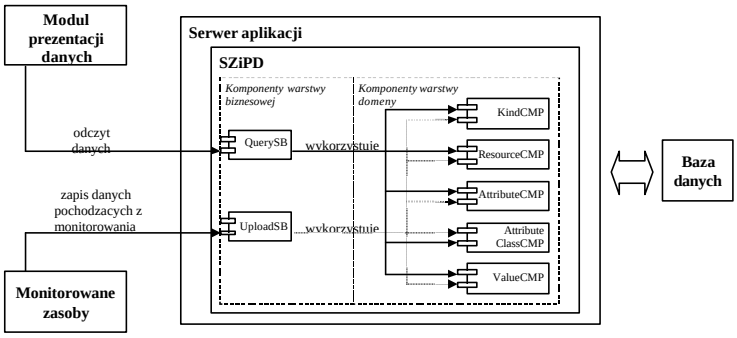
- konfigurowanie meta-danych – operacje zwiazane z rejestracja nowego zasobu w systemie, uzgodnieniem (stworzeniem lub rozszerzeniem) struktury atrybutów udostepnionych przez zasób do monitorowania,
- przekazywanie wartosci atrybutów – wprowadzanie wartosci konkretnych monitorowanych atrybutów.

Adaptowalnosc systemu polega na tym, ze moga sie one wzajemnie przeplatac. Proponowany w pracy interfejs zapisu danych posiada postulowane wlasnosci: monitorowany zasób moze w dowolnym momencie zmieniac (dodawac, usuwac) parametry, jakie udostepnia do monitorowania, a pochodzace z zasobów dane moga byc zapisywane z rózna czestotliwoscia.

Pobieranie danych z systemu zostalo oparte na ogólnym, obiektowym modelu danych i bazuje na koncepcji przekazywania tworzacej drzewo struktury meta-danych miedzy aplikacja prezentacyjna, a systemem. Drzewo jest w kolejnych krokach uszczególawiane o jeden poziom w dól, rozwijane sa jednak jedynie wybrane liscie drzewa – Rys 3. W ostatnim kroku, kiedy w lisciach drzewa sa juz atrybuty proste, wywolywana jest operacja pobrania danych – zwracane drzewo jest rozbudowywane o wartosci atrybutów Operacja pobierania danych posiada dodatkowo parametry zwiazane z filtrowaniem wartosci ze wzgledu na znacznik czasowy oraz ilosc zwracanych wartosci.

Bazowy model SZiPD

W bazowym modelu SZiPD wyróżniono dwa komponenty warstwy biznesowej: UploadSB oraz QuerySB. Komponenty te implementuja uniwersalne interfejsy dostepu do systemu, zostaly zaimplementowane jako sesyjne komponenty bezstanowe. W warstwie domeny bazowego modelu SZiPD wykorzystano piec komponentów – Rys 4. Głównym zadaniem tych komponentów jest zapis i odczyt z bazy danych stanu obiektów zgodnych z ogólnym modelem danych. Sposród dostepnych komponentów warstwy domeny do celów implementacyjnych wybrane zostaly komponenty encyjne zarzadzane przez kontener (CMP). Sa one, pomimo pewnych wad, najbardziej dojrzale wsród rozwiazan aktualnie wykorzystywanych w ramach technologii J2EE. Zdefiniowane w komponentach nazwy atrybutów jak i relacje sa analogiczne do zawartych w ogólnym modelu danych. Model bazowy zostal zaimplementowany oraz pozytywnie przeszedl weryfikacje posiadanych wlasnosci funkcjonalnosci.



Rys. 4 Bazowy model SZiPD.

Rozszerzenia modelu bazowego

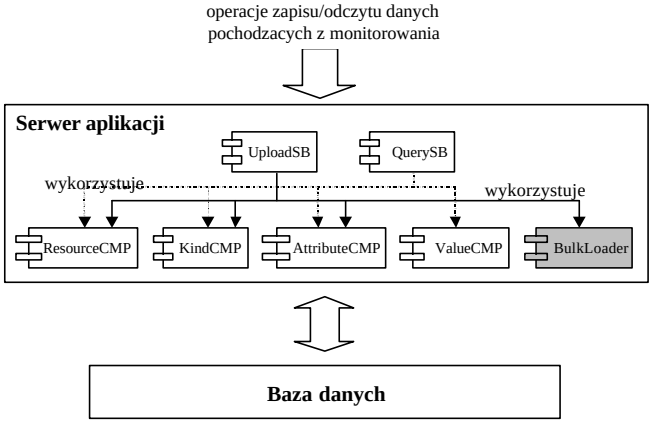
Celem pracy jest wykazanie, ze w technologii komponentowej mozliwe jest stworzenie systemu zbierania i przechowywania danych zapewniajacego odpowiednia wydajnosc i skalowalnosc. Termin wydajnosc zostal dla potrzeb tej pracy zdefiniowany jako maksymalna ilosc przetworzonych w jednostce czasu wywolan, przy których system zachowuje zadane parametry jakosciowe, obejmujace czas odpowiedzi oraz poprawnosc wykonania operacji. Skalowalnosc zostala zdefiniowana jako zdolnosc systemu do utrzymania parametrów jakosciowych pomimo zwiekszania przetwarzanego strumienia danych, uzyskiwana poprzez rozszerzanie zasobów, w oparciu, o które system dziala. W celu uzyskania obu wlasnosci, bazowy model SZiPD zostal poddany optymalizacji, a nastepnie szeregu modyfikacjom majacym na celu zrównoleglenie i uniezaleznienie przetwarzania we wszystkich jego warstwach. Autor zweryfikowal, w jaki sposób mechanizmy zwiekszania wydajnosci systemów komponentowych, poprawiaja parametry wydajnosciowe SZiPD.

Optymalizacja modelu bazowego (M1)

W SZiPD najczesciej wykonywana operacja jest operacja zapisu paczki danych udostepniana przez komponent *UploadSB*, wykonywana wewnetrznie za posrednictwem komponentu *ValueCMP*. Specyfika dzialania komponentów warstwy domeny, do których nalezy komponent *ValueCMP*, polega na indywidualnej obsludze kazdej zapisywanej wartosci, a tym samym kazdego wiersza bazy danych. Prowadzi to, bez wzgledu na przyjety sposób implementacji komponentu (CMP, BMP, OJB, DAO) do tworzenia duzej ilosci niezaleznych zadan zapisu danych do bazy danych, wykonywanych w ramach jednej operacji zapisu wywolanej przez klienta. Proponowana modyfikacja modelu bazowego polega na wprowadzeniu komponentu dla grupowego zapisu danych o nazwie *BulkLoader* – Rys. 5. Umozliwia on zapis w jednym wywolaniu operacji na bazie danych jednej badz wielu wartosci *ValueDTO* równoczesnie. Jest on wykorzystywany przez komponent *UploadSB* do efektywnego zapisu paczki danych otrzymanych od klienta. Sposób obslugi meta-danych oraz odczytu danych nie zmienia sie.

Model ze sklastrowanym serwerem aplikacji (M2)

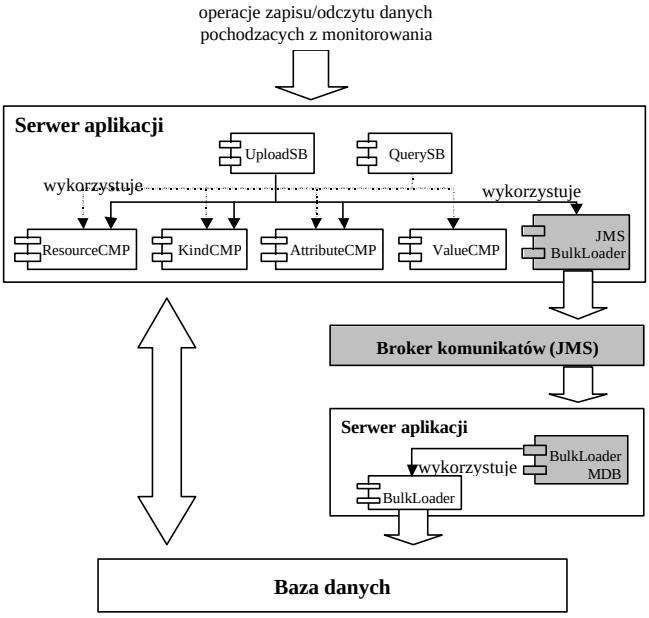
Zwiekszenie wydajnosci w srodowisku klastra serwerów aplikacji jest realizowane poprzez odpowiednia konfiguracje serwerów oraz rozmieszczenie komponentów. Procesowi temu moga podlegac dowolne komponenty EJB, jednak uzyskiwany wzrost wydajnosci jest warunkowany rodzajem komponentu oraz sposobem jego implementacji. Klasteryzacja komponentów SZiPD nie wymagala zmian w ich implementacji, a jedynie drobnych modyfikacji plików deskryptorów, co jest waznym atutem zaproponowanego rozwiazania.



Rys. 5 Zoptymalizowany model SZiPD.

Model oparty na brokerze komunikatów (M3)

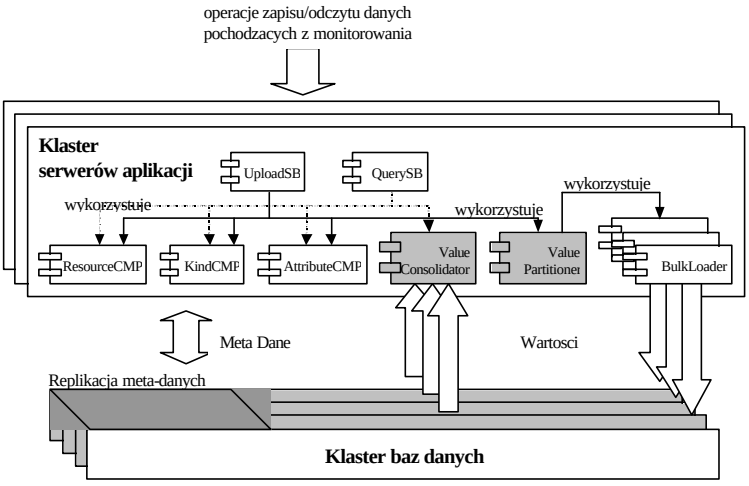
W celu poprawienie wydajnosci w modelu M3 zastosowano broker komunikatów. Podstawowe wlasnosci brokera: asynchronicznosc i kolejkowanie operacji oraz równowazenie obciazenia sa bardzo pozadane dla operacji o dlugim czasie wykonania. Z taka sytuacja mamy do czynienia w SZiPD. Operacja zapisu paczki danych do bazy danych w przypadku duzej ilosci danych jest dlugotrwala; jej wykonanie wymaga ponadto uzyskania dostepu do polaczenia z baza danych, na którego zwolnienie – w przypadku duzego obciazenia systemu – trzeba oczekiwac. Koncepcja zastosowania sytemu kolejkowego w SZiPD rozszerza model systemu o element brokera komunikatów, komponent przekazujacy dane do kolejki brokera oraz komponent sterowany komunikatami przetwarzajacy komunikaty z kolejki – Rys 6. Model SZiPD wykorzystujacy broker komunikatów wymaga implementacji komponentu *JMSBulkLoader*, którego zadaniem jest stworzenie komunikatu na podstawie zbioru obiektów *ValueDTO* i jego umieszczenie w dedykowanej kolejce brokera komunikatów dzialajacej w modelu punkt-punkt. Komunikat jest nastepnie przetwarzany przez komponent *BulkLoaderMDB*; pobierany jest z niego zbiór obiektów *ValueDTO* i zapisywany w bazie danych z wykorzystaniem, komponentu *BulkLoader*. Wprowadzenie brokera komunikatów umozliwia rozluznienie zaleznosci pomiedzy komponentami przyjmujacymi wywolania od klientów, a komponentami zapisujacymi dane w bazie, powoduje zwiekszenie zrównoleglenia przetwarzania oraz uniezaleznia zakonczenie wykonywania operacji przez klienta od obciazenia bazy danych.



Rys. 6 Ogólny model SZiPD wykorzystujacy broker komunikatów.

Model oparty na partycjonowaniu danych (M4)

Jedynym sposobem pokonania granicy wydajnosci jednej instancji bazy danych jest wykorzystanie klastra baz danych. Sposród dostepnych rozwiazan, najwyzsza skalowalnosc zapewnia architektura bez współdzielen (ang. shared nothing), której wykorzystanie wymaga partycjonowania danych. Proponowane wczesniej modele traktowaly baze danych jako system jednolity (SSI – ang. single system image). Konfiguracja bazy danych mogla, zatem wewnetrznie wykorzystywac mechanizmy klasteryzacji, jednak dla zewnetrznych systemów pozostawala jednolitym, spójnym elementem infrastruktury. Takie „widzenie" bazy danych jest przyjete dla systemów klasy J2EE; umozliwia ono wykorzystanie mechanizmów partycjonowania, jednak ich zastosowanie jest warunkowane ich dostepnoscia w wykorzystywanej bazie danych (wiekszosc popularnych, dostepnych na licencji otwartej baz nie wspiera tego mechanizmu). Aby wykorzystac mechanizmy partycjonowania w sposób niezalezny od bazy danych, zaproponowano model 'swiadomy partycjonowania'. Model SZiPD bazujacy na klasterze baz danych oraz mechanizmach partycjonowania danych przedstawia Rys. 7.

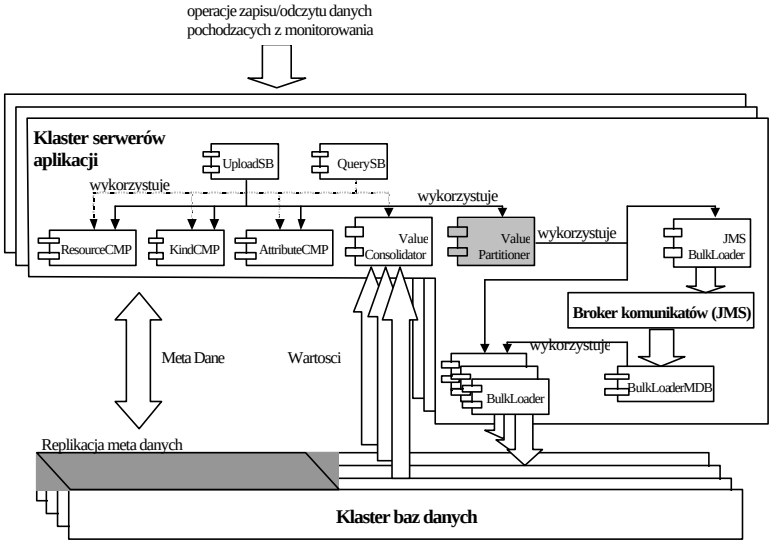


Rys. 7 SZiPD wykorzystujacy partycjonowanie danych

Proponowany model wykorzystuje klaster bazy danych, w którym meta-dane sa replikowane miedzy instancjami, a zawarte dotychczas w jednej tabeli wartosci rozdzielone i zapisywane w róznych instancjach. Podejscie takie wymagalo stworzenia dwóch nowych komponentów *ValuePartitioner* i *ValueConsolidator* odpowiedzialnych odpowiednio za partycjonowanie danych wedlug przyjetego klucza oraz za pobieranie spelniajacych zadania odczytu danych zróznych instancji bazy danych.

Model hybrydowy(MH)

Ostatnim z proponowanych modeli jest model hybrydowy – Rys. 8. Laczy on cechy modelu opartego na brokerze komunikatów oraz modelu opartego na partycjonowaniu danych. Model hybrydowy wykorzystuje partycjonowanie danych oraz dwa tryby zapisu danych. Przy malym obciazeniu wykorzystywane jest partycjonowanie danych oraz synchroniczny tryb zapisu, który gwarantuje male opóznienia w zapisie danych. Sytuacja przeciazenia systemu powoduje aktywowanie brokera komunikatów, do którego przekazywana jest „nadwyzka" danych, która nie moze byc



Rys. 8 Hybrydowy model SZiPD.

przetworzona w trybie synchronicznym. Model wykorzystuje komponenty stworzone na potrzeby wczesniejszych rozwiazan. W trybie synchronicznym, dane zapisywane sa do róznych instancji baz danych przez komponenty *BulkLoader*. W trybie asynchronicznym komponent *JMSBulkLoader* przekazuje dane do brokera komunikatów, sa one nastepnie przetwarzane przez komponent *BulkLoaderMDB* i zapisywane w bazie danych przy wykorzystaniu komponentu *BulkLoader*. Odczyt danych jest realizowany podobnie jak w poprzednim modelu poprzez komponent *ValueConsolidator*.

Prezentowany w poprzednim punkcie model wykorzystujacy klaster bazy danych nie wprowadzal dodatkowych opóznien przy zapisie, nie radzil sobie jednak z chwilowym zwiekszaniem strumienia danych. Proponowany model hybrydowy laczy zalety obu rozwiazan, umozliwia zachowanie parametrów wydajnosciowych w sytuacji chwilowego przeciazenia systemu oraz niski czas zapisu, gdy system jest mniej obciazony. Kluczem do jego poprawnego dzialania jest wlasciwa detekcja momentu, w którym nalezy aktywowac broker komunikatów.

Testy skalowalnosci i wydajnosci

Przeprowadzone badania stanowia praktyczna weryfikacje proponowanych modeli SZiPD pod wzgledem uzyskiwanych parametrów wydajnosciowych oraz skalowalnosci. Autor proponuje metodologie, srodowisko testowe, zestaw testów, metryki oraz kryteria ewaluacji, które sa podstawa dla przeprowadzenia eksperymentów, obejmujacych rózne modele SZiPD w róznych konfiguracjach, poddawane zróznicowanemu obciazeniu.

Infrastruktura sprzetowa

Testy zostaly uruchomione na platformie SUN Fire B1600 Blade; dostepna w Katedrze Informatyki AGH instalacja zawiera 48 komputerów SUN Fire B100s Blade Server. Na potrzeby testów zostalo wydzielone 16 komputerów, tworzacych dedykowany klaster, calkowicie niezalezny od pozostalych komputerów platformy oraz izolowany dzieki zastosowaniu technologii wirtualnych sieci prywatnych (VPN) oraz prywatnej adresacji sieciowej. Sposród dostepnych serwerów aplikacji autor wybral WebLogic Application Server 9.0 jako produkt najbardziej uznany w srodowisku komercyjnym. Bardzo wysoka wydajnosc, mozliwosci klastrowania oraz posiadana licencja sklonily do wyboru bazy danych Oracle 10g jako motoru bazy danych dla przeprowadzanych testów. W testach wykorzystano równiez broker komunikatów dostarczany z serwerem aplikacji WebLogic.

Metodyka testów

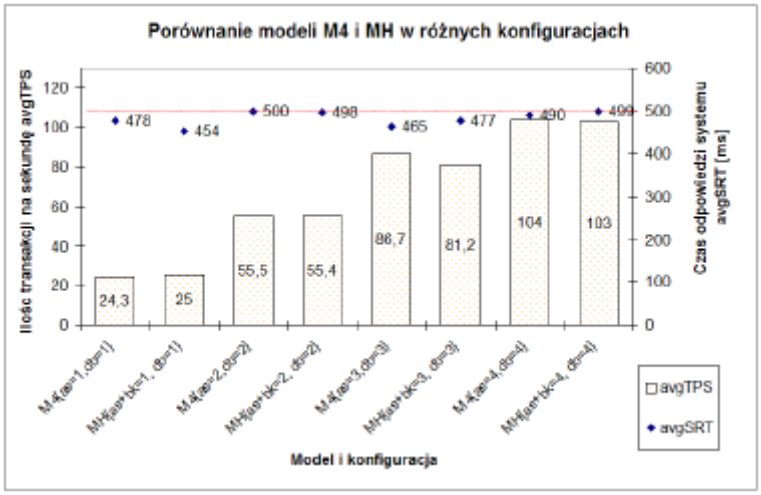
Do badania parametrów wydajnosciowych systemu zaproponowana zostala koncepcja, bazujaca na pojeciu punktu pracy systemu.

> **Punkt pracy systemu** jest taka konfiguracja symulowanego obciazenia konkretnej konfiguracji testowej (a wiec modelu i elementów infrastruktury), przy której osiagane sa maksymalne wartosci parametrów wydajnosciowych oraz zachowane zadane parametry jakosciowe.

Porównanie poszczególnych modeli i konfiguracji polega na porównaniu parametrów wydajnosciowych uzyskanych w punktach pracy, a wiec najlepszych, sposród uzyskanych przez modele w konkretnej konfiguracji wyników, przy których zachowane byly zadane parametry wydajnosciowe. Konieczne bylo wielokrotne wykonywanie testów konkretnego modelu w okreslonej konfiguracji, poddawanemu róznemu obciazeniu. Sposród uzyskanego zbioru wyników, wybierany byl jeden o najlepszych parametrach wydajnosciowych i spelniajacy zadane kryterium jakosciowe.

Uzyskane rezultaty

W pracy przedstawiono szczególowe rezultaty wraz z ich interpretacja. Najwazniejszym jest wykazanie, ze SZiPD jest skalowalny i zapewnia odpowiednia dla tego typu systemów wydajnosc – Rys. 9. Waznym rezultatem jest opracowanie, bazujacej na koncepcji punktu pracy systemu, metodologii testów oraz srodowiska narzedziowego usprawniajacego wykonanie kilkuset testów.



Rys. 9 Porównanie modeli M4 i MH w róznych konfiguracjach

Wnioski

Skonstruowany system zbierania i przechowywania danych pochodzacych z monitorowania systemów rozproszonych posiada wymagane dla tej klasy systemów wlasnosci, co zostalo wykazane doswiadczalnie. System potrafi dzialac w warunkach duzej zmiennosci monitorowanych zasobów, które moga w dowolnym momencie zmieniac (dodawac, usuwac) parametry, jakie udostepniaja do monitorowania. System moze laczyc sie z istniejacymi agentami systemów monitorujacych oraz zbierac dane w dowolnym z trybów: raportowanie, odpytywanie oraz sledzenie zmian wartosci; dane moga byc zapisywane z rózna czestotliwoscia. Posiadane wlasnosci funkcjonalne system zawdziecza, w glównej mierze, ogólnemu obiektowemu modelowi danych oraz uniwersalnym interfejsom dostepu. Spelnieniu wymagan odnosnie wydajnosci systemu poswiecone zostaly badania eksperymentalne. Pokazuja one, ze najbardziej dojrzaly sposród modeli – model hybrydowy, posiada odpowiednia wydajnosc, jest skalowalny oraz odporny na chwilowe zwiekszanie obciazenia.

Sposób okreslania i zaspakajania wymagan aplikacji komponentowych, migracja komponentów, mechanizmy optymalnego rozmieszczania komponentów oraz przydzielania odpowiedniej liczby zasobów, podnoszenie wydajnosci systemu oraz uodparnianie na nierównomiernosc strumienia danych, bez potrzeby zwiekszania zasób sprzetowych, sa dzisiaj aktualnymi problemami stojacymi przed twórcami platform i aplikacji komponentowych. Aktualnie wykorzystywane sa jedynie proste mechanizmy, w sposób automatyczny dostosowujace wartosci wybranych parametrów konfiguracyjnych. Zdaniem autora przyszlosc nalezy do platform, które beda zaspakajac nie tyle zdefiniowane przez aplikacje wymagania odnosnie wykorzystywanych zasobów, ale okreslone przez nia parametry jakosciowe, których zaspokojenie, poprzez wlasciwy przydzial zasobów, bedzie domena platformy. Platform, potrafiacych w sposób dynamiczny przydzielac zasoby na podstawie okreslonych polityk jakosci uslugi oraz mogacych ingerowac w konfiguracje i sposób dzialania poszczególnych komponentów. Mozliwosci adaptowania sposobu dzialania aplikacji komponentowej w celu dotrzymania parametrów jakosciowych, wykorzystane przez autora przy implementacji modelu hybrydowego, wpisuja sie w ten nurt oraz lokuja przeprowadzone prace w bardzo aktualnym kontekscie.