STUDIA INFORMATICA 2009
Volume 30 Number 2B (84)

Arkadiusz GABIGA, Adam PIÓRKOWSKI, Tomasz DANEK
Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej

ANALIZA WYDAJNOŚCI TECHNOLOGII SERWLETOWEJ W WYBRANYCH ZASTOSOWANIACH BAZODANOWYCH

Streszczenie. Magazynowanie i udostępnianie dużej ilości danych binarnych wymaga dobrze zaprojektowanej architektury. Obecnie jedną z wiodących technologii tworzenia aplikacji typu cienki klient jest technologia serwletowa. Zasadne jest dokonanie oszacowania jej wydajności we współpracy z wybranymi systemami zarządzania bazami danych pod kątem tworzonego internetowego serwisu odsłonięć geologicznych.

Słowa kluczowe: serwlet, serwer, cienki klient, PostgreSQL, MySQL

EFFICIENCY ANALYSIS OF SERVLET TECHNOLOGY IN SELECTED DATABASE OPERATIONS

Summary. This article focuses on servlet technology as a main technology for university server which role is to collect and serve geological data acquired by students. The web application was tested for photos 10KB, 100KB, 1000KB and 10000KB on two operating systems (Windows and Linux) using MySQL and PostgreSQL.

Keywords: servlet, server, thin client, PostgreSQL, MySQL

1. Wstęp

Każdego roku studenci Wydziału Geologii, Geofizyki i Ochrony Środowiska odbywają praktyki terenowe. Podczas zajęć dokumentują znalezione okazy skalne, a następnie wszystkie te dane łącznie ze zdjęciami odsłonięć wpisują do bazy danych znajdującej się na uczelnianym serwerze [1, 2, 3]. Na wydziale studiuje około 600 studentów, w związku z tym przed serwerem stoją niemałe wymagania. Każdy z tych studentów w ciągu trwania semestru

(a zwłaszcza w okresie sesyjnym, kiedy oblężenie serwera jest największe) łączy się z bazą przez stronę w celu przeglądania zbioru zdjęć. Zakładając, że każdy student przegląda kilkadziesiąt zdjęć, serwer musi obsłużyć dużą ilość zapytań zwracających różnej wielkości zdjęcia.

Aby sprostać powyższym problemom, użyto technologii serwletowej, która jest popularnym narzędziem używanym przy realizacji projektów biznesowych [4].

W dzisiejszych czasach użytkownicy posiadają coraz szybsze łącza internetowe, co pociąga za sobą większe wymagania stawiane przed serwerami aplikacji WWW. Od dłuższego czasu można zaobserwować, że wraz ze wzrostem ogólnodostępnej prędkości łącza internetowego, jakość stron internetowych wzrasta. Zwłaszcza jakość dostępnych multimediów, w tym zdjęć, jest coraz wyższa. Galerie internetowe, portale społeczne, pobieranie zdjęć oraz wrzucanie dużej ilości zdjęć powoduje bardzo duże obciążenie dla sieci. Dlatego też ważne jest dobre zaplanowanie całej architektury systemu.

2. Architektura aplikacji

Obecnie na rynku istnieje wiele technologii umożliwiających implementację przedsięwzięcia. Z komercyjnych rozwiązań na pewno należy wymienić ASP.NET w połączeniu z MS SQL. Z darmowych technologii należy wymienić środowisko Sun Java oraz PHP w połączeniu z darmowymi systemami zarządzania bazami MySQL [5] i PostgreSQL [6].

Wiadomo, że przy realizacji każdego projektu informatycznego ważne są koszty oraz czas realizacji. Dlatego też przy realizacji tego projektu zostały użyte niekomercyjne rozwiązania.

Warstwa logiki niniejszego projektu została zrealizowana w technologii Java 1.5 – środowisko Sun Microsystems od długiego czasu zdobywa sobie rzesze zwolenników. Jego niezaprzeczalną zaletą jest przenośność kodu pomiędzy systemami. Jest to możliwe dzięki kompilacji do kodu pośredniego, niezależnego od architektury sprzętowej. Taki kod bajtowy (ang. *bytecode*) jest interpretowany przez wirtualną maszynę Java, która tłumaczy go na kod dostosowany do specyfiki danego środowiska uruchomieniowego.

Nieodłączną częścią środowiska Java w zastosowaniach serwisów internetowych jest technologia serwletowa – odpowiada ona za przetwarzanie zapytań http klienta na serwerze. Przetwarza wszystkie dane przesłane przez przeglądarkę stron internetowych do serwera. Jej zaletą jest to, że implementacja obsługi zapytań to klasy Java, a zatem można korzystać z dobrodziejstw całego zestawu Java API. W ramach projektu została użyta wersja 2.5.

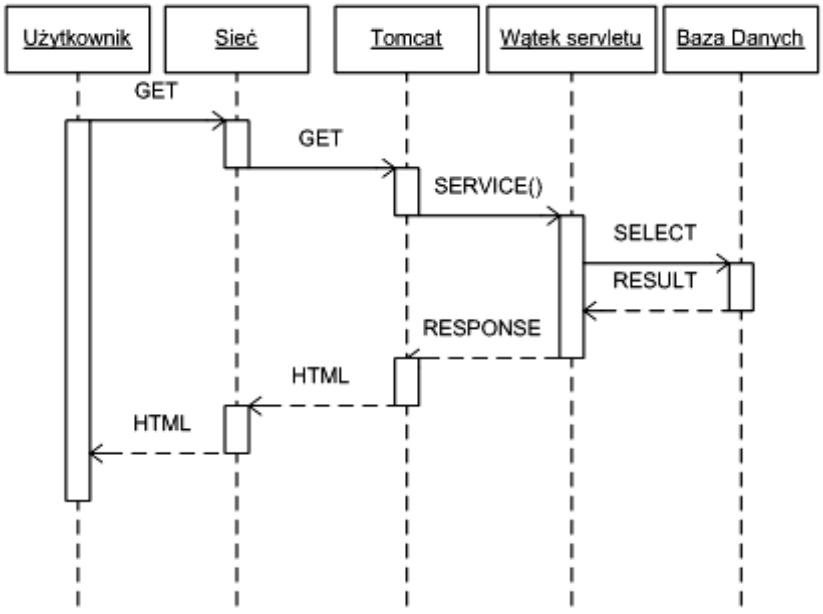
Do uruchomienia serwletów na serwerze konieczny jest tzw. kontener serwletów. Działa on analogicznie do serwera http, zajmuje się obsługą żądań klienta i jednocześnie pozwala na

uruchomienie serwletów, a także zarządza całym cyklem ich życia. Jeden z najpopularniejszych obecnie kontenerów serwletów to Apache Tomcat. Jest w całości napisany w języku Java. Używana w projekcie wersja 6.0 pozwala na obsługę serwletów w wersji 2.5. Tomcat używany jest zwykle w parze z innymi narzędziami jako uzupełnienie ich funkcjonalności, np. przez serwery aplikacyjne obsługujące całą gamę rozwiązań Java Enterprise Edition: JBoss, Apache Geronimo lub serwery http, takie jak Apache (wykorzystując moduł mod_jk).

Bazą używaną do przechowywania danych odsłonięć jest baza GeoKarpaty2 [2], której schemat zaprezentowany jest w pracy [3]. Do celów testowych schemat został zaimplementowany zarówno w MySQL, jak i w PostgreSQL. Wybór ten jest podyktowany jak najniższymi kosztami realizacji serwera uczelnianego.

Aplikacja testowana składa się z dwóch serwletów działających z i bez puli zapytań:

1. ConnPoolTest, który wykorzystuje pulę połączeń zaimplementowaną w Tomcat'cie przez udostępnienie jej jako zasób Java Naming and Directory Interface. Cała konfiguracja zasobu sprowadza się do napisania pliku context.xml, w którym znajduje się nazwa zasobu oraz nazwa klasy sterownika do bazy danych wraz parametrami. Taki plik zostaje umieszczony w katalogu META-INF aplikacji, dzięki temu podczas uruchamiania Tomcata automatycznie zostaje zbudowany wpis w JNDI.
2. NoConnPoolTest, w którym dla każdego wywołania serwletu jest tworzony nowy obiekt połączenia do bazy, co jest bardzo czasochłonne.



Rys. 1. Diagram przebiegu zapytania
Fig. 1. Schema of a request

Proces zapytania składa się z następujących etapów:

- wywołanie żądania sieciowego przez klienta,
- przesłanie żądania do serwera http (Apache),

- przekazanie żądania do kontenera (Tomcat),
- wywołanie odpowiedniego serwletu z puli lub powołanie nowej jego instancji,
- odczyt z bazy danych obrazu graficznego,
- zwrócenie wyników kolejno do serwletu, kontenera, serwera http i klienta.

Całość zobrazowana jest na rys.1.

Testy obciążeniowe całego serwera wykazują, które zestawienie technologii jest najbardziej optymalne do tego zadania.

3. Środowisko testowe

Środowisko testowe składa się z następującego sprzętu komputerowego:

- serwer BLADE – 2,8 GHz x 4 core Intel, 8MB RAM, - sys. op. Windows 2003 server,
- serwer BLADE – 2,8 GHz x 4 core Intel, 8MB RAM, - sys. op. Linux Fedora Core 3,
- przełącznik Gigabit Ethernet,
- komputer testujący (PC Intel 3GHz, 2GB RAM).

Oba serwery posiadają identyczną konfigurację sprzętową i różnią się tylko zainstalowanym systemem. Wybór gigabitowej sieci został podyktowany dużą ilością danych wymienianych pomiędzy klientem (przeglądarką) a serwerem. Duża ilość zdjęć przesyłanych w krótkim czasie powoduje bardzo szybie wyczerpanie zasobów łącza 100 Mb.

Badanie wydajności serwera internetowego jest rzeczą dość trudną, jednakże są dostępne aplikacje wspomagające ten proces. Podążając ścieżką *open source* do przeprowadzania testów, zostały wybrane dwie aplikacje:

Apache JMeter [7] – aplikacja napisana w Javie, o bardzo dużych możliwościach konfiguracyjnych. Trzeba przyznać, że jak na poważne narzędzie do testowania nie jest zbyt skomplikowany i nawet niedoświadczony w tym temacie użytkownik poradzi sobie z konfiguracją prostych testów. Ogólnie i w dużym skrócie mówiąc, cała procedura sprowadza się do ustalenia, ilu użytkowników, w jakim czasie próbuje dostać się na określone strony. Interfejs graficzny jest bardzo przejrzysty i pozwala w bezproblemowy sposób skonfigurować obszerny test obejmujący logowanie, przekierowania, testowanie rozproszone itp. Dodatkową zaletą jest możliwość używania programatora dla testów oraz trybu wsadowego, gdzie podaje się plik ze schematem testu oraz nazwę pliku, do którego zapisać wyniki. Dzięki temu można bez problemu zaplanować testowanie na noc, a następnego dnia analizować wyniki.

WebLOAD firmy RadView [8] – do 2007 roku narzędzie komercyjne, w tej chwili darmowe z bardzo dobrze przygotowaną dokumentacją. Składa się z dwóch głównych części: WebLOAD IDE i WebLOAD Console. WebLOAD IDE służy do tworzenia schematu testu.

Można go napisać używając języka JavaScript bądź włączyć specjalny tryb nagrywania ruchów użytkownika. Polega to na tym, że tuż po wybraniu opcji nagrywania pojawia się okno przeglądarki Internet Explorer i od tego momentu nagrywany jest każdy ruch – wejścia na różne strony WWW, logowanie, wysyłanie formularzy itp. Całe „nagranie" jest widoczne w postaci wygenerowanego kodu w JavaSript. Aplikacja potrafi również symulować czasy, w których użytkownik „zastanawia się" nad „kliknięciem" w kolejne łącze. Wszystko jest zrobione po to, aby test był jak najbardziej zbliżony do rzeczywistości. WebLOAD Console – część odpowiedzialna za uruchomienie testu. Określa się sposób przeprowadzenia testu (może się składać z kilku wcześniej przygotowanych schematów), wybiera się wielkość obciążenia strony, jak jest ono rozłożone w czasie. Do wyboru jest m.in. liniowy wzrost obciążenia, losowe generowanie obciążenia, a także tzw. *warm up*, gdzie można ustalić czas „rozgrzania" serwera. WebLOAD generuje liczne statystyki, które można zapisać jako raporty w formatach html, txt oraz xls.

3.1. Przebieg testów

Aplikacja JMeter, mimo iż jest uznawana za wiodącą i referencyjną w swojej kategorii, okazała się być mało użyteczna – dla pewnych parametrów nie działała poprawnie (odbiór dużych bloków danych jako wyniki zapytania). Dlatego też testy zostały wykonane przy wykorzystaniu aplikacji WebLOAD. W ramach testów zwrócono uwagę na następujące parametry:

- średni czas realizacji żądania,
- średnią liczbę obsłużonych zapytań w ciągu sekundy (przepustowość).

Dodatkowo mierzony był czas pobrania połączenia w serwlecie, a także czas samego wykonania zapytania do bazy. Czasy te pozwalają zobrazować wyniki bez narzutu transmisji sieciowej.

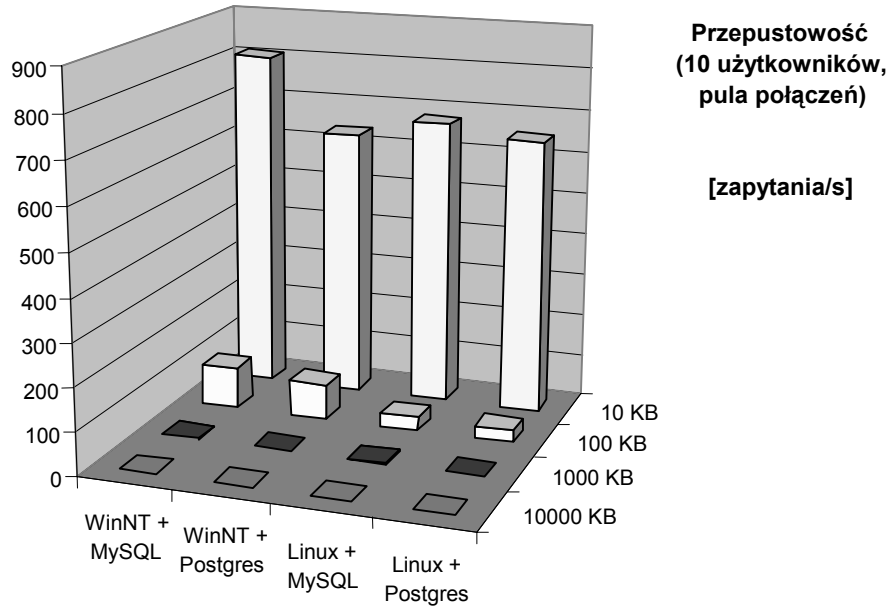
Testy przeprowadzono dla następujących przypadków:

- 10 i 100 jednoczesnych użytkowników,
- wykonanie zapytań z puli połączeń i bez puli,
- odczyt obrazów o rozmiarach 10, 100, 1000, 10000 KB,
- odczyt obrazów z baz danych MySQL i Postgresql.

3.2. Wyniki testów

Otrzymane dane zaprezentowano na wykresach. Ze względu na ograniczony rozmiar pracy przedstawione zostaną wybrane przypadki. Z tego samego względu odstąpiono od za-

mieszczania pełnych tabel danych eksperymentów, a skupiono się na wykresach 3D wraz z odpowiadającymi im danymi w tabelach.

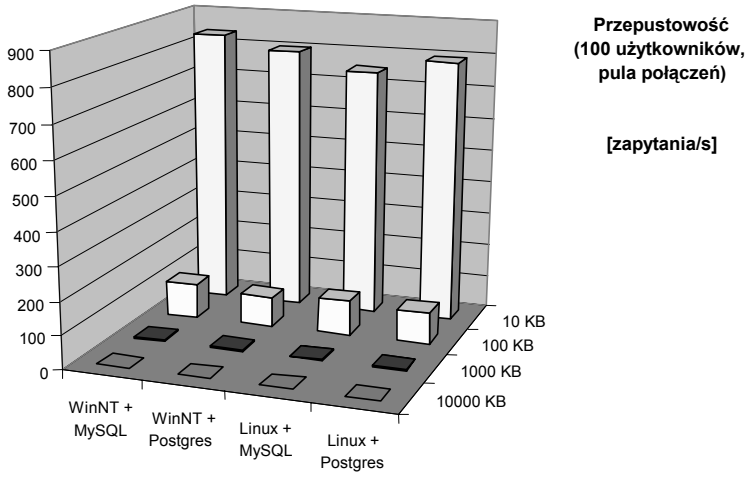


	WinNT +	WinNT +	Linux + MySQL	Linux + Postgres
10000 KB	0,15	0,14	0,13	0,13
1000 KB	2,09	2,06	1,75	1,65
100 KB	100,4	83,8	32,78	28,31
10 KB	807,3	633,2	675,7	649,4

Rys. 2. Wyniki przepustowości dla 10 jednoczesnych użytkowników z włączoną pulą połączeń
Fig. 2. Throughput for 10 simultaneous users with connection pool

Na rys. 2 przedstawiono wyniki przepustowości dla 10 jednoczesnych użytkowników z włączoną pulą połączeń. Najwyższą przepustowość (792 żądań/s) uzyskano dla niewielkich obrazków (rozmiar 10KB) przy użyciu bazy danych MySQL pod kontrolą systemu Windows 2003 Server.

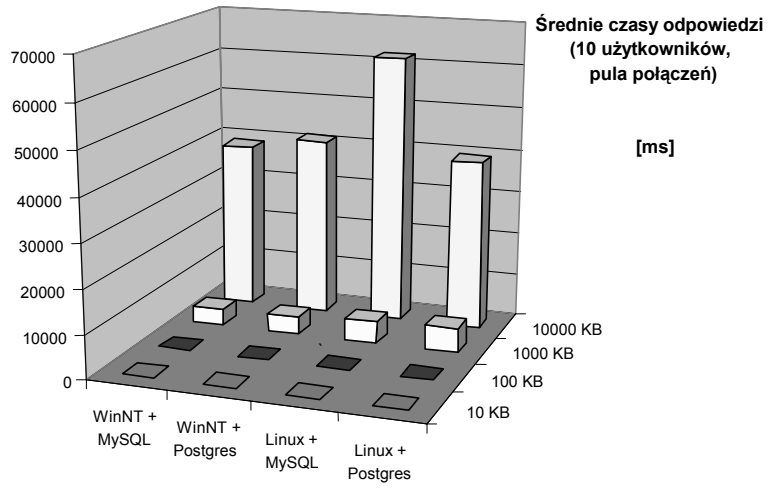
Bardzo podobne wyniki otrzymano dla tych samych warunków, z tą różnicą, że zaprogramowano 100 jednoczesnych użytkowników żądających obsługi. Wyniki zostały przedstawione na rys. 3.



	WinNT + MySQL	WinNT + Postgres	Linux + MySQL	Linux + Postgres
10000 KB	0,27	0,28	0,35	0,22
1000 KB	5,24	4,62	3,89	3,95
100 KB	106,3	89,86	105,6	96,4
10 KB	834,4	797,7	748,3	792,3

Rys. 3. Wyniki przepustowości dla 100 jednoczesnych użytkowników z włączoną pulą połączeń
Fig. 3. Throughput for 100 simultaneous users with connection pool

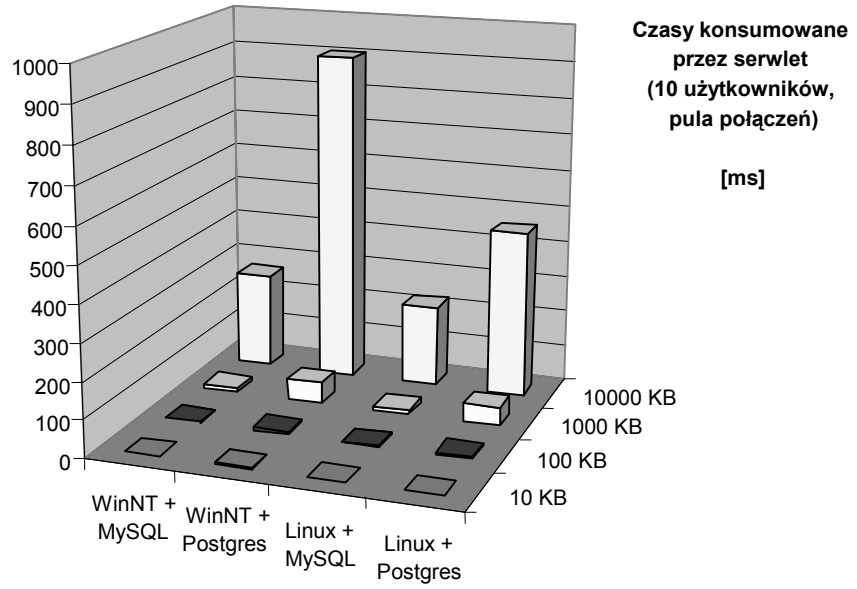
Warto przedstawić wartości średnich czasów zapytania dla pierwszego przypadku (rys. 4). Dla małych obrazków czasy są niewielkie, dla obrazków o rozmiarach przeciętnego zdjęcia odsłonięcia to ok. 4 sekundy dla zdjęć 1 MB i prawie proporcjonalnie 39,8 sekund dla zdjęcia o rozmiarze 10 MB. W przypadku zdjęć 10 MB najszybszą parą technologii okazała się para (Windows, MySQL) – 38,8s i zaraz po niej plasuje się para (Linux, Postgresql) - 39,4s.



	WinNT + MySQL	WinNT + Postgres	Linux + MySQL	Linux + Postgres
10 KB	0	0	0	0
100 KB	0	0	0	0
1000 KB	4000	4043	5130	5695
10000 KB	38812	41352	62157	39421

Rys. 4. Średnie czasy odpowiedzi dla 10 użytkowników z pulą połączeń
Fig. 4. Average response times for 10 users with connection pool

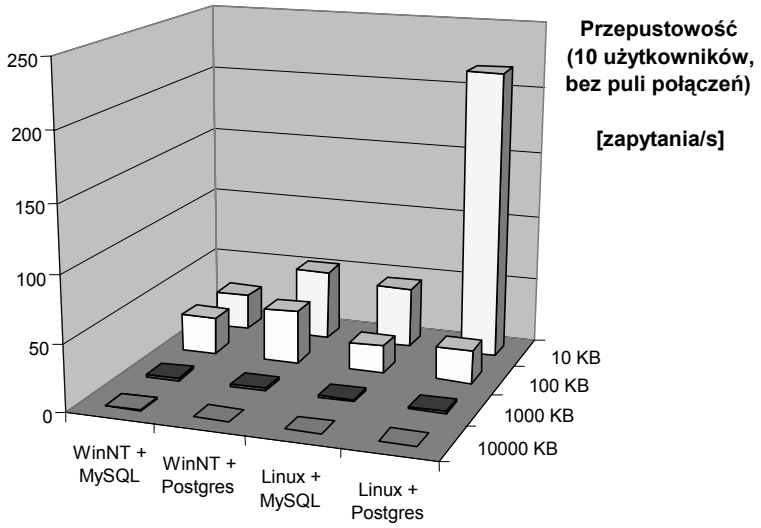
Czasy wykonania operacji przez serwlet dla zapytań przy 10 użytkownikach i użyciu puli połączeń przedstawiono na wykresie (rys. 5). Z analizy danych wynika, że PostgreSQL jest dla środowiska kilka razy wolniejszą bazą, niż MySQL, jednakże czasy te są pomijalnie małe w stosunku do narzutów, jakie są związane z technologią serwletową i transmisją zapytań przez sieć komputerową (ok. kilkadziesiąt razy większe).



	WinNT + MySQL	WinNT + Postgres	Linux + MySQL	Linux + Postgres
10 KB	0,94	2,33	0,55	1,11
100 KB	2,02	8,11	1,43	5,15
1000 KB	14,78	57,28	11,84	50,07
10000 KB	263,2	903,1	223,6	457

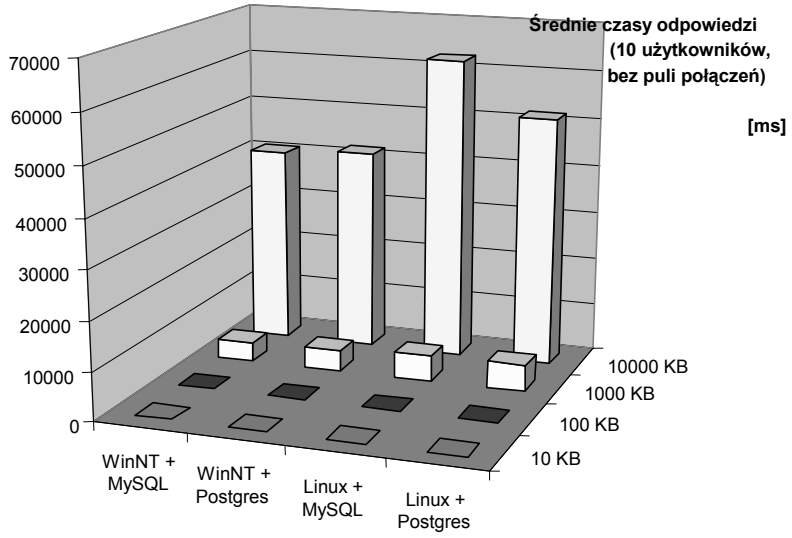
Rys. 5. Czasy konsumowane przez serwlet – 10 użytkowników, pula połączeń
Fig. 5. Servlet times – 10 users, connection pool

Na rysunkach 6 oraz 7 przedstawiono podobne rozważania dla testów bez korzystania z puli połączeń. Średnie czasy odpowiedzi są podobne jak dla przypadku z pulą połączeń. Inaczej ma się sprawa z przepustowością, która w tym przypadku wyraźnie spadła.



	WinNT + MySQL	WinNT + Postgres	Linux + MySQL	Linux + Postgres
10000 KB	0,14	0,14	0,12	0,13
1000 KB	2,07	1,99	1,7	1,66
100 KB	28,4	40,2	20,8	25,1
10 KB	27,3	52,8	45,9	217,5

Rys .6. Wyniki przepustowości dla 10 jednoczesnych użytkowników z wyłączoną pulą połączeń
Fig. 6. Throughput for 10 simultaneous users without a connection pool



	WinNT + MySQL	WinNT + Postgres	Linux + MySQL	Linux + Postgres
10 KB	0	0	0	0
100 KB	0	0	0	0
1000 KB	4050	4434	5391	5565
10000 KB	40764	41823	62550	51666

Rys .7. Średnie czasy odpowiedzi dla 10 użytkowników bez puli połączeń
Fig. 7. Average response times for 10 users without a connection pool

4. Podsumowanie

Technologie serwletowe są obecnie wiodącymi technologiami dynamicznego tworzenia stron WWW. Wyniki przeprowadzanych testów wykazują jednak dość duże narzuty czasowe oraz zjawiska spadku wydajności, jakie mają miejsce przy ich wykorzystaniu. Celowe jest przetestowanie innych technologii, aby wyłonić optymalną. W dziedzinie oprogramowania darmowego ciekawą alternatywą jest środowisko Mono 2.0 + ASP.

Praca finansowana w ramach badań statutowych KGIS nr 11.11.140.561.

BIBLIOGRAFIA

1. Kotlarczyk J., Krawczyk A., Leśniak T., Słomka T.: Geologiczna baza danych GeoKarpaty dla polskich Karpat fliszowych. Wydawnictwo własne WGGiOŚ AGH, Kraków 1997.
2. Malec O., Kyc M., Piórkowski A.: Internetowa baza danych odsłonięć GeoKarpaty II. Materiały kongresowe Pierwszego Polskiego Kongresu Geologicznego, Kraków 26–28 czerwca 2008. Polskie Towarzystwo Geologiczne, 2008.
3. Gajda G., Piórkowski A.: Możliwości konstrukcji hurtowni danych geologicznych. Bazy danych – rozwój metod i technologii – bezpieczeństwo, wybrane technologie i zastosowania. Praca zbiorowa pod red. Stanisława Kozielskiego [et al.]. WKŁ, Warszawa 2008.
4. Java Servlet Technology – @: http://java.sun.com/products/servlet/.
5. MySQL : The world's most popular open source database – @: http://www.mysql.com/.
6. PostgreSQL: The world's most advanced open source database – @: http://www.postgresql.org/.
7. JMeter – Apache JMeter – @: http://jakarta.apache.org/jmeter/.
8. WebLOAD – Open Source Load Testing – @: http://www.webload.org/.

Recenzent: Dr inż. Adam Świtoński

Wpłynęło do Redakcji 30 stycznia 2009 r.

Abstract

Nowadays, in a fast paced world where almost everyone has a broadband Internet connection, the high performance is something greatly desired. Designing and implementing web applications that fulfill those new-world-criteria is not that simple, however there are a lot of frameworks which can be used to achieve this goal. As usual those can be divided into two grups:

- commercial (like ASP.NET, MSSQL),
- free, open source (like PHP, Servlets, MySQL, PostgreSQL).

This article focuses on servlet technology as a main technology for university server which role is to collect and serve geological data acquired by students. This data consists of photos and text descriptions, so the main task for the webapp is to serve big amounts of binary data as fast as possible.

To check if the whole application is ready to be deployed, some load testing is needed. There are a lot of performance and load testing tools along the Internet but definitely open source Radview WebLOAD is worth looking at. The variety of test configuration options allows to imitate user actions as if they where made by a real user (allows even to configure user think-times between clicking on links).

Web application mentioned in this article was tested for photos 10KB, 100KB, 1000KB and 10000KB on two operating systems (Windows and Linux) using MySQL and PostgreSQL. The result of all tests helps to choose an appropriate "technology mix" for geological web application implementation.

Adresy

Tomasz DANEK: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30-059 Kraków, Polska.
Arkadiusz GABIGA: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30-059 Kraków, Polska.
Adam PIÓRKOWSKI: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30-059 Kraków, Polska, pioro@agh.edu.pl.