Testy wydajności

W tym rozdziale zostaną przedstawione wyniki przeprowadzonych testów wydajności nowego klastra. Przy projektowaniu testów były brane pod uwagę dwa zasadnicze aspekty: poprawność oraz szybkość działania. Testy zostały wykonane w laboratorium komputerowym na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Dostępne były dwa typy maszyn o następującej charakterystyce:

Parametr	Maszyna typu A	Maszyna typu B
Procesor	350 MHz	700 MHz
RAM	64 MB	128 MB
Sieć	100 Mb/s	100 Mb/s
SO	Linux	Linux

Do testowania został wykorzystany specjalnie w tym celu napisany program, którego zadaniem było generowanie odpowiedniej liczby równoległych żądań do systemu, uwzględniających przesłany przez system identyfikator sesji. Aby wyniki były bardziej miarodajne, program testujący był uruchamiany równolegle na kilku maszynach (w ten sposób wyniki zostały uniezależnione od wydajności konkretnej maszyny testującej, co miało szczególne znaczenie przy testowaniu połączeń szyfrowanych).

Po stronie serwera do testów zostały wykorzystane dwa typy aplikacji. Pierwsza aplikacja (dalej nazywana aplikacją sesyjną) działała w następujący sposób:

1. Pobierała z żądania nazwę parametru oraz j ego wartość.
2. Wstawiała tę parę do sesji.
3. Zwracała wszystkie pary klucz-wartość znajdujące się w sesji.

Aplikacja sesyjna została napisana głównie w celu praktycznego udowodnienia poprawności działania klastra. Program testujący

generując kolejne żądania, odwołujące się do tej samej sesji, sprawdzał czy wynik żądania zgadza się z oczekiwaniami. Jeżeli w wyniku nie znajdował się jeden z dodanych przez niego parametrów, to podnoszony był wyjątek i test kończył się niepowodzeniem. Należy zwrócić uwagę na fakt, iż z wydajnościowego punktu widzenia aplikacja jest skrajnie pesymistycznym przypadkiem wykorzystania klastra. Przetworzenie żądania powoduje znikome obciążenie serwera, jednocześnie zmuszając klaster do replikowania wciąż rosnących sesji.

Druga aplikacja (dalej nazywana aplikacją XSL) działała w następujący sposób:

1. Przy pierwszym żądaniu ustawiała w sesji dwa parametry:
2. ścieżkę do pliku zawierającego dane zapisane w formacie XML,
3. ścieżkę do pliku zawierającego przekształcenie XSL prezentujące dane z pliku XML.
4. Przy drugim żądaniu aplikacja pobierała oba parametry z sesji i wykonywała przekształcenie XSL na danych z pliku XML.

Aplikacja nieco lepiej odwzorowuje praktyczne zastosowanie serwera Tomcat. Co prawda nie komunikuje się z bazą danych, co ma miejsce w większości prawdziwych zastosowań serwerów aplikacyjnych, niemniej jednak wykonuje obliczenia mające na celu uatrakcyjnienie wyświetlanych informacji. Wykorzystuje bardzo modne ostatnio technologie XML oraz przekształcenia XSL oddzielające warstwę danych od warstwy prezentacji.

Każdy z przeprowadzonych scenariuszy testowych był wykonywany kilkakrotnie w celu zwiększenia wiarygodności wyników. Tabele wynikowe prezentują uśredniony czas. Podczas badań system zachowywał się stabilnie – standardowe odchylenie oscylowało pomiędzy 5%, a 15%. Ponieważ różnice w wynikach były nieznaczne, nie zajmowałem się badaniem odchylenia samego w sobie.

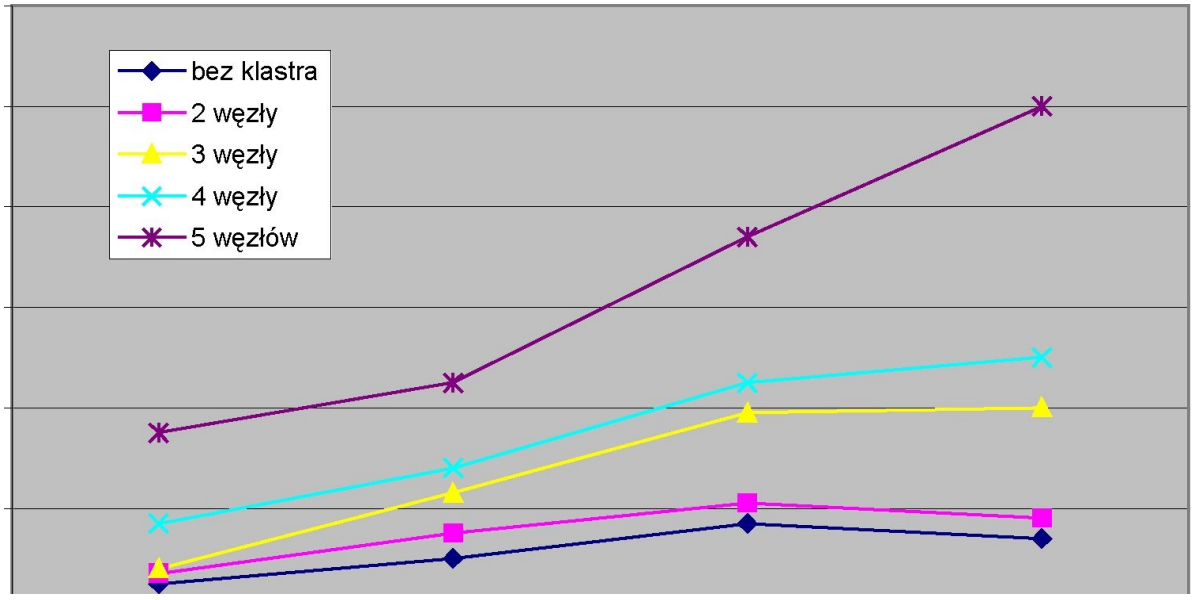
Test poprawności

Test poprawności został przeprowadzony na maszynach typu A. Program testujący generował kolejne żądania do aplikacji sesyjnej sprawdzając zgodność danych. Przy każdym żądaniu wielkość sesji była zwiększana o około 130 bajtów. Charakterystyka testu:

- Serwer zarządca znaj dowal się wewnątrz jednego z serwerów Tomcat.
- Przy każdym kolejnym żądaniu program testujący wybierał losowo jeden z węzłów, do którego przesyłał żądanie.

	30 w. 50 ż.	60 w. 50 ż.	90 w. 50 ż.	30 w. 100 ż.
bez klastra	5	10	17	14
2 węzły	7	15	21	18
3 węzły	8	23	39	40
4 węzły	17	28	45	50
5 węzłów	35	45	74	100

Rys. 5.1. Wykres wyniku testów przy losowym wyborze serwera



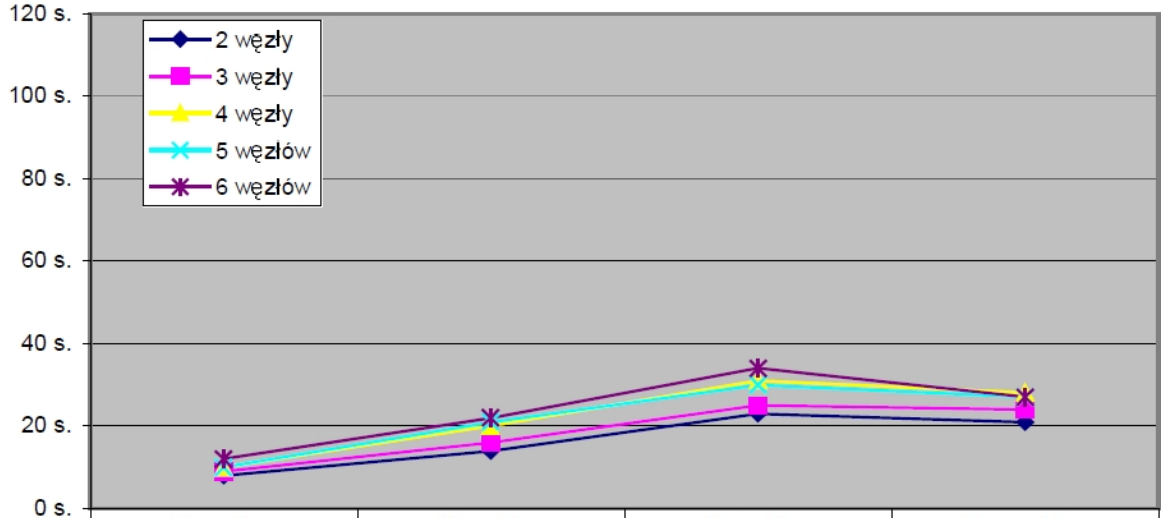
Wykres na rys. 5.1 prezentuje czasy trwania kolejnych testów mierzone w sekundach. Zapis „30 w. 50 ż.” oznacza, że zostało wygenerowanych 30 wątków, z których każdy wysyłał kolejno bez opóźnień 50 żądań, sukcesywnie zwiększających sesję (każdy wątek miał osobny identyfikator sesji).

Podstawowym zadaniem testu było pokazanie poprawności działania modułu co zostało osiągnięte. Mimo skrajnie trudnych warunków pracy w jakich przeprowadzono test (tzn. ciągła zmiana kontekstu wykonania żądań bez jakiegokolwiek opóźnienia) moduł działał poprawnie. Testy kończyły się z 99,9 procentową poprawnością. Przy bardzo dużym obciążeniu czasami zdarzało się, że klaster przekazywał błąd (status odpowiedzi na żądanie 500), który był spowodowany przekroczeniem maksymalnego czasu oczekiwania na zajęcie sesji. Niemniej jednak nie wystąpiła sytuacja, w której program testujący zaobserwowałby niezgodność danych – czyli moduł nie dopuścił do odczytu (modyfikacji) nieaktualnej sesji.

Jak można się było spodziewać zwiększanie liczby węzłów w klastrze powodowało zwiększanie czasu trwania testu. Nie jest to wielkim zaskoczeniem zważywszy na charakterystykę przeprowadzonego testu. Nie powinno również być wielkim zaskoczeniem, że czas trwania testu na pojedynczym serwerze był najniższy. Pojedynczy serwer odwołuje się do danych bezpośrednio w pamięci operacyjnej i nie musi ich przesyłać przez sieć. Jeżeli dodatkowo żądanie nie generuje prawie żadnego obciążenia na serwerze, to obsługa nawet kilku tysięcy żądań będzie trwała bardzo krótko.

Drugi test został przeprowadzony w bardzo podobnych warunkach, z tym że system wykorzystywał zintegrowany serwer ruchu. Każdy z wątków testujących łączył się z serwerem ruchu, który na podstawie przesyłanego identyfikatora sesji wybierał węzeł aktualnie zajmujący sesję. Każdy z węzłów miał ustawiony czas opóźnienia wysyłania replik oraz opóźnienia zwolnienia sesji na 2 sekundy.

Rys. 5.2. Wykres wyniku testów przy zastosowaniu zintegrowanego serwera ruchu



	30 w. 50 ż.	60 w. 50 ż.	90 w. 50 ż.	30 w. 100 ż.
2 węzły	8	14	23	21
3 węzły	9	16	25	24
4 węzły	10	20	31	28
5 węzłów	10	21	30	27
6 węzłów	12	22	34	27

Wykres na rys. 5.2 prezentuje wyniki testów mierzone w sekundach. Jak można zauważyć zastosowanie zintegrowanego serwera ruchu bardzo mocno polepszyło wydajność klastra. Prawdopodobnie zysk byłby nawet jeszcze większy przy odpowiednio dobrym zaimplementowaniu serwera ruchu. Z testów wynikało, że tak naprawdę wąskim gardłem tego testu mógł okazać się właśnie serwer ruchu. Niemniej jednak można zauważyć, że przy tej architekturze zwiększanie liczby węzłów w klastrze przestało negatywnie wpływać na wydajność systemu. W zasadzie różnice czasu po dodaniu kolejnych węzłów stają się nieznaczące.

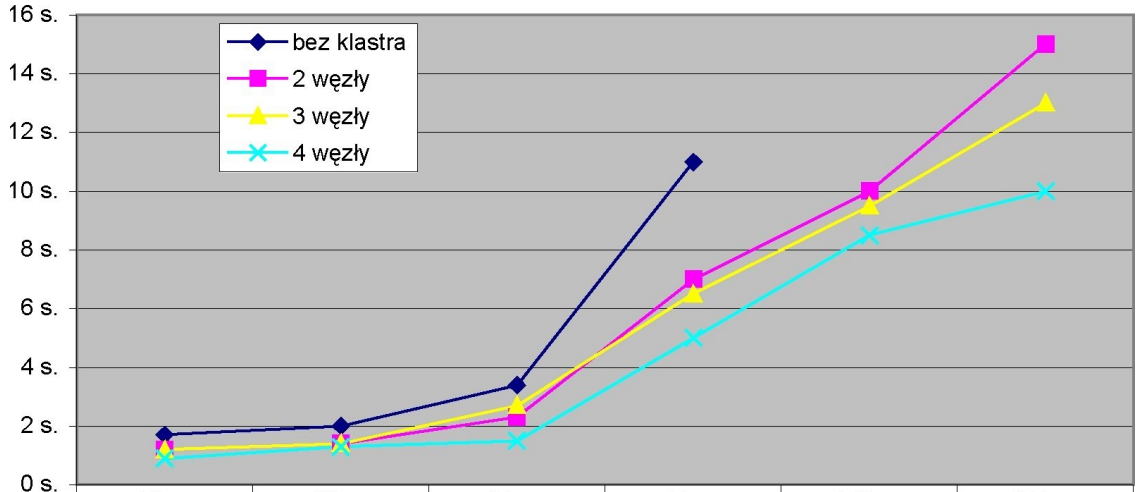
Test wydajności

Test wydajności został przeprowadzony na maszynach typu B. Każdy wątek testowy wykonywał pierwsze żądanie do aplikacji XSL w celu ustawienia w sesji ścieżek do plików. W kolejnym żądaniu odbierał rezultat wykonania przekształcenia na pliku XML. Należy zwrócić uwagę na fakt, że w przeprowadzanym teście

również były odwołania do replikowanych sesji, a nie tylko generowanie obliczeń. Charakterystyka testu:

- Serwer zarządca znaj dował się w j ednym z serwerów Tomcat.
- Testy były przeprowadzane bez użycia zintegrowanego serwera ruchu.
- Aplikacja testująca wybierała kolejne węzły w sposób losowy.

Rys. 5.3. Wyniki testów aplikacji XSL



	10 w.	20 w.	30 w.	90 w.	150 w.	210 w.
bez klastra	1,7	2	3,4	11		
2 węzły	1,2	1,4	2,3	7	10	15
3 węzły	1,2	1,4	2,7	6,5	9,5	13
4 węzły	0,9	1,3	1,5	5	8,5	10

Wykres na rys. 5.3 prezentuje wyniki testów. Jak można zaobserwować zysk z zastosowania klastra jest niebagatelny w stosunku do pracy pojedynczego serwera. Dodanie kolejnych węzłów obniża czas trwania testu, co jest szczególnie widoczne przy obsłudze bardzo wielu żądań jednocześnie. Zysk z zastosowania klastra najlepiej widać dla 150 oraz 210 wątków. Pojedynczy serwer Tomcat bez klastra w ogóle nie przeszedł testu – serwer przestał odpowiadać i trzeba było go zrestartować. Czyli zainstalowanie klastra pozwoliło na obsługę znacznie większej liczby klientów jednocześnie, co tak naprawdę jest najważniejszym wyznacznikiem wydajności systemu.

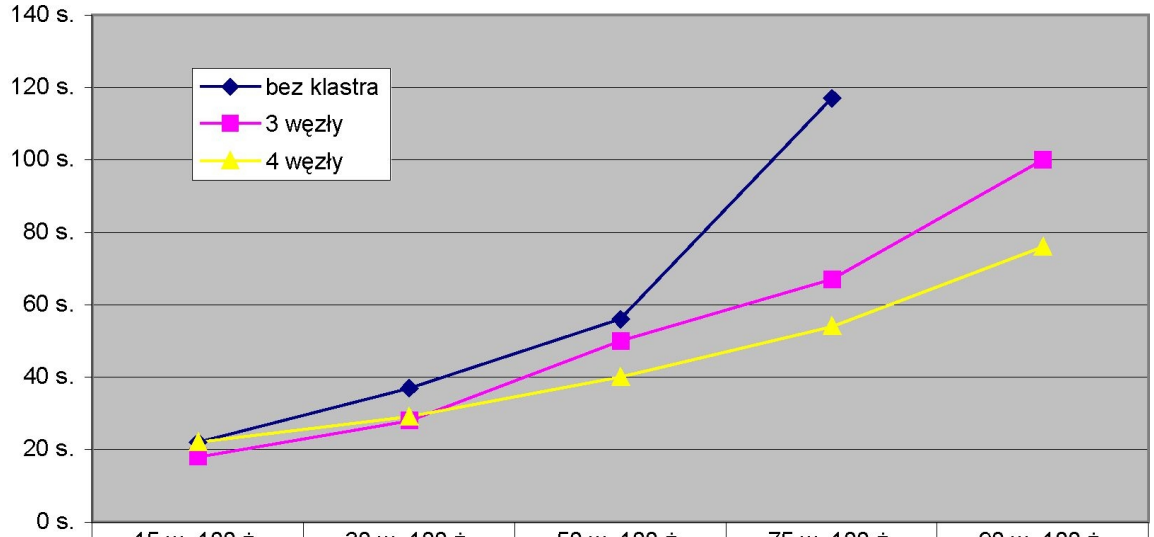
Prawdopodobnie zastosowanie dobrze zaimplementowanego serwera ruchu pozwoliłoby na uzyskanie jeszcze lepszych wyników, szczególnie przy większej liczbie węzłów.

Test HTTPS

Ostatni z testów miał na celu pokazanie skuteczności działania klastra w przypadku stosowania połączeń szyfrowanych. Obliczenia związane z deszyfrowaniem połączeń zostały rozrzucone na wszystkie węzły w klastrze (dokładniejszy opis konfiguracji znajduje się w p. 4.4.2). Do testów wykorzystano maszyny typu B. Charakterystyka testu:

- Serwer zarządca znaj dował się wewnątrz j ednego z serwerów Tomcat.
- Do testów wykorzystano aplikację sesyjną.

Rys. 5.4. Wyniki testów przy wykorzystaniu połączeń szyfrowanych



	15 w. 100 ż.	30 w. 100 ż.	50 w. 100 ż.	75 w. 100 ż.	90 w. 100 ż.
bez klastra	22	37	56	117	
3 węzły	18	28	50	67	100
4 węzły	22	29	40	54	76

Wykres na rys. 5.4 prezentuje wyniki przeprowadzonych testów. Co ciekawe przy wykorzystaniu połączeń szyfrowanych okazuje się, że zysk z zastosowania klastra osiąga się nawet dla skrajnie pesymistycznych przypadków z punktu widzenia klastra

(aplikacja sesyjna). Obliczenia związane z szyfrowaniem i deszyfrowaniem są tak kosztowne, że złączenie wielu fizycznych maszyn rekompensuje straty związane z przesyłaniem replik sesji w sieci. Jak widać na wykresie pojedynczy serwer jest dużo wolniejszy od klastra, a szczególnie wyraźnie widać to przy dużym obciążeniu. Podczas testu z 90 wątkami serwer przestał odpowiadać i trzeba było go zrestartować.

Warto zwrócić uwagę na kształtowanie się różnic czasów pomiędzy klastrem złożonym z 3 węzłów a klastrem złożonym z 4 węzłów. Dla małej liczby równoległych żądań (15 w. 100 ż.) klaster z trzema węzłami okazuje się szybszy od tego z 4. Przy małym obciążeniu systemu wąskim gardłem staje się strata związana z przesyłaniem replik w sieci. Natomiast wraz ze wzrostem liczby równoległych żądań zysk z doinstalowania kolejnego węzła staje się coraz bardziej widoczny. Jest to bardzo ważna obserwacja z punktu widzenia administratora systemu. Zwiększenie liczby węzłów w klastrze musi iść w parze ze zwiększeniem liczby użytkowników systemu; w przeciwnym przypadku dodanie węzła spowoduje spadek wydajności systemu.

Test został przeprowadzony dla aplikacji sesyjnej, aby uwidocznić zysk jaki można osiągnąć przy instalacji klastra do obsługi połączeń szyfrowanych. Jeżeli test zostanie przeprowadzony dla bardziej wymagającej aplikacji, to można oczekiwać, że wyniki będą jeszcze korzystniejsze.