STUDIA INFORMATICA 2014
Volume 35 Number 3 (117)

Krzysztof HABRAT, Magdalena ŁADNIAK, Zdzisław ONDERKA
AGH Akademia Górniczo–Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej

ANALIZA WYDAJNOŚCI SERWISU INTERNETOWEGO OPARTEGO NA MODELU CHMURY OBLICZENIOWEJ[1]

Streszczenie. W niniejszej pracy zaprezentowano możliwości wykorzystania oprogramowania opartego na modelu przetwarzania w chmurze, w charakterze skalowalnego serwera aplikacji internetowej. Omówiono techniczne aspekty możliwości wykorzystania systemu EUCALYPTUS jako serwera aplikacji internetowej. Przedstawione zostały wyniki testów wydajnościowych przykładowego systemu, zarówno w kontekście wykorzystania różnej liczby i konfiguracji zasobów obliczeniowych, jak i możliwości wykorzystania oprogramowania do równoważenia obciążenia *Apache mod_proxy_balancer*.

Słowa kluczowe: chmura obliczeniowa, maszyna wirtualna, skalowalność, równoważenie obciążenia

EFFICIENCY ANALYSIS OF WEB APPLICATION BASED ON CLOUD SYSTEM

Summary. This paper presents the possibilities of using software based on the model of cloud computing as a scalable web application server. In this paper a performance issue for scalable Cloud Computing based systems was analyzed in the context of load balancing between the different configurations of resources. An application of the EUCALYPTUS system as well as the results of the performance tests, both for the case of the use of virtual machines with different number of resources and the use of Apache mod_proxy_balancer software were presented.

K**eywords**: cloud computing, virtual machine, scalability, load balancing

[1] Praca finansowana w ramach badań statutowych Katedry Geoinformatyki i Informatyki Stosowanej

1. Wstęp

Intensywny rozwój potrzeb informatyzacji przedsiębiorstw oraz ciągły wzrost konkurencyjności wymógł na dostawcach usług w zakresie IT konieczność oferowania optymalnych i tanich usług [1]. Rośnie popyt na nowe rozwiązania, które zoptymalizowałyby koszty, takie jak np. wirtualizacja danych [2]. Model chmury obliczeniowej (ang. *Cloud*) okazał się tańszym rozwiązaniem aniżeli znany dotychczas sposób rozszerzania funkcjonalności oraz użytkowania aplikacji jako usług IT. W ramach modelu chmury używane jest pojęcie „przetwarzania w chmurze" (ang. *Cloud Computing*) oraz „usługi w chmurze" (ang. *Cloud Services*). Z technicznego punktu widzenia jest to połączenie metod projektowania rozproszonych systemów, polegających na przetwarzaniu klastrowym i usługowym [1, 3]. Jedną z podstawowych cech tego typu systemów, wpływającą na jakość ich obsługi i działania, jest skalowalność systemu. Przez to pojęcie rozumiana jest łatwość w modyfikowaniu kolejnych zasobów (obliczeniowych, użytkowników) [4].

Skalowalność to miara możliwości rozbudowy komputera, usługi lub aplikacji stosownie do rosnących wymagań w zakresie wydajności. W przypadku klastrów-serwerów skalowalność określa możliwość stopniowego dodawania systemów do już istniejącego, wówczas gdy całkowite obciążenie przekracza jego możliwości [4]. Obsługa bardzo wielu zapytań na jednostkę czasu sprawia, iż pojedyncze serwery, nawet bardzo wydajne, nie radzą sobie z takim obciążeniem. Wówczas możliwe jest wykorzystanie technik rozpraszania obciążenia pomiędzy wiele zasobów obliczeniowych.

W niniejszym artykule zbadano możliwości wykorzystania oprogramowania, opartego na modelu przetwarzania w chmurze, w kontekście skalowalnego serwera aplikacji typu klient - serwer. Przedstawiono wyniki jego wydajności pod kątem liczby obsłużonych żądań użytkowników w jednostce czasu, zmierzone za pomocą dedykowanego do takich celów oprogramowania Apache JMeter. Ponadto, podano uśredniony minimalny czas odpowiedzi serwera zarówno dla przypadku wykorzystania zasobów w postaci maszyn wirtualnych z różną liczbą procesorów, różną ilością pamięci RAM oraz różną pojemnością dysku twardego, jak i użycia oprogramowania do równoważenia obciążenia.

2. Konstrukcja serwisu internetowego opartego na modelu przetwarzania w chmurze

Zastosowanie modelu systemu wykorzystującego ideę przetwarzania w chmurze pozwala na uniezależnienie się od fizycznej architektury zasobów obliczeniowych oraz związanych z nią ograniczeń. Zagadnienia budowy takich systemów oraz zagadnienia wykorzystania ich

zasobów jest w literaturze szeroko opisywaną tematyką, z tego względu w niniejszym artykule pominięto jej szczegółowy opis, odwołując się do istniejących publikacji [1, 3, 5, 6, 7].

2.1. Opis wykorzystanego oprogramowania serwerowego

W celu konstrukcji omawianego przykładu niezbędny był wybór oprogramowania pozwalającego na stworzenie systemu wirtualizacji oraz administracji dostępnymi fizycznie zasobami. Wybrano w tym celu system EUCALYPTUS. Dobór bazował zarówno na odpowiednim dopasowaniu sprzętowym (opisanym w podpunkcie 3.1), jak również na spełnieniu wymagań funkcjonalnych w zakresie realizacji celów postawionych w ramach niniejszej pracy.

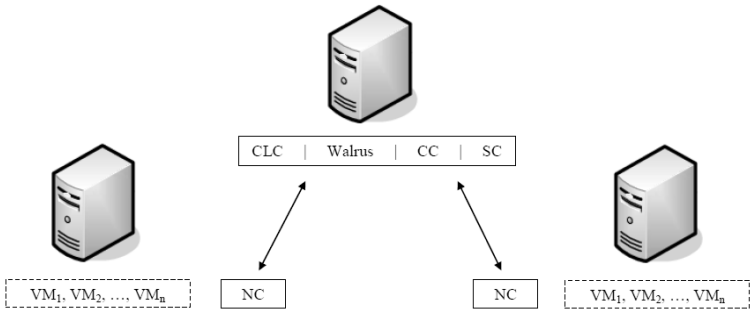
System EUCALYPTUS jest otwartym systemem informatycznym umożliwiającym zbudowanie prywatnej lub hybrydowej chmury obliczeniowej typu IaaS [8]. Odbywa się to poprzez zamianę fizycznej infrastruktury sprzętowej w możliwą do udostępnienia pulę zasobów, które po przydzieleniu do maszyny wirtualnej mogą być dynamicznie skalowane w górę lub dół [8, 9]. Architekturę systemu EUCALYPTUS cechuje modułowość, co daje jego większą uniwersalność i daje możliwość rozbudowy funkcjonalności wraz z rozwojem projektu. Budowa systemu opiera się na pięciu podstawowych komponentach, z których każdy posiada własne interfejsy typu Web Service, dokładnie określone komendy oraz posiada własne miejsce w fizycznej strukturze zasobów [8, 9]. Składowymi są kontroler chmury (CLC), kontroler klastra (CC), komponent zarządzania pamięcią (Walrus), kontroler magazynowania (SC), kontroler węzła (NC). Poszczególne moduły komunikują się ze sobą, tworząc jedno zintegrowane, rozproszone środowisko. Każdy z kontrolerów jest dedykowany do realizacji zdefiniowanych dla niego funkcjonalności [9].

2.2. Opis infrastruktury proponowanego podejścia

Aplikacja systemu testowego opierała się na wykorzystaniu trzech fizycznych jednostek komputerowych, każda o takich samych parametrach sprzętowych. Wykorzystano w tym celu trzy komputery IBM Blade HS21, połączone siecią Ethernet 1 Gb, o następujących parametrach każdy: CPU Intel Xeon 2.0 GHz (8 rdzeni), RAM 16 GB DDR-2, HDD SAS 73GB. Komputery pracowały na systemach operacyjnych: CentOS w wersji 6.x o architekturze x86_64 (komputer główny) oraz dla węzłów, a zatem platform udostępnianych potencjalnym klientom, Linux - CentOS o architekturze i386/x86_64.

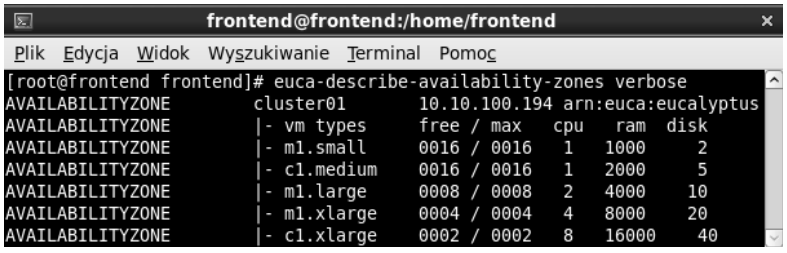
Na pierwszej fizycznej jednostce zainstalowano wszystkie komponenty główne systemu EUCALYPTUS. Dwa kolejne komputery przeznaczone zostały do instalacji tzw. węzłów, czyli zainstalowano na nich komponent systemu EUCALYPTUS, odpowiadający za umożli-

wienie wirtualizacji zasobów (rys. 1). W takiej konfiguracji maksymalna pula zasobów możliwa do udostępnienia klientom przez chmurę, to sumaryczna liczba zasobów dwóch fizycznych jednostek. Dzięki modułowości systemu EUCALYPTUS instalacja poszczególnych komponentów na fizycznych zasobach jest nieobowiązkowa, przez co pozwala na personalizację oczekiwań od systemu.



Rys. 1. Schemat zainstalowanego systemu wraz z podziałem komponentów pomiędzy jednostki
Fig. 1. Schematic installed system with the distribution of components between individuals

Udostępnienie przez system określonej ilości zasobów obliczeniowych rozumiane jest w kontekście niniejszego artykułu jako stworzenie instancji maszyny wirtualnej z przygotowaną liczbą zasobów. Każda instancja, podczas jej tworzenia, powinna mieć zdefiniowany typ, który będzie jednoznacznie definiował, jakie zasoby sprzętowe należy przydzielić nowo uruchomionej maszynie wirtualnej. System EUCALYPTUS domyślnie posiada 5 typów maszyn wirtualnych: *m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge*. Parametry, które będą cechowały dany typ instancji, można definiować w sposób niemal dowolny. Istotnie obowiązującą zasadą jest tutaj fakt, iż każdy „wyższy" stopniem typ musi posiadać wszystkie trzy parametry (liczba rdzeni procesora, ilość pamięci RAM oraz ilość przestrzeni dyskowej) większe lub równe od swojego poprzednika.



Rys. 2. Przykład odpowiedzi serwera na pytanie o dostępne zasoby
Fig. 2. Example of server response to a question about the available resources

Po spełnieniu minimalnych wymagań sprzętowych stawianych przez oprogramowanie [8] oraz po uprzednim rozplanowaniu budowy systemu i jego komponentowego podziału istotne jest, by dokonać synchronizacji zegarów wszystkich jednostek, umożliwić udostępnianie połączeń pomiędzy maszynami, ustalić otwarcie portu 8775 dla każdego węzła, otwarcie portów 8443, 8773, 8774 dla każdego serwera oraz dokonać konfiguracji mostka jako głównego interfejsu sieciowego dla każdego węzła. Od tego stanu, po zatwierdzeniu konfiguracji, moż-

na w dowolnym momencie działania systemu wysłać zapytanie o dostępne zasoby. System zwróci wynik w postaci liczby możliwych do uruchomienia maszyn wirtualnych według zdefiniowanych typów (rys. 2).

2.3. Idea równoważenia obciążenia serwisu internetowego

Istotnym aspektem projektowania modelu aplikacji internetowej dla rozproszonego systemu jest problem przydziału zadań do poszczególnych węzłów systemu [10]. Wyróżnia się dwa podejścia do tego problemu. Pierwszym jest dzielenie obciążania (ang. *load sharing*) – nowe zadanie jest uruchamiane na jednostce najmniej obciążonej w danym momencie. Drugim jest (zaaplikowane w niniejszej pracy z wykorzystaniem produktu Apache mod_proxy_balancer) równoważenie obciążenia (ang. *load balancing*) pomiędzy wszystkie jednostki. W praktyce, żądanie od klienta trafia do zarządcy klastra, który to decyduje, któremu z dostępnych węzłów należy przekazać żądanie [11, 12].

Znane są trzy algorytmy służące do równoważenia obciążenia [11, 13]: *Request Countin, Weighted Traffic Counting, Pending Request Countin*. Różnice pomiędzy nimi dotyczą metody dostarczania liczby zapytań do różnych węzłów. Pierwszy z nich najpierw sprawdza, która z maszyn posiada największe priorytety wykonania zadania, i to właśnie jej przydziela nadchodzący ruch. *Weighted Traffic Counting* określa rozmiar ruchu w bajtach, z jakim powinny sobie radzić węzły. Następnie bierze pod uwagę natężenie ruchu, które dociera do węzła, zamiast liczby zapytań do niego docierających. Trzeci algorytm - *Pending Request Counting* działa na zasadzie śledzenia liczby zapytań obsługiwanych w danym momencie przez węzły. Nowe zapytanie jest przypisywane do serwera, który ma najmniejszą liczbę aktywnych żądań. Poniżej zaprezentowano szczegóły aspektów wydajnościowych wykorzystania poszczególnych metod.

2.4. Testowana aplikacja internetowa

Dla realizacji badań wykorzystano serwis internetowy stworzony w technologii *Java EE*. Aplikacja testowa zawierała implementację arytmetycznych operacji przetwarzania obrazów cyfrowych, które sprowadza się do mnożenia dwóch macierzy. Wybór ten został podyktowany własnościami, jakie ma taki kod, a mianowicie determinizmem jego wykonania. Czas pojedynczego zadania zależy wówczas jedynie od przydziału zasobu procesora i zależy proporcjonalnie od kodu wykonywanego. Nie pojawiają się oczekiwania na pozostałe zasoby, takie jak dostęp do systemu plików (dysk twardy) czy inne, pochodzące od sprzętu lub systemu operacyjnego, o charakterze silnie niedeterministycznym.

Taki dobór został przedstawiony i omówiony w pracy [13]. Rozmiar macierzy dobrano w sposób nieprzypadkowy, tak aby czas odpowiedzi serwera dla pojedynczego żądania użytkownika dla pojedynczej maszyny jednoprocesorowej wynosił ok. 450 ms. Wymagania te spełniła macierz kwadratowa o rozmiarze 420 x 420. Liczba żądań wykonania przekształceń algebraicznych (tj. mnożenia macierzy) stała się podstawą dalszych rozważań wydajnościowych. Aplikacja, w każdym przypadku testowym, uruchomiona została na serwerze aplikacji internetowych *Apache Tomcat* w wersji 6.0.

3. Ocena wydajności systemu

Autorzy wyróżnili dwa czynniki, mogące wpływać na wydajność proponowanego rozwiązania. Pierwszym z nich jest liczba wykorzystanych maszyn wirtualnych oraz przydzielonych im zasobów obliczeniowych. Drugim czynnikiem jest proporcja, według której rozmieszczone zostały zasoby obliczeniowe dla poszczególnych maszyn wirtualnych o charakterze serwerowym. Różne konfiguracje ich wykorzystania stały się przedmiotem dalszych badań.

W poniższych zestawieniach dokonano porównania wydajności serwisu, mierzonej jako liczba obsłużonych żądań klientów w czasie jednej minuty (w tym przypadku żądaniem było wykonanie obliczenia mnożenia macierzy o rozmiarze 420x420, podpunkt 2.2). Obserwowano również wartości minimalnego czasu odpowiedzi serwera, czyli minimalnego czasu obsługi pojedynczego żądania, jednak ze względu na fakt, iż dla wszystkich analizowanych konfiguracji zasobów utrzymywał on średnią wartość równą 430 ms, z odchyleniem standardowym równym 9,8 ms, nie umieszczano jego szczegółowych zestawień w dalszych rozważaniach. Jako narzędzie pomiarowe wykorzystano aplikację Apache JMeter (aplikację typu *'open-source'* przystosowaną do testowania zachowania aplikacji internetowych).

3.1. Zestawienie wykorzystanych konfiguracji zasobów obliczeniowych

Podczas wykonywania testów poszczególne zasoby obliczeniowe, przestrzeń dyskowa oraz ilość pamięci RAM rozdzielane były pomiędzy maszyny wirtualne w różnych kombinacjach, z zachowaniem proporcji według opisanych w tabeli 1. W praktyce oznacza to, iż za równoważne uznano zasoby wykorzystane w dwóch przykładowych sytuacjach użycia: 4 VM x 2 CPU (16000 RAM, 40 GB HDD) oraz 1 VM x 8 CPU (16000 RAM, 40GB HDD).

Tabela 1

Analizowane konfiguracje zasobów maszyn wirtualnych

Liczba rdzeni procesora	1	2	3	4	5	6	7	8
Pamięć RAM [MB]	2000	4000	6000	8000	10000	12000	14000	16000
Przestrzeń dyskowa [GB]	5	10	15	20	25	30	35	40

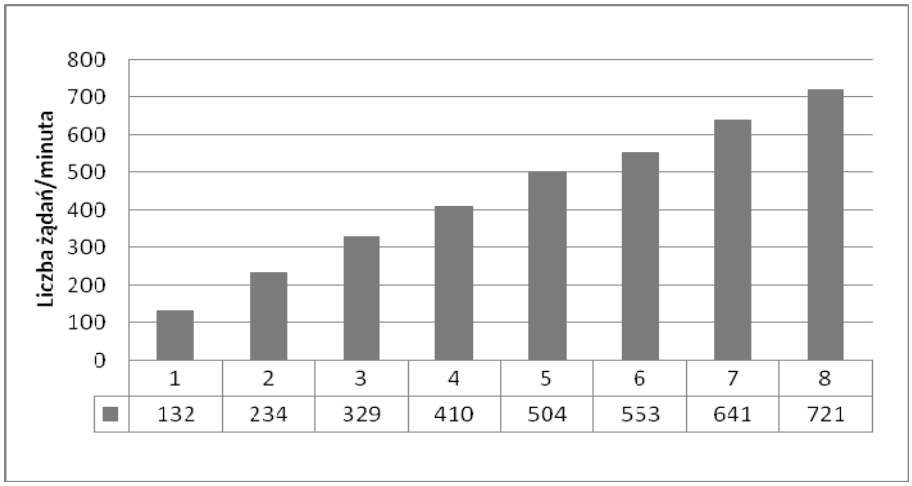
Kombinację możliwej do wykorzystania liczby rdzeni procesora oraz powołanych maszyn wirtualnych zdefiniowano według wzoru (1):

$$O = M * N \tag{1}$$

gdzie*:* O oznacza sumaryczną liczbę wykorzystanych rdzeni procesora, M oznacza liczbę wykorzystanych maszyn wirtualnych, N oznacza liczbę rdzeni procesora przydzielonych maszynie wirtualnej. Istota każdego z wykonanych testów regulowana jest założeniami, iż w przypadku badania wydajności aplikacji, w zależności od ilości użytych zasobów, zaleca się ustalenie O, M, N jako liczby zmienne oraz M, N jako całkowite podzielniki liczby O. Podczas badania wydajności aplikacji w zależności od kombinacji przydziału zasobów dla maszyn wirtualnych ustala się O jako liczbę stałą oraz M, N jako liczby zmienne, całkowite podzielniki liczby O.

3.2. Ocena wydajności serwisu wykorzystującego pojedynczą instancję maszyny wirtualnej

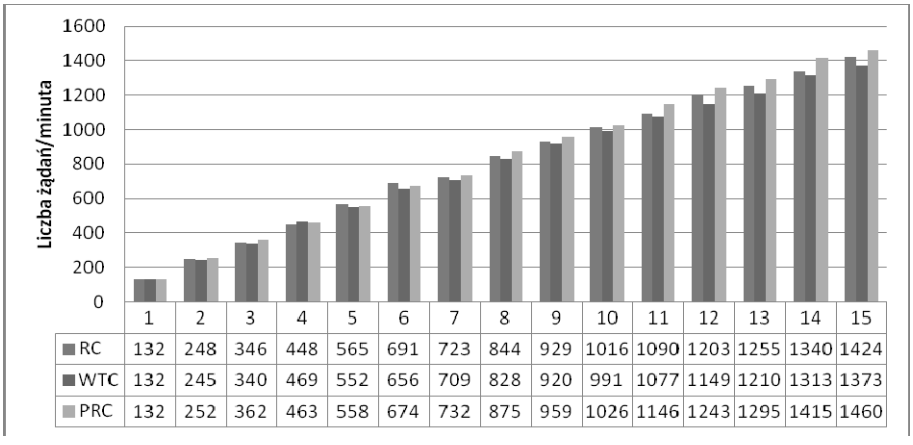
Pierwszym z wykonanych testów było porównanie wydajności serwisu internetowego w zależności od liczby przydzielonych zasobów dla jednej maszyny wirtualnej. Z budowy systemu EUCALYPTUS i jego komponentów wiadomo, iż instancje uruchamiane są na komputerach węzłów, a co za tym idzie, wykorzystywane przez nie zasoby są ograniczone z góry budową fizyczną komputera. Maksymalna ilość pamięci, liczba rdzeni procesora oraz ilość przestrzeni dyskowej, jaką można przydzielić jednej maszynie wirtualnej, zamyka się w ilości udostępnianej przez jeden fizyczny węzeł. Nie ma możliwości uruchomienia instancji z rozproszonym systemem operacyjnym, korzystającej z zasobów kilku węzłów jednocześnie. Z tego powodu maksymalny rozmiar badanej maszyny wirtualnej wynosił: 8 CPU, 16000 MB RAM, 40 GB HDD. Na rys. 3 przedstawiono wzrost maksymalnego obciążenia serwisu, jakie jest serwer w stanie obsłużyć, dla manualnie zwiększanej liczby zasobów (według proporcji z tabeli 1). Dla maszyny ośmioprocesorowej w stosunku do maszyny jednoprocesorowej wydajność ta jest ponadpięciokrotnie wyższa.



Rys. 3. Wydajność serwera dla pojedynczych instancji (oś x - liczba rdzeni procesora, oś y - miara wydajności (liczba żądań/minuta))
Fig. 3. Server efficiency for a single instance (x-axis - number of processor cores, y axis - efficiency (number of requests/minute))

3.3. Ocena wydajności serwisu w zależności od ilości przydzielonych zasobów

W przypadku wykorzystania więcej niż jednej maszyny wirtualnej do stworzenia serwera serwisu internetowego zastosowano oprogramowanie do równoważenia obciążenia. Pomiary wykonywane były w konfiguracjach maszyn wirtualnych zaprezentowanych w podpunkcie 3.1. W pierwszej kolejności dokonano porównania zmienności obciążenia serwera wraz ze zwiększaniem wykorzystanej liczby jednoprocesorowych maszyn wirtualnych (rys. 4, tabela 2), z uwzględnieniem rozróżnienia omówionych w podpunkcie 2.3 różnych algorytmów równoważenia obciążenia.



Rys. 4. Wydajność serwera dla wielu instancji jednoprocesorowych (oś x - liczba rdzeni procesora, oś y - miara wydajności (liczba żądań/minuta))
Fig. 4. Server efficiency for multiple single-instances of virtual machines (x-axis - number of processor cores, y axis - efficiency (number of requests / minute))

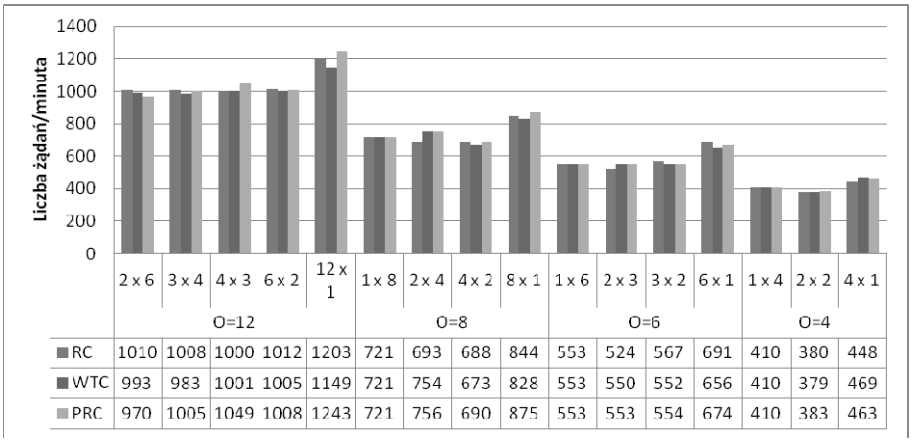
Tabela 2

Miara wydajności serwera (liczba żądań/minuta) w zależności od liczby powołanych maszyn wirtualnych (M) oraz liczby przyporządkowanych im rdzeni procesora (N) dla trzech algorytmów równoważenia obciążenia (RC, WTC, PRC)

N	**2**							**3**				**4**		
M	1	2	3	4	5	6	7	1	2	3	4	1	2	3
RC	234	380	567	688	877	1012	1065	329	524	769	1000	410	693	1008
WTC	234	379	552	673	851	1005	1082	329	550	734	1001	410	754	983
PRC	234	383	554	690	861	1008	1079	329	553	755	1049	410	756	1005

3.4. Ocena wydajności serwisu w zależności od kombinacji konfiguracji przydzielonych zasobów

Innym aspektem oceny wydajności serwisu internetowego jest kombinacja, według której rozmieszczone zostały zasoby obliczeniowe dla poszczególnych maszyn wirtualnych. Autorzy, ze względu na ilość dostępnych zasobów, rozpatrzyli cztery przypadki: całkowity zasób dwunastu, ośmiu, sześciu oraz czterech procesorów. Szczegółowe zestawienie wszystkich kombinacji konfiguracji zasobów zaprezentowano na rys. 5.

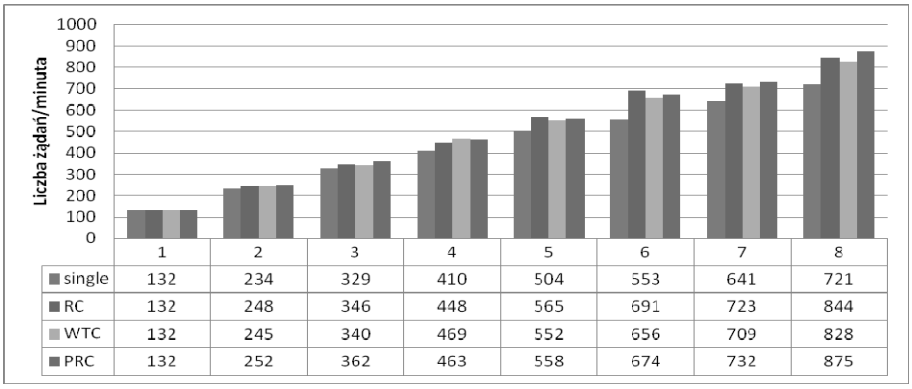


Rys. 5. Wydajność serwera dla różnych kombinacji konfiguracji zasobów (oś x – suma wykorzystanych rdzeni procesora, oś y - miara wydajności (liczba żądań/minuta))
Fig. 5. Server efficiency for different combinations of bandwidth resource configuration (x-axis - sum of the processor cores, y axis - efficiency (number of requests / minute))

3.5. Porównanie otrzymanych wyników

Poniżej zaprezentowano porównanie przepustowości dla rozwiązań opartych na wykorzystaniu wielu instancji maszyn jednoprocesorowych oraz wykorzystania oprogramowania do równoważenia obciążenia (rys. 6). Na osi poziomej umieszczono sumaryczną liczbę wykorzystanych rdzeni procesora. Określenia RC, WTC, PRC dotyczą algorytmów równoważenia obciążenia, określenie „single" dotyczy wykorzystania jednej instancji maszyny wirtualnej. Rezultaty przeprowadzonych testów wykazują przewagę wykorzystania oprogramowania do równoważenia obciążenia w kontekście wydajności serwisu. Zestawienie obrazuje ogólny

wniosek, iż maszyny jednoprocesorowe połączone *load balancerem* wykazują się większą wydajnością od maszyn wieloprocesorowych (niezależnie od użytego algorytmu).



	1	2	3	4	5	6	7	8
single	132	234	329	410	504	553	641	721
RC	132	248	346	448	565	691	723	844
WTC	132	245	340	469	552	656	709	828
PRC	132	252	362	463	558	674	732	875

Rys. 6. Porównanie wydajności dla rozwiązań opartych na wykorzystaniu jednej instancji maszyny wirtualnej oraz wykorzystaniu oprogramowania do równoważenia obciążenia (oś x: suma wykorzystanych rdzeni procesora, oś y: miara wydajności (liczba żądań/minuta))

Fig. 6. Comparison of server efficiency for solutions based on the use of single instance of virtual machine and single-use load balancing software (x-axis - sum of the processor cores, y axis - efficiency (number of requests / minute))

4. Podsumowanie

Celem niniejszej pracy było porównanie i przedstawienie wydajności aplikacji internetowej, opierającej się na modelu przetwarzania w chmurze w zależności od dwóch czynników: skali technicznego rozwiązania systemu (co oznacza liczbę użytych maszyn wirtualnych oraz przydzielonych im zasobów obliczeniowych) oraz konfiguracji, według której rozmieszczone zostały zasoby obliczeniowe dla poszczególnych maszyn wirtualnych (które pracowały w charakterze serwerowym). Przedstawiono metodę wykorzystania chmury obliczeniowej do stworzenia wydajnego, otwartego i rozszerzalnego serwisu internetowego. Opisano szczegóły instalacji i wykorzystania systemu EUCALYPTUS, za pomocą którego działał system internetowy, bazujący na instancjach maszyn wirtualnych. Zaprezentowano wyniki jego testów wydajnościowych, co umożliwia dokonanie wyboru optymalnego rozwiązania podczas planowania konstrukcji analogicznego schematycznie systemu.

Wyniki wydajności (liczby żądań/minuta) oraz minimalnego czasu odpowiedzi porównane zostały zarówno dla przypadku użycia pojedynczych maszyn wirtualnych z różną liczbą przydzielonych procesorów, ilością pamięci RAM oraz pojemnością dysku twardego, jak i wykorzystania oprogramowania *Apache mod_proxy_balancer* do równoważenia obciążenia dla trzech jego algorytmów: *Request Counting, Weighted Traffic Counting oraz Pending Request Counting*.

Zestawienie ze sobą wyników różnych konfiguracji wykazało stosunkowo silną przewagę użycia wielu instancji maszyn wirtualnych jednoprocesorowych oraz *load balancera*, jako narzędzia do rozdzielania zadań pomiędzy dostępne procesory, nad inne konfiguracje, opierające się na maszynach wirtualnych z dwoma lub większą ilością rdzeni CPU. Wykazano, iż wykorzystanie oprogramowania do przetwarzania w chmurze, jako skalowalnego serwera aplikacji internetowej, jest korzystnym rozwiązaniem z punktu widzenia wydajności.

BIBLIOGRAFIA

1. Kandziora P., Kozielski S.: Silesian Cloud-koncepcja budowy chmury w modelu IaaS. Studia Informatica, vol.32 No. 3A, Wyd Pol. Śl., Gliwice 2011.
2. Trojnar D.: Wirtualizacja jako przyszłość sieci teleinformatycznych. SECON – Materiały konferencyjne, WAT, Warszawa 2010.
3. Jestratjew A., Kwiecień A.: Using Cloud Storage in Production Monitoring Systems. Computer Networks. Springer, CCIS vol. 79, Berlin Heidelberg 2010.
4. Zieliński K.: Zagadnienia konstrukcji oprogramowania komponentowego. Problemy i metody inżynierii oprogramowania, 327–343.
5. Bajerski P.: Bazy danych a chmury obliczeniowe. Studia Informatica vol. 33 (2A), Wyd Pol. Śl., Gliwice 2012.
6. Schäffer B., Baranski B., Foerster T.: Towards spatial data infrastructures in the clouds. Geospatial Thinking, Springer, Berlin Heidelberg 2010.
7. Augustyn D., Warchał Ł.: Cloud Service Solving N-Body Problem Based on Windows Azure Platform. Computer Networks. Springer, CCIS vol. 79, Berlin Heidelberg 2010.
8. Dokumentacja techniczna systemu EUCALYPTUS, www.eucalyptus.com.
9. Nurmi D., Wolski R., Grzegorczyk C., Obertelli G., Soman S., Youseff L., Zagorodnov D.: The eucalyptus open-source cloud-computing system. In Cluster Computing and the Grid, CCGRID'09. 9th IEEE/ACM International Symposium, IEEE, 2011, s. 124–131.
10. Czerwiński D.: Wpływ zarządcy maszyny wirtualnej na wydajność systemów eksploracji danych. Studia Informatica, vol. 33(3A), Wyd Pol. Śl., Gliwice 2012.
11. Piórkowski A., Kempny A., Hajduk A., Strzelczyk J.: Load Balancing for Heterogeneous Web Servers. Computer Networks. Springer, Berlin Heidelberg 2010, s. 189–198.
12. Strzelczyk J., Piórkowski A.: Metody równoważenia obciążeń w serwisach internetowych, Współczesna problematyka sieci komputerowych. Wydawnictwa Komunikacji i Łączności, Warszawa 2010.
13. Nowak Z., Porczyński R.: Modele funkcjonalne systemów do równoważenia obciążeń w serwisach WWW. Studia Informatica, vol. 22 (3), Wyd. Pol. Śl., Gliwice 2001.

Wpłynęło do Redakcji 9 kwietnia 2014 r.

Abstract

This paper discusses the performance issues of a scalable system based on the model of Cloud Computing in the context of load balancing between the different configurations of resources. A method of using Cloud Computing to create an expandable Web Service was shown. An application of the EUCALYPTUS system was described. The results of the performance tests were presented, which allows to select the optimal solution when planning the construction of a similar system. The results for both cases, the use of virtual machines with different numbers of cores, RAM memory and hard disk capacity, and the use of Apache mod_proxy_balancer software for load balancing, were compared. Relatively strong advantage was showed to use multiple single-core instances of virtual machines and a load balancer as a tool for distributing tasks among the available processors, on the other configurations, based on virtual machines with two or more CPU cores.

Adresy

Krzysztof HABRAT: AGH Akademia Górniczo–Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30–059 Kraków, Polska, krzysztofhabrat@yahoo.pl

Magdalena ŁADNIAK: AGH Akademia Górniczo–Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30–059 Kraków, Polska, mladniak@geol.agh.edu.pl

Zdzisław ONDERKA: AGH Akademia Górniczo–Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30–059 Kraków, Polska, zonderka@agh.edu.pl