1. Optymalizacja równoważenia obciążeń w systemach klasy grid

Jerzy Balicki, Tomasz Boiński, Waldemar Korłub, Jacek Paluszak, Adam Polak

Politechnika Gdańska,

Wydział Elektroniki, Telekomunikacji i Informatyki,

Katedra Architektury Systemów Komputerowych

e-mail: balicki@eti.pg.gda.pl

Streszczenie

W rozdziale zaprezentowano techniki równoważenia obciążeń testowane w systemie rozproszonym Comcute o architekturze typu grid. Omówiono niezbędne uwarunkowania środowiska prowadzenia obliczeń w zestawie laboratoryjnym Politechniki Gdańskiej, a także odniesiono się do kryteriów kierunkujących równoważenie obciążeń. Przedstawiono wielopoziomową metodę równoważenia obciążeń.

Słowa kluczowe: równoważenie obciążeń, systemy klasy grid, systemy rozproszone.

1.1. Środowisko do optymalizacji obciążenia w zestawie laboratoryjnym PG

Optymalizacja obciążenia w systemie Comcute polega na przydziale aplikacji warstw *Z, W* i *S* do węzłów obliczeniowych w taki sposób, aby nie tylko wykorzystać specyfikę przetwarzania poszczególnych modułów pod kątem skrócenia czasu obliczeń, ale także w celu równoważenie obciążeń. W architekturze laboratoryjnego zestawu systemu Comcute skonstruowanego na Politechnice Gdańskiej w Katedrze Architektury Systemów Komputerowych WETI przewidziano 9 węzłów obliczeniowych, macierz dyskową i lokalną sieć komputerową Gigabit Ethernet. Węzły od nr 1 do 7 dedykowane są do eksploatacji głównego prototypu programistycznego napisanego w technologii programistycznej JEE.

W systemie Comcute JEE każdy węzeł obliczeniowy zawiera dwa procesory po 6 rdzeni. 2 węzły wyposażone są w pamięć RAM o wielkości 24 GB, a pięć – 48 GB. Ponadto w każdym węźle jest dysk twardy o pojemności 3 TB. Możliwe jest zatem przetwarzanie 84 wątki jednocześnie. Komunikację między węzłami zapewnia sieć Gigabit Ethernet o przepustowości 1 Gb/s. Może być stosowana wersja 10 Gb/s lub 100 Gb/s. System wykorzystuje macierz dyskową

składającą się z 8 dysków o pojemności 2 TB każdy. Na prototyp napisany w języku Java przeznaczono siedem węzłów (nr 1–7).

System operacyjny Linux CentOS w wersji 6.1 jest zainstalowany w domenie o 50 GB pamięci na lokalnym dysku twardym. Ponadto w każdym węźle systemu Comcute JEE zainstalowano serwer NFS v3 (ang. *Network File System*) z protokołem do zdalnego udostępniania systemu plików. Wówczas po wysłaniu żądania przez klienta, żądanie zostaje odebrane przez serwer, a operacja jest wykonywana na dysku. Następnie potwierdzenie jest wysłane przez serwer, a klient odbiera to potwierdzenie. W rezultacie użytkownik może uzyskać dostęp do plików znajdujących się na zdalnym komputerze, a także z nich korzystać tak, jakby były na lokalnym dysku twardym w jego komputerze. Odnosi się to także do macierzy dyskowej.

Do węzła nr 1 przydzielono następujące zasoby dyskowe:

- 50 GB - system operacyjny;
- 3 TB - katalogi domowe udostępniane przez NFS pozostałym komputerom linuksowym;
- 568 GB - katalog /opt na potrzeby serwera GlassFish i stron WWW;
- 6,4 TB - katalog /macierz podpięty z macierzy dyskowej.

Natomiast pozostałe węzły linuksowe od nr 2 do nr 7 mają przydzielone następujące zasoby dyskowe:

- 50 GB - system operacyjny;
- 3 TB - katalogi domowe pobrane z serwera DNS;
- 3,6 GB - katalog /opt na potrzeby serwera GlassFish i stron WWW;
- 6,4 TB - katalog /macierz podpięty z macierzy dyskowej.

W węźle nr 1 zainstalowano także serwer stron WWW Apache 2.2.15. Istotną rolę odgrywa także system zarządzania bazą danych MySQL 5.1.52 w węźle nr 1. Serwer i biblioteki klienckie MySQL umożliwiają korzystanie z serwera bazodanowego MySQL z poziomu aplikacji implementowanych dla wielu platform i języków programowania, w tym także dla języka Java.

Reasumując, zainstalowane oprogramowanie podstawowe w węźle nr 1 to:

- baza danych MySQL 5.1.52;
- serwer DNS Bind 9.7.3;
- serwer NFSv3;
- serwer WWW Apache 2.2.15;
- serwer WWW JEE Glassfish 3.1.1;
- środowisko Java 1.7.0 update 2.

Zainstalowane oprogramowanie podstawowe w pozostałych węzłach (2-7) to:

- serwer WWW JEE GlassFish 3.1.1;
- środowisko Java 1.7.0 update 2.

Problem równoważenia obciążeń powinien zatem odbywać się w tym wielowarstwowym środowisku, a więc można postrzegać ten proces jako

równoległe, wielowarstwowe równoważenie obciążeń w siedmiu węzłach obliczeniowych. Wyróżnić można kilka warstw równoważenia obciążeń:

- DNS;
- system operacyjny CentOS;
- serwer aplikacji GlassFish;
- poziom aplikacji *S*, *W* i *Z*.

1.2. Równoważenie obciążenia na poziomie DNS

W węźle nr 1 zainstalowano DNS BIND 9.7.3 (ang. *Berkeley Internet Name Domain*, poprzednio: *Berkeley Internet Name Daemon*), który jest popularnym serwerem DNS wykorzystywanym w systemach Linux i Unix. DNS stanowi niezmiernie ważny składnik zapewniający poprawne działanie systemu nazw w Internecie. DNS zapewnia zamianę adresów znanych użytkownikom Internetu na adresy zrozumiałe dla urządzeń tworzących sieć komputerową. Nazwa mnemoniczna, np. onet.pl, może zostać zamieniona na odpowiadający jej adres IP, np.: *123.138.144.232* Usługa DNS warstwy aplikacji w modelu TCP/IP jest związana z portem nr 53 TCP/UDP.

Równoważenie obciążenia w DNS polega na przekazaniu żądań do różnych serwerów, mimo przydzielenia nazwy domeny do różnych adresów IP serwerów. Kiedy żądanie DNS jest skierowane do serwera DNS, to wówczas w celu ustalenia nazwy domeny, przydziela się jeden z adresów IP serwera w oparciu o strategie planowania, takie jak *round-robin* lub *planowanie geograficzne*. Następnie przekierowuje się żądanie do jednego z serwerów w grupie serwerów. Gdy domena żądania jest przydzielona do jednego z serwerów, kolejne żądania od klientów korzystających z tej samej lokalnej pamięci podręcznej serwera DNS są wysyłane do tego samego serwera.

DNS BIND 9.7.3 zapewnia równoważenie obciążenia i jest łatwy w użyciu. Zalety to:

- równoważenie obciążenia ruchu dla aplikacji opartych na protokole IP, w tym: HTTP, HTTPS (SSL), SSH, SMTP, IMAP, POP3, RDP (Terminal Services), DNS, LDAP, RADIUS i TFTP;
- przekierowywanie ruchu do właściwych serwerów usług na podstawie zawartości pakietów danych (wartości nagłówka, *cookies*, nazwy domeny lub adresu URL);
- mechanizm wykrywania serwerów i usług w celu dołączania nowych serwerów w systemie;
- przepustowość do 1 Gbps.

DNS BIND 9.7.3 jest wyposażony w mechanizm automatycznego wykrywania serwerów i narzędzia konfiguracyjne dostępne z poziomu intuicyjnego interfejsu WWW. Ponieważ serwer dystrybucyjny *S* może przydzielić kilka milionów zadań dziennie, to istotne jest zarządzenie tym serwerem pod względem obciążenia za pomocą DNS BIND 9.7.3. Na obciążenie serwera wpływa obciążenie związane z wykonywaniem procesów, które sterują przepływem kodu obliczeń, strumienia danych i wyników. Na to obciążenie

DNS BIND 9.7.3 ma mniejszy wpływ. Ponadto na obciążenie serwera *S* wpływa transmisja danych między zadanym *S* a *I*, na którą DNS BIND 9.7.3 ma stosunkowo duży wpływ podczas zgłaszania się internautów do serwera *S*.

Warto podkreślić, że skala obciążenia jest znacząca, gdyż jak szacuje się, podczas łamania szyfru DES, wymaga się przydzielenia ok. 64 tys. komputerów internautów, na których łamany jest już znacznie krótszy klucz o długości 40 bitów. Wówczas w ciągu kilku minut złamanie szyfru DES jest możliwe.

Istotne jest także uwzględnienie obciążeń wynikających z interakcji między zadanym *S* a nadrzędnym *W*.

Równoważenie obciążeń odgrywa za pomocą DNS BIND 9.7.3 zwiększy wydajność systemu rozproszonego Comcute. Może być postrzegane globalnie jako równoważenie obciążeń całego systemu Comcute ze względu na przyjęte kryterium jakości systemu lub też może być postrzegane lokalnie jako równoważenie obciążenia w warstwie *W* lub *S*.

W styczniu 2012 roku rozpoczęto weryfikowanie na PG zalety DNS BIND 9.7.3 pod kątem współpracy z laboratoryjnym zestawem Comcute JEE. W języku Java napisano następujące aplety: aplet do wyznaczanie liczb pierwszych, aplet do łamania szyfru DES oraz aplet do kompresji plików. Algorytm do wyznaczanie liczb pierwszych posiada także implementację w języku JavaScript.

Ponieważ na obniżenie wydajności wpływa niezbyt efektywny język implementacji zadań JavaScript, który będzie stanowił podstawowy formalizm opisu algorytmów obliczeniowych w docelowym Comcute, to zaproponowano, aby serwer dystrybucyjny ***S*** rozpoznawał, czy komputer internauty może prowadzić obliczenia w Javie, czy też konieczne jest stosowanie „wolniejszej" aplikacji napisanej w JavaScript. Wymaga to dysponowania dwiema implementacjami kodu obliczeń w Javie i JavaScript, które dostarczane są do internautów w zależności od możliwości realizacji przez komputer internauty. Po wykonaniu takich implementacji taka weryfikacji zalet DNS BIND 9.7.3 pod kątem efektywności w zakresie równoważenia obciążeń będzie możliwa.

1.3. Równoważenie na poziomie systemu operacyjnego CentOS

Równoważenie obciążenia obliczeniowego to podział zadań obliczeniowych w wielu różnych węzłów w klastrze, tak że cały klaster może zapewnić większą wydajność. Ten rodzaj systemów klastrowych jest również znany jako wysokiej wydajności klastry, który powinien być stosowany w ramach projektu Comcute w odniesieniu do klastrów składających się ze zbioru serwerów *S*, *W* i *Z* zlokalizowanych w jednym pomieszczeniu..

Implementacja równoważenia obciążeń w systemie Comcute polega na wykorzystaniu gotowych już mechanizmów stosowanych w systemach operacyjnych, serwerach aplikacji, protokołach komunikacyjnych oraz na przygotowania oryginalnego oprogramowania aplikacyjnego. Równoważenia obciążenia systemu Comcute powinno być implementowane w osobnym węźle

lub jako komponent używany na stronach internetowych. Wówczas program do równoważenia obciążenia akceptuje przychodzące żądanie HTTP i wybiera serwer ***S***, do którego powinno ono zostać przekierowane. Równoważenie obciążenia można zaimplementować za pomocą gotowych już komponentów systemu Linux lub w oparciu o protokół HTTP.

Na dystrybucji Linuksa CentOS zbudowano zestaw laboratoryjny Comcute i dlatego też rekomenduje się równoważenie obciążeń w oparciu o *Linux Virtual Server* (akronim *LVS*) w celu uzyskania wysokiej wydajności i wysokiej dostępności serwerów *Z, W* z *S* z wykorzystaniem technologii klastrów, co zapewnia dobrą skalowalność, niezawodność i łatwość obsługi. Architektura klastra serwerów jest w pełni przezroczysta dla użytkowników końcowych i interakcji użytkowników, jak gdyby był jednym wysokiej wydajności serwerem wirtualnym. W ramach projektu LVS dostępne są:

- zaawansowane oprogramowanie IP do równoważenia obciążenia (*IPVS*);
- równoważenie obciążenia na poziomie aplikacji (*KTCPVS*);
- elementy zarządzania klastrami.

IPVS (IP Virtual Server) wspomaga warstwę transportową (warstwa 4), równoważąc obciążenia w kernelu Linuksa. IPVS uruchomiony na komputerze równoważy obciążenia na klastrze serwerów, może on kierować żądania o usługi TCP/UDP do serwerów, a widziany jest jako wirtualny serwis pod jednym adresem IP .

W warstwie aplikacji (7) równoważenie obciążenia na poziomie aplikacji jest przetwarzaniem żądań i dystrybucji tych żądań o dopuszczenie do serwerów opartych na różnego rodzaju treści żądania tak, że może zapewnić jakości usług dla różnych typów treści i poprawy ogólnej wydajności klastra. Jego skalowalność jest ograniczona, w porównaniu do warstwy czwartej równoważenia obciążenia. KTCPVS jest implementacją równoważenia obciążenia dla jądra Linuksa właśnie w warstwie aplikacji. Z odpowiednich modułów *Apache*, *Lighttpd* i *nginx* serwer WWW może zapewnić równoważenia obciążenia jako *reverse proxy*.

Alternatywnym rozwiązaniem do LVS odnośnie równoważenia obciążeń jest *Ultra Monkey*, który koncentruje się na sieci lokalnej w zakresie równoważenia obciążenia i wysokiej dostępności usług, wykorzystujących komponenty w systemie Linux. *Ultra Monkey* jest również skalowalnym dla wysoko dostępnych serwerów w Internecie.

Z kolei *Red Hat Cluster Suite* implementuje dwa różne rodzaje klastrów: aplikacji lub usług, a także *IP Load Balancing*. Klaster zapewnia możliwość równoważenia obciążenia ruchu IP w obrębie farmy serwerów.

Dostępne w ramach projektu Linux-HA jest oprogramowanie *Heartbeat* na licencji GPL do zarządzania klastrami w celu uzyskania wysokiej dostępności.

Równoważenie obciążenia w *MPLS* to zrównoważenie usług sieciowych opartych na informacjach na etykiecie (ang. *Multiprotocol Label Switching*).

1.4. Równoważenie na poziomie serwera aplikacji GlassFish

Kluczową rolę w środowisku Comcute odgrywa serwer aplikacji GlassFish 3.1.1, który umożliwia przechowywanie i uruchamianie aplikacji *S, W, Z*, aplikacja do wyznaczania liczb pierwszych, aplikacja do łamania kodu DNS czy aplikacja do badania podobieństw skompresowanych plików. GlassFish składa się z komponentów udostępnionych programiście przy pomocy specjalnego API, których zadaniem jest wspieranie programisty przy pisaniu aplikacji poprzez zwolnienie go z odpowiedzialności za elementy nie związane bezpośrednio z logiką biznesową. Przykładem takich funkcjonalności mogą być:

- zarządzanie sesją,
- zarządzanie transakcjami danych,
- obsługa współdzielenia zasobów,
- obsługa skalowalności aplikacji i *load balancing*.

Architektura serwera aplikacji GlassFish zakłada obecność *cienkiego* klienta, który zajmuje się mało wymagającymi zadaniami, jak wyświetlanie GUI. Logika biznesowa przetwarzana jest właśnie na serwerze aplikacji. Aplikacja działająca na tymże serwerze przy pomocy udostępnionych metod komunikuje się z warstwą danych.

Wykorzystanie serwera aplikacji niesie ze sobą wiele zalet. Architektura cienkiego klienta nie tylko zdejmuje z klienta obciążenie związane z przetwarzaniem, ale również powoduje, iż łatwiej jest zarządzać logiką biznesową, bowiem jest ona scentralizowana na jednym serwerze (bądź małej ich liczbie). Od serwera WWW klasy Apache, serwer GlassFish odróżnia bardziej zaawansowana funkcjonalność, często np. dodatkowa kontrola dostępu do bazy danych.

Dla aplikacji webowych klasy *S, W i Z*, komponenty serwera GlassFish działają w tym samym węźle obliczeniowym, co serwer webowy, przeważnie wspomagając tworzenie dynamicznych stron. GlassFish udostępnia funkcjonalność znacznie wykraczającą poza generację stron webowych, jak na przykład klasteryzacja, rozwiązania *fail-over* oraz równoważenie obciążenia, pozwalając programiście skupienie się na rozwijaniu logiki biznesowej aplikacji.

Główne funkcje serwera aplikacji GlassFish:

- Spójność kodu i danych – poprzez centralizację logiki biznesowej wszystkie zmiany w aplikacji będą dotyczyły wszystkich użytkowników;
- Scentralizowana konfiguracja – wszystkie zmiany w konfiguracji aplikacji, jak na przykład zmiana serwera bazy danych lub ustawień systemowych odbywa się centralnie;
- Bezpieczeństwo – serwer aplikacji służy jako punkt wejścia do warstwy danych dla wszystkich użytkowników, rozdzielając potencjalnie

niebezpieczną warstwę kliencką od warstwy bazodanowej i implementując mechanizmy autoryzacji i uwierzytelniania;
- Wydajność – ograniczenie ruchu sieciowego;
- Oszczędność– powyższe punkty mogą skutkować w redukcji kosztów organizacji rozwijającej aplikacje biznesowe;
- Zarządzanie transakcjami – transakcja jest jednostką aktywności systemu, w której wiele zmian w zasobach aplikacji może zostać wykonanych atomowo.

GlassFish udostępnia obsługę wielu protokołów komunikacji sieciowej. Klienci webowi komunikują się za pomocą HTTP lub HTTPS, natomiast klienci EJB porozumiewają się przy pomocy ORB (ang. *Object Request Broker)* poprzez protokoły IIOP i IIOP/SSL. Ponadto możliwe jest uruchomienie aplikacji będącej usługą sieciową opartą na Java API for *XML-Based Remote Procedure* Call (JAX-RPC).

GlassFish umożliwia implementację usługi katalogowej obiektów za pomocą interfejsu JNDI (ang. *Java Naming and Directory Interface*) oraz zestawu kontraktów bezpieczeństwa zdefiniowanych dla kontenerów *Java Authorization Contract for Containers*.

Zarządzanie transakcjami i dostęp do systemu zarządzania bazą danych (JDBC), wysyłanie wiadomości pomiędzy komponentami programowymi (ang. *Java Messaging Service*), wysyłanie wiadomości e-mail (JavaMail) oraz administracja serwerem GlassFish to kolejne ważne funkcjonalności.

Model klient-serwer z serwerem aplikacji opiera się na architekturze wielowarstwowej, w którym warstwa pośrednicząca przyjmuje żądania cienkiego klienta, przetwarza je, komunikuje się z serwerem baz danych i zwraca klientowi wyniki w czytelnej dla niego formie. Rozwiązanie to zapewnia bezpieczeństwo, elastyczność, a także daje możliwość równoważenia obciążenia i replikacji.

Kontenery to środowiska uruchomieniowe dostarczające usług takich jak bezpieczeństwo i zarządzanie transakcjami dla komponentów aplikacji. Kontener sieciowy zawiera część aplikacji odpowiedzialną za prezentację. Kontener logiki (EJB w JavaEE) zawiera komponenty odpowiedzialne za logikę aplikacji.

GlassFish zapewnia klientom dostęp do aplikacji przez różne protokoły – HTTP i HTTPS dla przeglądarek, IIOP (*Internet Inter-ORB Protocol*) i IIOP/SSL dla klientów EJB czy SOAP. Możliwe jest wdrożenie aplikacji dostarczającej usługi sieciowej (ang. *Web Service*) implementowanej przez *Java API for XML-Based RPC* (JAX-RPC). Aplikacja lub komponent może być także klientem dla usługi sieciowej i komunikować się przez *Java API for XML Registries* (JAXR)

*K*ontenery dostarczają usług dla aplikacji:

- *Nazewnictwo* – usługa przypisuje obiekty do nazw. Aplikacja znajduje obiekt przez nazwę JNDI (*Java Naming and Directory Interface*).
- *Bezpieczeństwo* – zestaw zasad bezpieczeństwa zdefiniowanych dla kontenerów (*JACC – Java Authorization Contract for Containers*).

JDBC (*Java Database Connectivity*) zapewnia dostęp do baz danych, a JMS (*Java Messaging Service*) to metoda komunikacji pomiędzy komponentami i aplikacjami, *Connector* to komunikacja z systemami EIS (*Enterprise Information Systems*), a *Server Administration* – podsystem do zarządzania serwerem.

Istnieje również nowe rozwiązanie komercyjne, pozwalające na uruchomienie zarówno aplikacji w JEE jak i .NET: *Interstage Application Server powered by Windows Azure* – serwer aplikacji uruchomiony w 2011 roku przez firmę Fujitsu.

Należy podkreślić, że serwery aplikacji GlassFish ze swoimi usługami wybiegają dużo dalej niż funkcjonalności serwerów WWW (dzięki czemu programiści mogą skupić się przede wszystkim na implementacji logiki biznesowej) oferując takie usługi jak klasteryzacja, czy równoważenie obciążenia.

GlassFish służy jako punkt wejścia do warstwy danych dla aplikacji, rozdzielając potencjalnie niebezpieczną warstwę kliencką od warstwy bazodanowej i implementując mechanizmy autoryzacji i uwierzytelniania.

Zarządzanie wydajnością w serwerze GlassFish sprowadza się najczęściej do ograniczenia ruchu sieciowego. Zarządzanie transakcjami też jest możliwe, przy czym transakcja jest jednostką aktywności systemu, w której wiele zmian w zasobach aplikacji może zostać wykonanych atomowo.

Duża skalowalność (obsługa klastrów) i zapewnienie zintegrowanego, zcentralizowanego zarządzania to kolejna zaleta serwera GlassFish. Obsługa standardów polega na tym, że klienty webowe komunikują się za pomocą HTTP lub HTTPS, natomiast klienty EJB porozumiewają się przy pomocy ORB (ang. *Object Request Broker*) poprzez protokoły IIOP i IIOP/SSL. Istotne jest także zapewnienie środowiska do tworzenia, testowania, wdrażania i zarządzania aplikacjami, a także dostarczenie biblioteki klas (API).

Architektura serwera aplikacji zakłada obecność cienkiego klienta, który zajmuje się mało wymagającymi zadaniami, jak wyświetlanie GUI. Logika biznesowa przetwarzana jest właśnie na serwerze aplikacji. Aplikacja działająca na tymże serwerze przy pomocy udostępnionych interfejsów. Funkcjonalność serwerów na różne platformy definiują odpowiednie specyfikacje (np. w przypadku Java EE 6 specyfikacje JSR-316).

Z konieczności współpracy z wieloma typami rozwiązań wynika jedna z najistotniejszych cech serwera GlassFish – otwartość na standardy wymiany informacji. Tylko dzięki temu możliwe jest integrowanie różnych rozwiązań informatycznych.

1.5. Równoważenie obciążeń na poziomie aplikacji

Na poziomie aplikacji możliwe jest równoważenia obciążeń w systemie Comcute za pomocą opracowanej metody w [1]. Jeżeli przydział zadań obliczeniowych i pakietów danych do serwerów w systemie Comcute minimalizuje łączne obciążenie, to może wystąpić sytuacja skupienia modułów

(rozumianych jako zadania obliczeniowe lub pakiety danych) w wybranym węźle obliczeniowym. Ten węzeł będzie zatem najbardziej obciążony pod względem liczby przydzielonych zadań, a pozostałe węzły mogą być obciążone w znacznie mniejszym stopniu.

Przyjęto, że miarą obciążenia serwera w systemie Comcute jest Z_{max} – łączny czas przetwarzania danych i obciążenia zewnętrzną komunikacją w najbardziej wykorzystywanym serwerze. W wyniku przydzielenia modułów do *i*-tego węzła serwer w nim usytuowany jest zajęty realizacją tych modułów w czasie $\hat{Z}_i(x)$, który nazywamy czasem zajętości serwera *Z* lub *W* przydzielonego do *i*-tego węzła. Wartość czasu $\hat{Z}_i(x)$ wyznacza się w następujący sposób:

$$\hat{Z}(x) = \sum_{j=1}^{J}\sum_{v=1}^{V} t_{vj} x_{vi}^{m} x_{ij}^{\pi}. \tag{1}$$

gdzie:

I – liczba węzłów w systemie Comcute , aktualnie *I*=7;

$x^m = (x_{11}^m,...,x_{vi}^m,...,x_{VI}^m)^T$ – binarny wektor przydziału dystrybuowanych zadań i pakietów danych do serwerów, *v* – nr modułu (zadania lub pakietu danych);

$x^\pi = (x_{11}^\pi,...,x_{ij}^\pi,...,x_{IJ}^\pi)^T$ – binarny wektor odwzorowujący hosty na węzły obliczeniowe, *i* – nr węzła, *j* – nr hosta;

t_{vj} – szacowany czas wykonania modułu m_v na hoście π_j, *J* – liczba rodzajów serwerów (przyjmujemy *J*=2, serwer typu *S* ma numer 1, a *W* – 2)

Istotnym obciążeniem podsystemu komunikacyjnego *i*-tego węzła jest wysyłanie i odbiór danych. Czas $\widetilde{Z}_i(x)$ nadawania i odbioru danych w *i*-tym serwerze wyznacza się, jak niżej:

$$\widetilde{Z}_i(x) = \sum_{v=1}^{V}\sum_{\substack{u=1 \\ u\neq v}}^{V}\sum_{\substack{i_2=1 \\ i_2\neq i}}^{I} \tau_{vu} x_{vi}^{m} x_{ui_2}^{m}. \tag{2}$$

Obciążeniem *i*-tego węzła nazywamy łączny czas zajętości serwera przydzielonego do *i*-tego węzła oraz czas nadawania i odbioru danych w *i*-tym węźle. Obciążenie *i*-tego węzła $Z_i^+(x)$ dla zadanego przydziału x wyznaczamy w następujący sposób:

$$Z_i^+(x) = \sum_{j=1}^{J}\sum_{v=1}^{V} t_{vj} x_{vi}^{m} x_{ij}^{\pi} + \sum_{v=1}^{V}\sum_{\substack{u=1 \\ u\neq v}}^{V}\sum_{\substack{i_2=1 \\ i_2\neq i}}^{I} \tau_{vu} x_{vi}^{m} x_{ui_2}^{m}. \tag{3}$$

Serwer o największym obciążeniu $Z_i^+(x)$ stanowi „wąskie gardło" systemu Comcute. Im mniejsze jest jego obciążenie, tym bardziej moduły są zdecentralizowane. Ponadto mniej obciążony serwer generuje mniejszy ruch danych w sieci komputerowej. Dlatego minimalizacja największego obciążenia w węzłach systemu Comcute prowadzi do równoważenia obciążeń.

Maksymalne obciążenie węzła dla zadanego przydziału modułów do serwerów x wyraża się następującą formułą:

$$Z_{\max}(x) = \max_{i \in \overline{1,I}} \left\{ Z_i^+(x) \right\} \tag{4}$$

Jeżeli moduły zostaną przesunięte z hosta w najbardziej zajętym węźle na komputery obciążone w znacznie mniejszym stopniu, to zostanie zmniejszona zajętość najbardziej obciążonego serwera, co skutkuje dążeniem do wyrównania obciążenia wszystkich węzłów w systemie. Postulat równoważenia obciążenia w systemie Comcute może być uwzględniony przez funkcję celu $Z_{\max}$ lub też poprzez wprowadzenie dodatkowego ograniczenia, aby $Z_{\max}(x) \le T_{\text{gr}}$, gdzie T_{gr} reprezentuje założone największe dopuszczalne obciążenie serwera w systemie Comcute. Sformułowano zadanie minimalizacji funkcji $Z_{\max}$ umożliwiające równoważenie obciążeń w systemie Comcute:

Dla danych $\mathbf{T} = [t_{vj}]_{V \times J}$, $\tau = [\tau_{vu}]_{V \times V}$ należy wyznaczyć x^*, takie że

$$Z_{\max}(x^*) = \min_{x \in \mathcal{X}} Z_{\max}(x), \tag{5}$$

gdzie:

$$\mathcal{X} = \{ x \in \mathcal{B}^{I(V+J)} |$$
$$x = [x_{11}^m, \ldots, x_{1i}^m, \ldots, x_{1I}^m, \ldots, x_{vi}^m, \ldots, x_{VI}^m, x_{11}^\pi, \ldots, x_{1j}^\pi, \ldots, x_{1J}^\pi, \ldots, x_{ij}^\pi, \ldots, x_{IJ}^\pi]^T,$$
$$\sum_{i=1}^{I} x_{vi}^m = 1 \text{ dla } v = \overline{1,V}, \qquad \sum_{j=1}^{J} x_{ij}^\pi = 1 \text{ dla } i = \overline{1,I} \}.$$

Zadanie optymalizacji (5) charakteryzuje się skończonym zbiorem rozwiązań dopuszczalnych oraz nieliniową funkcją celu. Dla $V \ge 1$, $J=1$ i $I=7$ istnieje co najmniej jedno rozwiązanie dopuszczalne. Liczba przydziałów modułów do komputerów wynosi 7^V, rośnie wykładniczo wraz ze wzrostem liczby modułów oraz liniowo wraz ze wzrostem liczby węzłów.

Zbiór możliwych wariantów w problemie (5) dla reprezentacji binarnej przydziałów modułów do komputerów zawiera $2^{7(V+1)}$ alternatyw. Przyjmując reprezentację całkowitoliczbową przydziału modułów do hostów, uzyskuje się rozwiązanie dopuszczalne, a zbiór rozwiązań dopuszczalnych zawiera 7^V przydziałów.

Do rozwiązania zadania (5) zastosowano algorytm przeszukiwania tabu TSZmax [1]. Optymalne przydziały uzyskano dla instancji o 32 binarnych zmiennych decyzyjnych ($V=11$, $I=2$, $J=5$) dla 100 różnych punktów startowych.

Wraz ze wzrostem rozmiaru zadania średnia liczba wyznaczanych rozwiązań optymalnych L^* zaczęła maleć. Dla instancji o 36 binarnych zmiennych decyzyjnych ($V=10$, $I=3$, $J=2$) względny błąd maksymalny $B_{\max}$ wyniósł 9,4%, średni błąd względny $\overline{B}$ – 2,5%, a w 41% wypadków wyznaczono przydziały optymalne. Ta instancja cechuje się ponad 68 miliardami binarnych przydziałów

modułów do komputerów, a zbiór rozwiązań dopuszczalnych zawiera 472 392 przydziały dopuszczalne. Czas obliczeń programu implementującego algorytm TSZmax wyniósł 0,4 sekundy. Program uruchamiano w środowisku Matlab na komputerze klasy PC. Algorytm TSZmax w ciągu minuty umożliwia wyznaczenie rozwiązań o wyższej jakości niż algorytm przeglądu przydziałów dopuszczalnych w ciągu godziny.

Dla instancji o 60 binarnych zmiennych decyzyjnych (V=12, I=4, J=3) w wyniku 100 eksperymentów numerycznych wyznaczono wynik referencyjny, w odniesieniu do którego względny błąd maksymalny $B_{\max}$ wyniósł 7,2%, średni błąd względny $\overline{B}$ był równy 5,2%, a w 6% wypadków wyznaczono przydziały referencyjne. Ta instancja cechuje się ponad 10^{18} binarnymi przydziałami modułów do komputerów, a zbiór rozwiązań dopuszczalnych zawiera 1 358 954 496 przydziałów dopuszczalnych. Czas obliczeń programu implementującego algorytm TSZmax wyniósł 20 sekund.

Dla instancji o 100 binarnych zmiennych decyzyjnych (V=20, I=4, J=5) w wyniku 100 eksperymentów numerycznych wyznaczono wynik referencyjny, w odniesieniu do którego względny błąd maksymalny $B_{\max}$ wyniósł 4,7%, średni błąd względny $\overline{B}$ był równy 2,1%, a w 5% wypadków wyznaczono przydziały referencyjne. Ta instancja cechuje się ponad 10^{30} binarnymi przydziałami modułów do komputerów, a zbiór rozwiązań dopuszczalnych zawiera prawie 7×10^{14} przydziałów dopuszczalnych. Czas obliczeń programu implementującego algorytm TSZmax wyniósł 115 sekund.

Powyższe eksperymentalne wyniki weryfikują pozytywnie hipotezę, że można równoważyć obciążenia w systemie siedmiowęzłowego Comcute JEE za pomocą algorytmu obliczeniowego TSZmax, ale czas potrzebny na wyznaczenie takiego obciążenia jest na poziomie kilku minut

1.6. Symulowanie równoważenia obciążeń za pomocą oprogramowania

Interesujące podejście do oszacowania równoważenia obciążeń opiera się na wykorzystaniu specjalistycznego oprogramowania symulującego obciążenie systemu. Oprogramowanie to służy do badania obciążenia (w tym równoważenia obciążenia), testowania, a także badania projektów systemów rozproszonych w zależności od założonych warunków początkowych lub parametrów technicznych. Daje możliwość zbadania systemu Comcute pod kątem jego modyfikacji.

Oprogramowanie OPNET umożliwia jest zarządzanie wydajnością aplikacji, a także planowanie i projektowanie zestawu laboratoryjnego Comcute. Zarządzanie obejmujące monitorowanie pracy, analizę działania aplikacji, zaawansowane rozwiązywanie problemów, planowanie wydajności oraz analizy systemu rozproszonego. Ponadto dostępne są opcje skutecznego zarządzania wydajnością aplikacji w warunkach zestawu laboratoryjnego Comcute. Wydajne, intuicyjne środowisko analityczne umożliwia certyfikację aplikacji przed ich wdrożeniem, sprawdzanie skutków zaplanowanych zmian oraz

przyspieszenie procesu rozwiązywania problemów związanych z wydajnością aplikacji produkcyjnych.

Istotną funkcjonalnością jest zbieranie informacji przez agentów programistycznych. Warto także zasygnalizować możliwość analizowania informacji na temat pakietów sieciowych poparte automatycznym prowadzeniem statystyk. Intuicyjne, zaawansowane schematy graficzne umożliwiają wizualizację działania aplikacji. Z punktu widzenia badania obciążenia istotne są raporty diagnostyczne umożliwiające określenie wąskich gardeł wydajności i źródeł opóźnień czasu odpowiedzi w otoczeniu internetowym. Możliwe jest szybkie i dokładne przewidywanie czasów odpowiedzi aplikacji w środowiskach wirtualnych obejmujących wiele klientów, serwerów i aplikacji, co jest szczególnie dogodne dla systemu Comcute.

Planowanie wdrażania optymalizacji systemu Comcute umożliwia sprawdzania wpływu proponowanych modyfikacji na czas reakcji w zoptymalizowanych środowiskach. Rozwiązanie pozwala organizacjom dużo skuteczniej planować wdrażanie urządzeń optymalizujących sieci poprzez identyfikację lokalizacji i aplikacji, które mogłyby przynosić możliwie największe korzyści.

Symulacje pozwalają przewidzieć zachowanie infrastruktury w różnych sytuacjach. Środowisko uwzględnia skomplikowane relacje pomiędzy poszczególnymi elementami otoczenia internetowego (warstwy *I* w Comcute) i dostarcza szerokie spektrum informacji niezbędnych do oceny wydajności aplikacji i pracy Comcute. Dzięki tym informacjom możliwe jest planowanie efektywnych systemów informatycznych, ich konfiguracja, rozbudowa i diagnostyka.

Podstawową korzyścią ze stosowania tego oprogramowania jest uniknięcie eksperymentów na „żywym organizmie”, jakim jest demonstrator Comcute.

Możliwe jest także planowanie pojemności serwerów przy zastosowaniu analizy konfiguracji i obciążeń roboczych systemów, a także prognozowania skutków wdrożeń nowych aplikacji, zmian w profilach obciążeń roboczych i efektów dostrajania wydajności. Wykorzystując integrację z narzędziami do monitorowania wydajności serwerów, można zaplanować wydajność serwerów, aby sprostać rozrostowi systemu, konsolidacji i wirtualizacji serwerów, a także wdrożeniom kolejnych aplikacji.

Można symulować wbudowane specyficzne dla konkretnych producentów systemy z wartościami opisującymi ich wydajność dla testu SPEC (ang. *Standard Performance Evaluation Corporation*), a także skonfigurowania własnych serwerów. Ważna cecha to integracja ze statystykami o wydajności serwerów z wiodących na rynku rozwiązaniami do monitorowania: *CA Unicenter*, *Network and Systems Management*, *HP OpenView Performance Agent* i *HP OpenView Performance Manager*, *BMC Performance Assurance Perform Collector* i *Microsoft Windows Vista/7 Perfmon*.

Symulacja systemu Comcute umożliwia opracowanie charakterystyki obciążenia roboczego aplikacji przy użyciu zaawansowanych algorytmów filtrujących. Ocena wydajności architektury aplikacji to kolejna interesująca

opcja z punktu widzenia symulacji środowiska Comcute, co pozwala przewidzieć skutki odnośnie wydajności, jakie mogą nastąpić z powodu zmian architektury aplikacji w czasie jej projektowania. Symulacja bada, czy możemy skonsolidować aplikacje z różnych systemów na jednej platformie oraz jak mamy skonfigurować serwer, żeby wspierał systemy wirtualne.

Można oszacować:

- pojemność i na ile ona wystarczy,
- ile pojemności powinniśmy dodać do naszych systemów,
- w jaki sposób wzrost obciążenia roboczego będzie wpływał na poziomy dostępności aplikacji,
- czy mamy wystarczającą pojemność dla każdej warstwy architektury aplikacji?

1.7. Równoważenie obciążeń w węźle nr 9

Węzeł nr 9 to serwer z systemem operacyjnym Windows 2008 Server R2. Zasoby dyskowe obejmują”

- dysk twardy;
- 199 GB - system operacyjny;
- 3,4 TB - dysk twardy na pozostałe rzeczy, prototyp Comcute itp.

Zainstalowane oprogramowanie:

- serwer WWW IIS 7.0
- wtyczka pGina 2.1.1 - do autoryzacji użytkowników z LDAP
- XenCenter - do zarządzania maszynami wirtualnymi na serwerze poligon,
- szczegóły podam później, jak to przetestujemy

Dla aplikacji napisanych na platformie .NET za serwer aplikacji może służyć odpowiednio skonfigurowany serwer IIS7. Składa się on z modułów podzielonych na kategorie. Moduły te spełniają podstawową i rozszerzoną funkcjonalność, jaką powinien zapewniać serwer aplikacji. Kategorie podziału to modułów to:

- moduły HTTP,
- moduły bezpieczeństwa,
- moduły treści,
- moduły kompresji,
- moduły *cache*,
- moduły rejestrowania danych i diagnozowania,
- moduły zarządzania rozszerzeniami,
- moduły rozszerzające.

Warto także wspomnieć o istnieniu serwerów aplikacji przeznaczonych na inne platformy niż JEE i .NET, jak na przykład Zend Server dla aplikacji napisanych w PHP.

1.8. Podsumowanie

Kluczowe podejście do implementacji równoważenia obciążeń w systemie Comcute polega na doborze i implementacji mechanizmów równoważenia obciążeń stosowanych w protokołach komunikacyjnych lub oprogramowaniu aplikacyjnym. Równoważenia obciążenia systemu Comcute zaimplementowano na hoście oraz jako komponent używany na stronach internetowych.

Wielopoziomowe równoważenie obciążeń dotyczy następujących poziomów:

- DNS BING;
- systemu Linux CentOS;
- serwer aplikacji GlassFish
- aplikacji za pomocą algorytmu TZmax;

Wskazane jest przed modyfikacją systemu Comcute stosować symulację komputerową.

Na poziomie systemu operacyjnego *Linux Virtual Server* zapewnia dobrą skalowalność, niezawodność i łatwość obsługi w wyniku zastosowania następujących aplikacji:

- zaawansowane oprogramowanie IP do równoważenia obciążenia (IPVS);
- oprogramowania równoważenie obciążenia (KTCPVS);

Ponadto za pomocą modułów *Apache* o nazwie *Lighttpd* i *nginx* serwer WWW może zapewnić równoważenie obciążenia jako *reverse proxy*.

Równoważenie obciążeń w systemie Comcute może być postrzegane globalnie jako równoważenie obciążeń całego systemu ze względu na przyjęte kryterium jakości systemu (za pomocą algorytmu TZmax) lub też może być postrzegane lokalnie jako równoważenie obciążenia w obszarze aplikacji *W*, *S* lub *I*.

1.9. Wykaz literatury

1. J. Balicki: *Algorytmy ewolucyjne i tabu* search *do optymalizacji przydziałów modułów programistycznych w rozproszonych systemach komputerowych*. Wyd. AMW, Gdynia 2000
2. Kaskbook: *Serwery aplikacji*, Gdańsk–Frombork 2002;
3. Kaskbook: *Aplikacje rozproszone i systemy internetowe*, Gdańsk–Stawiska 2006;
4. http://download.oracle.com/docs/cd/E19159-01/819-3671/ablat/index.html, marzec 2012
5. http://en.wikipedia.org/wiki/Application_server, marzec 2012
6. http://learn.iis.net/page.aspx/101/introduction-to-iis-architecture/, 2012
7. http://technet.microsoft.com/pl-pl/library/cc772115%28WS.10%29.aspx, 2012
8. http://www.bestpricecomputers.co.uk/glossary/application-server.htm, 2012
9. http://www.cc.com.pl/docs/serw_aplikacji-98.pdf, marzec 2012;
10. http://www.fujitsu.com/global/services/software/windows-azure/, marzec 2012