STUDIA INFORMATICA 2001
Volume 22 Number 3 (45)

Ziemowit NOWAK, Radosław PORCZYŃSKI
Politechnika Wrocławska, Instytut Sterowania i Techniki Systemów

MODELE FUNKCJONALNE SYSTEMÓW DO RÓWNOWAŻENIA OBCIĄŻEŃ W SERWISACH WWW

Streszczenie. W artykule przedstawiono taksonomię systemów do równoważenia obciążeń w serwisach WWW. W pierwszej części przeprowadzono analizę prostych oraz złożonych modeli funkcjonalnych skalowalnych systemów serwerów webowych. W drugiej części przeprowadzono analizę algorytmów i strategii stosowanych do równoważenia obciążeń w systemach serwerów webowych. Artykuł jest podsumowaniem prac związanych z poszukiwaniem najlepszego modelu do badań nad wydajnością nowych algorytmów równoważenia obciążeń w serwisach WWW.

FUNCTIONAL MODELS OF LOAD BALANCING SYSTEMS FOR DISTRIBUTED WEB SERVICES

Summary. In this article we presented taxonomy of load balancing systems for Web servers. In first part of the article we presented analyses of functional models of simply and complex scalable web server systems. Second part contains analysis of algorithms and strategies used in load balancing systems for distributed Web servers. This article is conclusion of looking for best model of distributed Web server. This model will be used in research on new algorithm in load balancing systems for distributed Web servers.

1. Wstęp

Serwisy WWW mogą udostępniać swoje zasoby opierając się na platformach jednoserwerowych lub wieloserwerowych. Ponieważ zwiększanie ilości procesorów i pamięci w platformach jednoserwerowych ma swoje granice, przyszłość należy do platform wieloserwerowych. Ich główną zaletą jest możliwość stopniowej i teoretycznie nieograniczonej rozbudowy wraz ze wzrostem potrzeb użytkowników, czyli tzw.

skalowalność. Wieloserwerowe platformy wykorzystywane na potrzeby serwisów WWW określane są mianem skalowalnych systemów serwerów webowych (scalable web-server systems).

Kluczową rolę w skalowalnych systemach serwerów webowych odgrywają:

- **mechanizm szeregowania** kierujący żądania klientów do najlepszego serwera webowego,
- **algorytm szeregowania** wyznaczający najlepszy serwer,
- **egzekutor**, czyli urządzenie, w którym zaimplementowany jest mechanizm i algorytm szeregowania.

2. Taksonomia skalowalnych systemów serwerów webowych

Na rysunku 1 przedstawiono wszystkie proste (prymitywne) modele funkcjonalne systemów do równoważenia obciążeń w serwisach WWW. Ich charakterystyczną cechą jest to, iż w każdym wykorzystywany jest jeden mechanizm szeregowania. Można je traktować jak swego rodzaju klocki, z których budowane są rzeczywiste systemy. Cześć z nich odpowiada rzeczywistym systemom bez dodatkowych modyfikacji, część odzwierciedla rzeczywiste systemy dopiero we wzajemnych kombinacjach, wreszcie model c) ze względu na niebezpieczeństwo częstych przeciążeń nie jest wykorzystywany w praktyce. Wyodrębnienie prostych modeli umożliwia zbudowanie przejrzystej taksonomii skalowalnych systemów serwerów webowych.

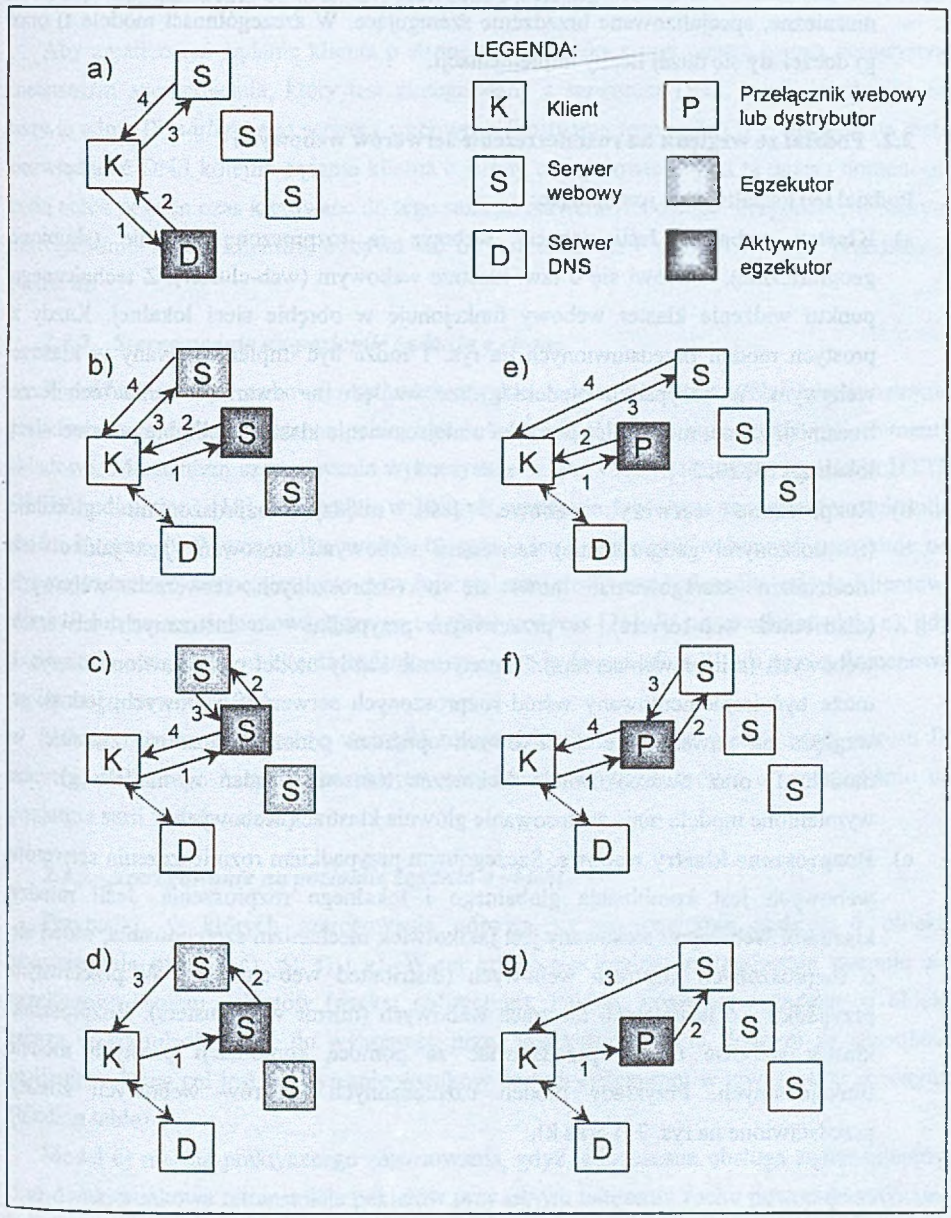
2.1. Podział ze względu na umiejscowienie mechanizmu szeregowania

Mechanizm szeregowania może być umiejscowiony: po stronie klienta, po stronie serwera DNS (Domain Name System), w serwerze webowym lub w dedykowanym urządzeniu. Pierwszy przypadek, ze względu na ograniczone zastosowanie, w niniejszych rozważaniach nie będzie brany pod uwagę.

a) **Mechanizm szeregowania zintegrowany z serwerem DNS.** Przypadkowi temu odpowiada model a) [7, 12, 14, 22]. Egzekutorem jest serwer DNS lub inne urządzenie wspomagające albo przejmujące rolę serwera DNS. Należy uściślić, że chodzi tu o tzw. główny serwer DNS będący autorytatywnym źródłem informacji o określonej domenie (authoritative DNS).

b) **Mechanizm szeregowania umiejscowiony w serwerach webowych.** Przypadkowi temu odpowiadają modele b), c) oraz d), Jak już wspomniano, model c) nie ma

praktycznego zastosowania. Modele te nie występują samodzielnie. Przykłady złożonych modeli z udziałem b) i d) przedstawiano na rys. 2.



Rys.1. Proste (prymitywne) modele funkcjonalne skalowalnych systemów serwerów webowych

Fig.1. Basic functional models of scalable Web server systems

c) **Mechanizm szeregowania umiejscowiony w dedykowanym urządzeniu.** Przypadkowi temu odpowiadają modele e), f) oraz g). Rolę egzekutora pełni niezależne, specjalizowane urządzenie szeregujące. W szczególności modele f) oraz g) doczekały się dużej liczby implementacji.

2.2. Podział ze względu na rozmieszczenie serwerów webowych

Podział ten kształtuje się następująco:

a) **Klaster webowy.** Jeśli serwery webowe są rozproszone lokalnie (skupione geograficznie), to mówi się o tzw. klastrze webowym (web-cluster). Z technicznego punktu widzenia klaster webowy funkcjonuje w obrębie sieci lokalnej. Każdy z prostych modeli przedstawionych na rys. 1 może być implementowany w klastrze webowym. W przypadku modelu g), ze względu na uwarunkowania techniczne transmisji żądań, może zajść potrzeba umiejscowienia klastra w obrębie podsieci sieci lokalnej [15, 20, 21].

b) **Rozproszone serwery webowe.** Jeśli między rozproszonymi globalnie (rozproszonymi geograficznie) serwerami webowymi stosowany jest jakikolwiek mechanizm szeregowania, mówi się o rozproszonych serwerach webowych (distributed web-servers), w przeciwnym przypadku - o lustrzanych serwerach webowych (mirror web-servers). Teoretycznie każdy model przedstawiony na rys. 1 może być implementowany wśród rozproszonych serwerów webowych, jednak ze względu na powstawanie dodatkowych opóźnień podczas transmisji zwrotnej w modelu f) oraz uwarunkowania techniczne transmisji żądań w modelu g), oba wymienione modele mają zastosowanie głównie klastrach webowych.

c) **Rozproszone klastry webowe.** Szczególnym przypadkiem rozmieszczenia serwerów webowych jest kombinacja globalnego i lokalnego rozproszenia. Jeśli miedzy klastrami webowymi stosowany jest jakikolwiek mechanizm szeregowania, mówi się o rozproszonych klastrach webowych (distributed web-clusters). W przeciwnym przypadku - o lustrzanych klastrach webowych (mirror web-clusters). Rozproszone klastry webowe można przedstawiać za pomocą kombinacji prostych modeli funkcjonalnych. Przykłady modeli rozproszonych klastrów webowych zostały przedstawione na rys. 2 j) oraz k).

2.3. Podział ze względu na poziom szczegółowości szeregowania

2.3.1 Szeregowanie na poziomie serii żądań o strony

Aby zrealizować żądanie klienta o stronę umiejscowioną pod pewną nazwą domenową, mechanizm szeregowania, który jest zintegrowany z serwerem DNS, musi przydzielić tej nazwie adres IP *najlepszego* serwera webowego (hostname resolution). Ze względu na dużą bezwładność DNS kolejne żądania klienta o strony umiejscowione pod tą nazwą domenową będą przez pewien czas kierowane do tego samego serwera webowego. Przypadek, w którym szeregowanie żądań klientów odbywa się na poziomie serii żądań o strony, przedstawia model a).

2.3.2. Szeregowanie na poziomie żądania o stronę

W skład strony webowej wchodzi zwykle wiele obiektów. Na tym poziomie przekierowanie żądania o stronę pociąga za sobą przekierowanie serii żądań o jej elementy składowe. Mechanizm szeregowania wykorzystuje w tym celu właściwości protokołu HTTP (HTTP redirection) [18]. Przypadki, w których występuje tego typu szeregowanie, ukazuje model b) oraz e). W przypadku modelu b), gdy jeden z serwerów webowych otrzymuje od klienta żądanie o stronę, jeśli nie decyduje się sam zrealizować zlecenia, odsyła klientowi adres IP lub nazwę domenową *lepszego* od siebie serwera [23]. W przypadku modelu e), gdy przełącznik otrzymuje od klienta żądanie o stronę, odsyła mu adres IP lub nazwę domenową *najlepszego* serwera [19].

Należy zaznaczyć, że może wystąpić tu zjawisko buforowania przez klienta adresu IP serwera, do którego nastąpiło przekierowanie. Można wówczas mówić o szeregowaniu na poziomie serii żądań o strony.

2.3.3. Szeregowanie na poziomie żądania o obiekt

Przypadki, w których szeregowanie odbywa się na poziomie żądania o obiekt, przedstawiają modele c), d), f) i g). W tej grupie mechanizm szeregowania zajmuje się przekierowywaniem pakietów (packet redirection). Pakiety zawierające żądanie o obiekt muszą w komplecie trafić do wybranego przez egzekutor serwera. Jednym ze sposobów realizujących ten cel jest zapisywanie wyników decyzji egzekutora w tzw. tablicy powiązań (binding table).

Model c) nie ma praktycznego zastosowania, gdyż jednoczesna obsługa żądań klientów oraz dwukierunkowa retransmisja pakietów przy silnym natężeniu ruchu powoduje zatykanie się serwerów. Ponieważ w modelu d) ruch pakietów powracających (znacznie większy niż w przypadku pakietów z żądaniami) jest kierowany do klienta z pominięciem serwera-egzekutora, model ten jest z powodzeniem implementowany. Charakterystyczną cechą modeli

f) i g) jest to, że egzekutor maskuje serwery webowe za pomocą jednego, wirtualnego adresu IP-SVA (single virtual IP address). Aby przekazać żądanie do konkretnego serwera, stosuje różne, niekiedy wyrafinowane techniki. Gdy egzekutor przekierowuje pakiety z żądaniami bez świadomości ich treści (content information blind), nazywany jest dystrybutorem (dispatcher, level 4 web-switch). Gdy egzekutor podczas podejmowania decyzji o przekierowaniu wykorzystuje informację zawartą w żądaniu (content information aware), nazywany jest przełącznikiem webowym (content switch, level 7 web-switch).

Należy podkreślić, że niniejszy podział dotyczy wariantu, gdzie w modelach f) i g) rolę egzekutora pełni dystrybutor. W przypadku gdy egzekutorem jest przełącznik webowy, szeregowanie może odbywać się na poziomie żądania o obiekt, stronę oraz (dzięki umiejętności rozpoznawania znaczników *cookie*) serii żądań o strony.

2.4. Podział ze względu na poziom kontroli zapytań

Jest to podział wyodrębniający modele, w których egzekutory mają pełna kontrolę nad zapytaniami kierowanymi do systemu serwerów webowych.

a) **Pełna kontrola zapytań.** Teoretycznie do tej grupy należą modele b), c), d), e), f) oraz g). Klient, po otrzymaniu od serwera DNS adresu IP egzekutora, wszystkie zapytania kieruje tylko i wyłącznie do niego. W ten sposób egzekutor ma stuprocentową kontrolę nad zapytaniami. W rzeczywistości sytuacja taka zachodzi w przypadku modeli f) oraz g). Warto zwrócić uwagę, że awaria lub zapchanie się egzekutora powoduje unieruchomienie całego systemu serwerów webowych.

b) **Częściowa kontrola zapytań.** Typowym przypadkiem, w którym egzekutor sprawuje częściową kontrolę nad zapytaniami, jest model a). Częściowa kontrola wynika ze specyfiki funkcjonowania systemu DNS. Również modele b) oraz e) należy zaliczyć do tej grupy ze względu na możliwość występowania zjawiska buforowania przez klienta adresu IP serwera, do którego nastąpiło przekierowanie. Jeśli modele c) oraz d) byłyby implementowane samodzielnie, egzekutor, którego adres IP rozgłaszałby serwer DNS, sprawowałby pełną kontrole nad zapytaniami. Jednak w przypadku modelu d) typowym rozwiązaniem jest jego złożenie z modelem a) - patrz rys. 2 i). W takim przypadku każdy z egzekutorów implementowanych w serwerach webowych sprawuje częściową kontrolę nad zapytaniami kierowanymi do systemu serwerów webowych.

2.5. Podział ze względu na strategię rozmieszczenia mechanizmów szeregowania

Z tej perspektywy rysuje się podział na dwie grupy modeli: z centralnym mechanizmem szeregowania i z rozproszonymi mechanizmami szeregowania.

a) **Centralny mechanizm szeregowania.** Do tej grupy należą modele a), e), f) oraz g). W przypadku a) decyzja o przydzieleniu nazwie domenowej adresu IP najlepszego serwera webowego zapada centralnie, po stronie serwera DNS. Mimo scentralizowanego zarządzania, ze względu na specyfikę funkcjonowania DNS, skuteczność tego modelu jest niewielka [14]. W modelach e), f) oraz g) decyzja o przekierowaniu żądania klienta zapada centralnie, w dedykowanym urządzeniu. W odróżnieniu od modelu a), centralne zarządzanie oznacza stuprocentową kontrolę nad szeregowaniem.

b) **Rozproszone mechanizmy szeregowania.** W tej grupie decyzja o przekierowaniu żądania może zapaść na każdym serwerze, w którym zaimplementowano oprogramowanie egzekutora. Ideę rozproszonego szeregowania ukazują modele b), d) oraz, dla porządku, c). Rozwiązanie takie zapobiega efektowi wąskiego gardła, na które w szczególności narażony jest model f). Modele b) i d) stosowane są głównie w celu poprawy skuteczności szeregowania w modelu a) - patrz rys. 2. h), i).

2.6. Podział ze względu na poziom zaangażowania egzekutora

Egzekutor może być w różnym stopniu zaangażowany w przekierowywanie zapytań klienta. Może być również zaangażowany w proces przekazywania odpowiedzi. Rysują się tu cztery grupy modeli: bierne, zwrotne, jednokierunkowe oraz dwukierunkowe:

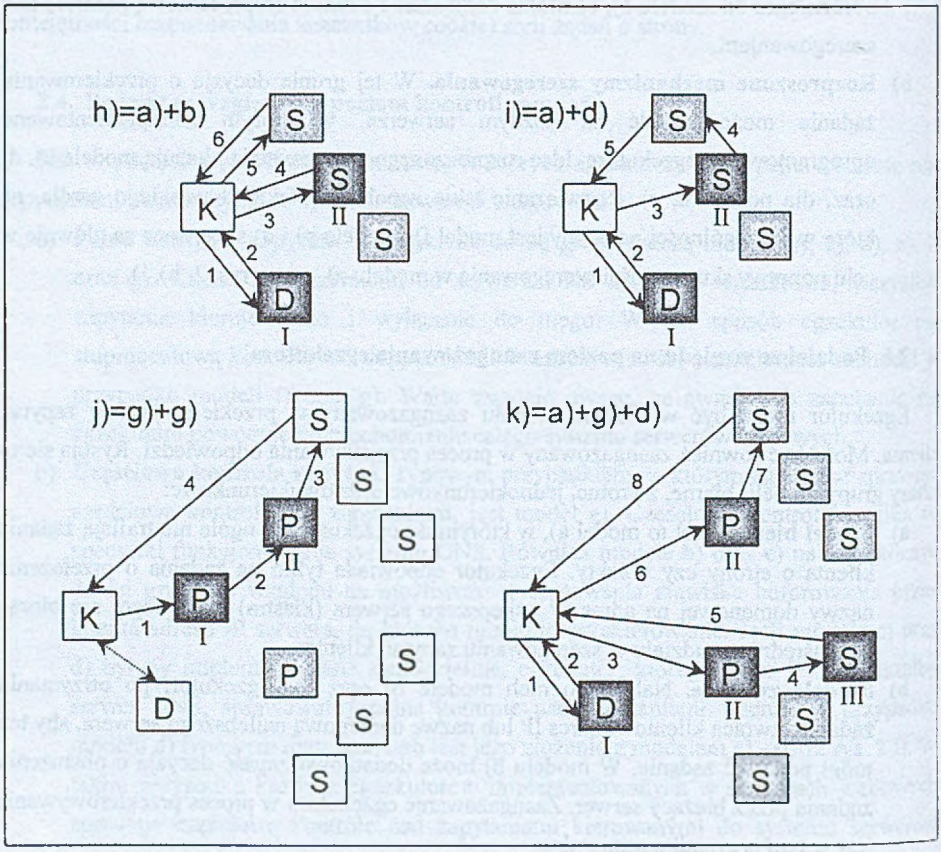
a) **Model bierny.** Jest to model a), w którym do egzekutora w ogóle nie trafiają żądania klienta o strony czy obiekty. Egzekutor odpowiada tylko na żądania o przełożenie nazwy domenowej na adres IP najlepszego serwera (klastra) webowego, nie biorąc bezpośredniego udziału w szeregowaniu zapytań klienta.

b) **Modele zwrotne.** Należą do nich modele b) oraz e). Egzekutor, po otrzymaniu żądania, zwraca klientowi adres IP lub nazwę domenową najlepszego serwera, aby ten mógł ponowić żądanie. W modelu b) może dodatkowo zapaść decyzja o obsłużeniu żądania przez bieżący serwer. Zaangażowanie egzekutora w proces przekierowywania jest w tym przypadku minimalne.

c) **Modele jednokierunkowe.** Należy do nich model d) oraz g) [15, 20, 21]. Żądania, które osiągają egzekutor, przekierowywane są bezpośrednio do serwerów webowych. W modelu d) może dodatkowo zapaść decyzja o obsłużeniu żądania przez bieżący serwer. Dzięki odpowiednim zabiegom technicznym serwery kierują odpowiedzi

bezpośrednio do klienta. Zaangażowanie egzekutora w proces przekierowywania jest w tym przypadku średnie.

d) **Modele dwukierunkowe.** Należy do nich model c) oraz f) [1, 11, 17]. Wszystkie żądania, które osiągają egzekutor, przekierowywane są bezpośrednio do serwerów webowych. W modelu d) może dodatkowo zapaść decyzja o obsłużeniu żądania przez bieżący serwer. Serwery webowe wszystkie odpowiedzi przesyłają do egzekutora, który musi je przekierowywać do klienta. Zaangażowanie egzekutora w proces przekierowywania jest w tym przypadku na tyle duże, że model c), ze względu na przeciążenia serwerów, staje się praktycznie bezużyteczny.



Rys.2. Złożone modele funkcjonalne skalowalnych systemów serwerów webowych
Fig.2. Complex functional models of scalable Web server systems

2.7. Podział ze względu na liczbę stopni szeregowania

Systemy serwerów webowych mogą jednocześnie wykorzystywać jeden lub więcej mechanizmów szeregowania. Mówi się wówczas o skalowalnych systemach serwerów webowych z szeregowaniem jednostopniowym, dwustopniowym lub trójstopniowym. Nie spotyka się systemów o większej ilości stopni szeregowania. Modele systemów z szeregowaniem wielostopniowym można budować poprzez złożenie modeli prostych. Przykładowe modele złożone zostały przedstawione na rys. 2.

a) **Systemy z szeregowaniem jednostopniowym.** Systemy z szeregowaniem jednostopniowym odzwierciedlają modele a), e), f) oraz g). W modelu a) egzekutor zintegrowany jest z serwerem DNS, natomiast w modelach e), f) i g) rolę egzekutora pełni dystrybutor lub przełącznik webowy.

b) **Systemy z szeregowaniem dwustopniowym.** Przykładowe systemy z szeregowaniem dwustopniowym obrazują modele h), i) oraz j).W przypadku h) [2, 10, 23] oraz i) [3] na pierwszym stopniu wykorzystywany jest model a), natomiast na drugim, odpowiednio model b) lub d). W obu przypadkach na pierwszym stopniu egzekutor zintegrowany jest serwerem DNS, natomiast na drugim rolę egzekutora pełni ten z serwerów webowych, który został wytypowany na poziomie serwera DNS. Przypadek j) ukazuje kaskadowe złożenie modeli g) w celu szeregowania żądań w systemie rozproszonych klastrów webowych [21]. Na obu stopniach rolę egzekutora może pełnić dystrybutor lub przełącznik webowy. Model ten charakteryzuje się krótką drogą zapytań, jednak awaria pierwszego stopnia szeregowania unieruchamia cały system rozproszonych klastrów.

c) **Systemy z szeregowaniem trójstopniowym.** Przykład systemu z szeregowaniem trójstopniowym pokazuje model k) [9]. Jest to system rozproszonych klastrów webowych, gdzie każdy z serwerów w klastrze ma wbudowany mechanizm szeregowania. Na pierwszym stopniu szeregowania wykorzystywany jest model a), na drugim model g), a na trzecim model b), czyli na pierwszym stopniu egzekutor zintegrowany jest z serwerem DNS, na drugim role egzekutora pełni dystrybutor lub przełącznik webowy, natomiast na trzecim ten serwer webowy, który został wytypowany przez egzekutor na drugim stopniu. Każdorazowe zadziałanie trzeciego stopnia szeregowania, może powodować znaczne opóźnienia, jednak model ten jest bardzo odporny na przeciążenia i awarie.

3. Taksonomia webowych algorytmów szeregowania

Wszystkie mechanizmy realizujące zarządzanie dostępem do rozproszonych serwerów WWW muszą mieć zaimplementowany konkretny algorytm, na podstawie którego będą mogły podejmować decyzję o przesłaniu zapytania do *najlepszego* serwera. W zależności od stopnia skomplikowania systemu i możliwości implementacyjnych wykorzystywane algorytmy mogą mieć różną złożoność obliczeniową. Zastosowane strategie zależne są również od informacji, jakie posiada system na temat ilości zapytań generowanych przez klientów oraz informacji o stanie serwerów wchodzących w skład rozproszonego systemu WWW. Można je podzielić na kilka kategorii w zależności od ilości informacji, jaka wykorzystywana jest przy podejmowaniu decyzji.

3.1. Algorytmy statyczne (nie wykorzystujące informacji zewnętrznych)

Jeżeli w procesie szeregowania nie są wykorzystywane żadne informacje o stanie serwerów lub strumieniu zapytań, mówi się o algorytmach statycznych. Te mechanizmy, to najprostsze rozwiązania gwarantujące w niewielkim stopniu rozłożenie obciążenia pomiędzy serwery. Rozkład ten jest bardzo przypadkowy i nie gwarantuje równomiernego wykorzystania zasobów ani wysokiej dostępności systemu. Do takich algorytmów należą:

- **Random** – system nie posiada żadnych informacji o stanie serwerów WWW, jak również nie posiada informacji o tym, do którego serwera ostatnio skierowane zostało zapytanie;
- **Round-robin** – system nie posiada żadnych informacji o stanie serwerów WWW, posiada jednak informację historyczną o tym, do którego serwera skierowane zostało ostatnio nadesłane zapytanie.

3.2. Algorytmy dynamiczne (wykorzystujące informację zewnętrzną)

W procesie szeregowania można stosować bardziej rozbudowane mechanizmy zarządzania. Aby zwiększyć efektywność szeregowania, można stosować strategie wykorzystujące informacje o kliencie. Wybór najlepszego serwera może odbywać się na podstawie adresu IP lub numer portu TCP wykorzystywanego przez klienta. Możliwe jest również wykorzystywanie tzw. alarmów generowanych przez serwery. Do mechanizmu decydującego o wyborze serwera w określonych odstępach czasowych przekazywana jest informacja zwrotna o obciążeniu serwera lub informacja o przekroczeniu jakiegoś określonego parametru. Szeregowanie może odbywać się na podstawie jednej z metryk określających obciążenie:

- ilości aktywnych połączeń z serwerem,
- wykorzystania dysku lub procesora serwera,
- przewidywanego strumienia zapytań w określonym czasie.

W przypadku modeli, gdzie mechanizmy szeregowania w całości kontrolują strumień zapytań, mogą być stosowane algorytmy dystrybuujące zapytania w zależności od wielkości napływającego strumienia oraz obciążenia serwerów obsługujących te zapytania. Dzięki informacjom zbieranym od serwerów możliwe jest szacowanie parametrów mających wpływ na szybkość ich odpowiedzi. Mogą to być:

- ilość zapytań zrealizowana w określonym czasie,
- obciążenie procesora w danym momencie,
- poziom wykorzystania dysków w serwerze.

Takie rozwiązania umożliwiają wyeliminowanie przeciążonych serwerów. W znacznym stopniu zwiększa to dostępność klientów do danych oraz wpływa na lepsze wykorzystanie zasobów.

3.3. Algorytmy szeregowania wykorzystywane po stronie serwera DNS

3.3.1. Round-robin DNS

Podejście to zostało zastosowane jako pierwsze przy budowie skalowalnego systemu serwerów webowych przez NCSA [22]. Kod serwera DNS został tak zmodyfikowany, aby realizować algorytm **Round-robin**. Obciążenie serwerów nie jest zbyt dobrze równoważone ze względu na mechanizm *cache* dla powiązań *nazwa domenowa – adres IP*. Stosując takie podejście, nie ma się również kontroli nad dostępnością serwerów oraz nie uwzględnia się możliwości zastosowania serwerów o różnej wydajności.

3.3.2. Algorytmy wykorzystujące informacje o stanie systemu

Alternatywą dla algorytmu **Round-robin** najczęściej stosowanego w przypadku serwera DNS są algorytmy wykorzystujące informacje o stanie systemu (wielkości przewidywanego strumienia zapytań, aktualnego obciążenia serwerów). Okazuje się, że reguły bazujące wyłącznie na stanie obciążenia serwerów WWW są nieefektywne. W rzeczywistości, w związku z hierarchicznym buforowaniem danych przez serwery DNS, pojedyncze zapytanie od klienta może spowodować napływ wielu innych zapytań. Wynika z tego, że informacja o aktualnym obciążeniu nie jest w żaden sposób związana z przyszłym obciążeniem.

Podejście pozwalające oszacować nadchodzący strumień zapytań (hidden load weight) w czasie TTL (Time To Live) jest najbardziej efektywne. Dzięki tym informacjom można przypisać różne wartości TTL dla różnych domen i estymować wielkość ukrytego, czyli nie

kontrolowanego przez serwer DNS strumienia zapytań. Przykładami tego typu algorytmów są [14]:

- **Multi tie round-robin** – dla każdej domeny lub ich określonych zbiorów przypisywane są różne wartości TTL, pozwalające na zrównoważenie strumieni zapytań do każdego serwera WWW.
- **Dynamically accumulated load (DAL)** – rozbudowana wersja algorytmu Multi tie round-robin polegająca na zbieraniu informacji o estymowanej wielkości obciążenia serwerów i zmianie łańcucha przypisań adresów w serwerze DNS w zależności od zakładanego obciążenia.
- **Minimum residual load** – modyfikacja algorytmu DAL polegająca na estymowaniu wielkości ukrytego obciążenia i wartości TTL. Po upływie TTL serwer WWW odpytywany jest o rzeczywistą wielkość strumienia zapytań, jaka została do niego skierowana. Dzięki temu algorytm ma informację o stanie serwera. Powoduje to również zmianę kolejności przypisań w serwerze DNS.

3.3.3. Algorytmy adaptacyjne

Algorytmy równoważenia bazujące na oszacowaniu *hidden load weight* oraz alarmach z krytycznie przeciążonych serwerów pozwalają na dużo bardziej efektywne, w porównaniu z algorytmami **Round-robin**, równoważenie serwerów WWW. Niestety, są one efektywne tylko w przypadku rozproszenia serwerów homogenicznych. Inną grupą algorytmów przeznaczonych do równoważenia obciążenia serwerów heterogenicznych są algorytmy adaptacyjne. W przypadku algorytmów adaptacyjnych wartość TTL przypisywana jest dynamicznie do każdego żądania na podstawie przewidywanego *hidden load weight* oraz wydajności serwera, do którego zapytanie zostało przypisane. Idea tego podejścia polega na uwzględnieniu pewnej wartości ξ_i opisującej moc obliczeniową serwera S_i. Algorytmy te podzielone zostały na dwie grupy [25]:

- **probabilistyczne** – realizacja metody bazuje na algorytmie **Round-robin**. Za każdym razem jest losowo generowana liczba γ $(0 \leq \gamma \leq 1)$. Jeżeli zapytania przypisane są aktualnie do serwera S_i, to jako następny zostanie wybrany S_{i+1} tylko wtedy, gdy $\gamma \leq \xi_i$, w przeciwnym przypadku warunek zostaje rozpatrzony dla kolejnego serwera. Wartość TTL oblicza się ze wzoru:

$$TTL_i = \frac{\eta}{\xi_i}. \tag{1}$$

- **deterministyczne** – ideą algorytmu jest takie dostosowanie czasu TTL, aby bardziej wydajne serwery obsługiwały więcej zapytań, a mniej wydajne nie były przeciążane.

Dla każdej domeny *i* obsługiwanej przez system serwerów webowych zostaje przypisana wartość TTL z uwzględnieniem wydajności serwera *j* według wzoru:

$$TTL_{ij} = \frac{\eta * c_i}{\xi_j}, \tag{2}$$

gdzie η jest wartością stałą, zależną od liczby zdefiniowanych domen generujących zapytania.

3.4. Algorytmy szeregowania wykorzystywane w dystrybutorach

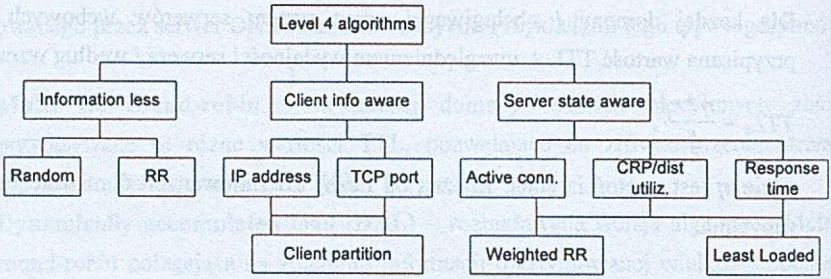
Stosowane tutaj algorytmy są najprostsze, ponieważ dystrybutor obsługuje wszystkie nadchodzące pakiety i konieczne jest zminimalizowanie czasu poświęcanego na ich obsługę. Rekompensatą za konieczność obsługi wszystkich pakietów jest pełna kontrola napływającego strumienia zapytań. Stosowane w tym przypadku algorytmy można podzielić na trzy podstawowe grupy:

- **Information less** – nie wykorzystujące żadnych informacji o stanie systemu,
- **Client info aware** – wykorzystujące informacje o adresie IP klienta lub numerze portu, przez który komunikuje się z serwerem,
- **Server state aware** – wykorzystujące informacje o stanie serwera, do którego przekazują zapytanie.

Tak jak w przypadku algorytmów stosowanych w serwerze DNS, najmniej efektywne są algorytmy nie wykorzystujące żadnych dodatkowych informacji, czyli **Random** i **Round-robin**.

Wykorzystując informacje o kliencie, np. jego adres IP, można zastosować bardziej efektywne rozwiązania. Przykładem takiego algorytmu jest **Client partition**.

W związku z zarządzaniem strumieniem zapytań na poziomie pakietów, odwołania nadchodzące z tego samego adresu, dotyczące pojedynczej sesji, muszą być przekierowywane do tego samego serwera. Dopiero żądania nowego obiektu mogą być przesłane na inny serwer. Wymaga to utrzymywania w systemie tablicy aktywnych połączeń. Dzięki temu można oszacować, który z serwerów obciążony będzie dodatkowymi zapytaniami w najbliższym czasie i na tej podstawie wybrać *najlepszy*, do którego skierowane zostanie kolejne zapytanie. Realizowane jest to na podstawie algorytmu **Least loaded server**.



Rys.3. Algorytmy stosowane w dystrybutorach [13]
Fig.3. Algorithms used in Level 4 Web-switch

Oprócz informacji o tym, kto przesyła zapytania do systemu webowego, można również uzyskać dane na temat pracy poszczególnych serwerów w systemie. Aby poprawić jakość szeregowania zapytań, w algorytmie dokonującym wyboru *najlepszego* serwera stosuje się różnego rodzaju metryki określające aktualne obciążenie serwerów. Najczęściej stosowanymi algorytmami w tym przypadku są **Least loaded server** oraz **Weighted round-robin**.

W przypadku algorytmu **Least loaded server** dzięki wybranemu kryterium (określonej metryce) wiemy, który z serwerów powinien przyjąć kolejne zapytanie. Odbywa się to na zasadzie: *najmniej obciążony serwer przyjmuje kolejne zlecenie*. Stosując algorytm **Weighted round-robin**, przy wyborze kolejnego serwera można uwzględniać metrykę jako parametr funkcji wyboru *najlepszego* serwera. Ważnym więc elementem funkcjonowania tego rozwiązania jest dobór metryki będącej powyższym parametrem. Istnieje możliwość kontrolowania następujących parametrów:

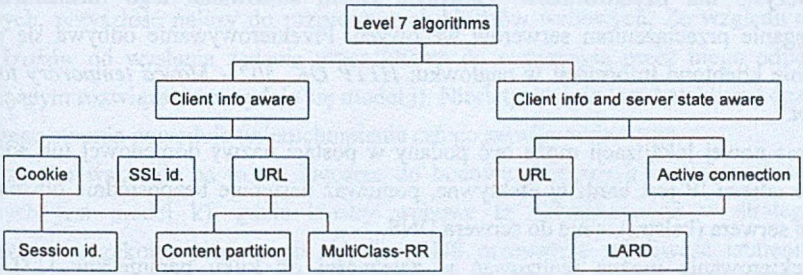
- **input metric** – informacja pozyskiwana jest przez dystrybutor bez współpracy z serwerami, np. liczba aktywnych połączeń miedzy dystrybutorem a poszczególnymi serwerami,
- **server metric** – informacja pozyskiwana jest przez serwery i dostarczana dystrybutorowi, np. wykorzystanie procesora lub dysku, czas miedzy otrzymaniem zapytania a wysłaniem odpowiedzi,
- **forward metric** – informacja pozyskiwana jest bezpośrednio przez dystrybutor, np. emulacja zapytań do serwerów webowych.

3.5. Algorytmy szeregowania wykorzystywane w przełącznikach webowych

Przełączniki webowe mają również możliwość kontrolowania 100% ruchu do serwerów. Zatem i tu konieczne jest wykorzystanie jak najprostszych mechanizmów zarządzania. W przypadku prostych rozwiązań wystarczająco wydajne są algorytmy statyczne. Sytuacja taka

występuje, gdy czasy realizacji zleceń przez serwery WWW są bardzo zbliżone do siebie i nie wychodzą poza określony przedział wartości.

W przypadku gdy w systemie występują więcej niż dwa ograniczenia czasu obsługi zlecenia, należy stosować algorytmy dynamiczne wykorzystujące informacje o kliencie lub stanie serwera (client info or server state aware). Mając do dyspozycji heterogeniczne środowisko serwerów trudno jest wybrać najlepszy, wspólny dla wszystkich parametr określający metrykę obciążenia serwera. W takim przypadku preferowane są algorytmy wykorzystujące informacje pochodzące od klientów.



Rys. 4. Algorytmy stosowane w przełącznikach webowych [13]
Fig. 4. Algorithms used in Level 7 Web-switch

Jak wynika z rysunku 4, istnieją trzy rodzaje algorytmów opartych na informacjach o kliencie:

- **Session identifiers** – odwołania HTTP, mające ten sam identyfikator SSL lub ten sam znacznik *cookie* przypisywane są do tego samego serwera – zmniejsza to czas konieczny na ponowną identyfikację klienta,
- **Content partition** – podział zasobów serwerów może nastąpić ze względu na:
  - typy plików – dane, pliki graficzne, pliki audio umieszczone są na specjalizowanych serwerach,
  - wielkość plików – duże pliki na szybszych serwerach lub równomierne rozłożenie plików w przypadku serwerów homogenicznych,
- **Multi-class round-robin** – zasoby są podzielone ze względu na złożoność obliczeniową i czasową, jaka zostanie wygenerowana podczas ich obsługi, np. połączenia szyfrowane wymagają mocy obliczeniowej procesora, odwołania do bazy danych wymagają zwiększonej obsługi dysków, czy wreszcie pobieranie dużych plików w znacznym stopniu obciąża sieć.

Zasada działania algorytmu wykorzystującego informacje o kliencie i serwerze jest następująca. Pierwsze zapytanie klienta o zasoby przekierowywane jest według algorytmu **Least loaded server** (metryką jest ilość aktywnych połączeń z serwerem). Pozostałe zapytania klienta o ten sam zasób przekierowywane są do tego samego serwera. Dzięki temu

zwiększona jest skuteczność odwołań do pamięci podręcznej (cache) tego serwera. Algorytm ten zwany jest **Locality-Aware Request Distribution (LARD)**.

3.6. Algorytmy szeregowania wykorzystywane w przypadku przekierowań na poziomie protokołu HTTP

Gdy stosuje się rozwiązania oparte na warstwowej strukturze, istnieje możliwość zarządzania zapytaniami z poziomu protokołu HTTP [18]. Takie rozwiązanie jest przeźroczyste dla użytkowników. Głównym celem stosowania tego mechanizmu jest zapobieganie przeciążeniom serwerów webowych. Przekierowywanie odbywa się poprzez przesłanie klientowi informacji w nagłówku: *HTTP OK. 302 - Moved temporary to a new location.*

Adres nowej lokalizacji może być podany w postaci nazwy domenowej lub adresu IP. Podanie adresu IP jest bardziej efektywne, ponieważ następuje bezpośrednie odwołanie do nowego serwera (kalstra), a nie do serwera DNS.

Przekierowania można realizować w zależności od kilku parametrów [13]. Proces przekierowań może dotyczyć:

- wszystkich stron,
- tylko stron przekraczających określoną wielkość,
- tylko stron, których ilość obiektów składowych przekracza określoną liczbę,

Wybór serwera, który powinien przejąć zapytanie, może odbywać się z wykorzystaniem jednej z poniższych strategii:

- **Round-robin**,
- **Least Loaded**,
- **Hash function**,
- **Client to server proximity**.

4. Podsumowanie

Przedstawiona w rozdziale 2 taksonomia skalowalnych serwerów webowych ukazuje bogactwo rozwiązań w tej dziedzinie, ale również mnogość czynników, na które trzeba zwracać uwagę przy budowie rzeczywistych systemów. Obok architektury systemu, nie mniej ważną dziedziną są algorytmy stosowane do równoważenia obciążeń serwerów webowych scharakteryzowane w rozdziale 3.

Analizując przedstawione w pracy modele i algorytmy trzeba mieć świadomość, że skończyła się era serwisów webowych zwracających co jakiś czas komunikat: *...not*

responding. Server may have crashed or is overloaded. Serwisy webowe stały się ważnym kanałem dystrybucji krytycznych informacji oraz narzędziem do prowadzenia interesów. Firmy czy instytucje zaczęły być oceniane na podstawie jakości swoich serwisów webowych. Pojawiło się w związku z tym kryterium QoWS (Quality of Web Service). Dotyczy ono przede wszystkim *dostępności* (np.: serwis musi być dostępny przez 99% czasu, czyli przestój nie może być dłuższy niż 7,2 godziny w miesiącu) oraz *wydajności* (np.: na 95% zapytań do serwisu, czas odpowiedzi musi być krótszy niż 4 sekundy).

Nie ulega wątpliwości, że w związku z wykładniczo rosnącą popularnością serwisów webowych, przyszłość należy do rozproszonych klastrów webowych. Ze względu na małą liczbę *kroków* od wysłania żądania przez klienta do otrzymania przez niego odpowiedzi, interesującym rozwiązaniem wydaje się model j). Niestety, awaria (np. kataklizm) egzekutora pierwszego stopnia powoduje unieruchomienie całego serwisu webowego.

Wszystko wskazuje na to, że kluczem do budowy *dostępnych* i *wydajnych* serwisów webowych jest model k), gdzie klastry webowe są rozmieszczone w strategicznych miejscach, a egzekutor pierwszego stopnia (DNS przewiduje możliwość istnienia wielu awaryjnych kopii serwera) odsyła adresy IP klastrów najbliższych klientowi. Jeśli w klastrach egzekutorami byłyby przełączniki webowe (mające świadomość treści żądania), to egzekutory trzeciego stopnia, zamiast być umieszczane w serwerach webowych, mogłyby być zintegrowane z egzekutorami drugiego stopnia. Przyspieszyłoby to i uprościło proces szeregowania żądań klientów.

W Zakładzie Rozproszonych Systemów Komputerowych w Instytucie Sterowania i Techniki Systemów zostały już przeprowadzone pierwsze eksperymenty związane z problematyką projektowania i eksploatacji webowych systemów informatycznych [6]. Problematyka równoważenia obciążeń serwerów WWW została również przedstawiona w pracy [4].

LITERATURA

1. Anderson E., Patterson D., Brewer E.: The Magicrouter, an application of fast packet interposing. University of California, Berkley 1996.
2. Andresen D., Yang T., Holmedahl V., Ibarra O.: SWEB: Toward a scalable World Wide Web server on multi-computers. Proc. of 10th IEEE Int'l. Symp. on Parallel Processing, Honolulu 1996.
3. Bestavros A., Crovella M. E., Liu J., Martin D.: Distributed Packet Rewriting and its application to scalable Web server architectures. Proc. of 6th IEEE Int'l. Conf. on Network Protocols (ICNP'98), Austin 1998.

4. Borzemski L., Nowak Z., Porczyński R.: Metody, algorytmy i rozwiązania systemowe równoważenia obciążeń serwerów WWW. ZN Pol. Śl. Studia Informatica Vol. 21, No 1(39), Gliwice 2000.
5. Borzemski L., Nowak Z., Porczyński R.: The Architectures and Algorithms for Load Balancing on Web-Server Systems. ISAT'2000, Wrocław 2000.
6. Borzemski L.: Pomiary wydajności klastra serwerów WWW z równoważeniem obciążenia. ZN Pol. Śl. Studia Informatica Vol. 22, No 3(45), Gliwice 2001.
7. Brisco T.: DNS Support for Load Balancing. RFC 1794, 1995.
8. Cardellini V., Colajanni M., Yu P.: Dynamic Load Balancing on Web-server Systems. IEEE Internet Computing, vol. 3, no. 3, 1999.
9. Cardellini V., Colajanni M., Yu P.: Geographic Load Balancing for Scalable Distributed Web Systems. Proc. IEEE Mascots 2000, San Francisco 2000.
10. Cardellini V., Colajanni M., Yu P.: Redirection algorithms for load sharing in distributed Web-server systems. Proc. of 19th IEEE ICDCS, Austin 1999.
11. Cisco's LocalDirector. http://www.cisco.com/warp/public/cc/pd/cxsr/400/index.shtml
12. Cisco's DistrbutedDirector. http://www.cisco.com/warp/public/cc/pd/cxsr/dd/index.shtml
13. Colajanni M., Yu P., Cardellini V.: Scalable Web-server systems: Architectures, models and load balancing algorithms. ACM Sigmetrics 2000, Santa Clara 2000.
14. Colajanni M., Yu P., Dias D.: Analysis of task assignment policies in scalable distributed web-server systems. IEEE Trans. on Parallel and Distributed Systems, Vol. 9 Is. 6, 1998.
15. Damani O., Chung P., Hang Y., Kintala C., Wang Y.: ONE-IP: Techniques for hosting a service on a cluster of machines. Journal of Comp. Net. and ISDN Syst., vol. 29, 1997.
16. Dias D., Kish W., Mukheree R., Tewari R.: A scalable and highly available Web server. Proc. of 41st IEEE Computer Society Int'l. Conf. (COMPCON96), 1996.
17. Egevang K., Francis P.: The IP Network Address Translator (NAT), RFC 1631, 1994.
18. Fielding R., Gettys J., Mogul J., Frystyk H.,Masinter L., Leach P., Berners-Lee T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, 1999.
19. Garland M., Grassia S., Monroe R., Puri S.: Implementing Distributed Server Groups for the World Wide Web. Tech. Rep. CMU-CS-95-114, Carnegie Mellon Univ., 1995.
20. Hunt G., Goldszmidt G., King R., Mukherjee R.: Network Dispatcher: A Connection Router for Scalable Internet Services. Proc. of 7-th Int. WWW Conf., Brisbane, 1998.
21. IBM SecurWay Network Dispatcher Version 2.1. White Paper, IBM, 1999.
22. Katz E., Butler M., McGrath R.: A Scalable HTTP Server: The NCSA Prototype. First International Conference on the World Wide Web, CERN, Geneva 1994.
23. Mourad A., Huiqun L.: Scalable Web server architectures. Second IEEE Symposium on Computers and Communications, Proceedings, 1997.

24. Narendran B., Rangarajan S., Yajnik S.: Data distribution algorithms for load balanced fault-tolerant Web access. Proc.16-th Symposium on Reliable Distributed Systems, 1997.
25. Porczyński R.: Wybrane elementy architektury globalnych serwerów WWW. ZN Pol. Śl. s. Informatyka z. 34, Gliwice 1998.

Recenzent: Dr inż. Rafał Cupek

Wpłynęło do Redakcji 30 marca 2001 r.

Abstract

In this article we presented taxonomy of load balancing systems for Web servers. In load balancing system we differ three elements:

- scheduling mechanism,
- scheduling algorithm,
- executor (to carry out scheduling algorithm).

Taxonomy of load balancing system for Web servers was presented in two parts. First part considers scheduling mechanisms with executors. Second considers algorithms. This is logical because architectures are build from scheduling mechanisms with different executor. There can be used some algorithms depends on what information system is aware of. Load balancing architectures were divided into seven main categories depends on:

- position (place) of scheduling mechanism,
- position (place) of Web server,
- what does executor (IP packet, HTTP request, DNS request) redirect,
- how executor controls requests (full control, only DNS request),
- place of executor, central or distributed scheduling,
- which element of HTTP request does executor control,
- scheduling levels in load balancing system.

Fig. 1. presents general and theoretical set of basic models of load balancing systems. Next we describe ability of those models. Fig. 2. presents set of complex models of load balancing systems. Those systems can be implemented. It shows how basic models can be joins to achieve better performance and availability.

Taxonomy of load balancing algorithm was based on place where algorithm is used and what information system is aware of. Load balancing algorithm were divided into six main categories:

- static (no information about system is used),
- dynamic (some information about system is used),
- DNS algorithm,
- Dispatcher algorithm,
- Web switch algorithm,
- HTTP redirection algorithm,

First two categories are based categories because those algorithms are used in any of executor. The difference is what parameter is used to describe system state. Performance metric describes the type of algorithm.