Linux Virtual Server

Projekt z przedmiotu RSO

Jan Boboli
jboboli@gmail.com

17 czerwca 2005

Spis treści

1 Omówienie LVS

LVS Na podstawie artykułu pt. Łinux Virtual Server for Scalable Network Services”http://www.LinuxVirtualServer.org

1.1 Wstęp

Wraz z rozwojem Internetu pojawiło się zapotrzebowanie na rozwiązania dające możliwość łatwego zwiększania wydajności serwisów, jak i zapewnienia dostępności serwisu w przypadku częściowej jego awarii. Wymagania wobec sprzętu i oprogramowania wspierającego wysoko dostępne i skalowalne usługi można scharakteryzować w następny sposób:

- Skalowalność - wraz ze wzrostem obciążenia systemu, system musi być rozszerzalny dla spełnienia nowych wymagań.
- Dostępność 24x7 - usługa jako całość musi być dostępna 24 godzin przez 7 dni w tygodniu, niezależnie od częściowych awarii sprzętu i oprogramowania.
- Zarządzalność - niezależnie od wielkości systemu, musi byc on w łatwy sposób zarządzalny.
- Wydajność cenowa - system musi być tani w utrzymaniu i rozwoju.

Pojedyńczy serwer nie spełnia tych wymagań - jego rozszerzanie jest skomplikowane i powoduje czasowe wstrzymanie działania systemu, tak jak w przypadku jego awarii.

Rozwiązaniem, które spełnia postawione wcześniej wymagania jest grupa serwerów połączona w klaster, w którym węzły połaczone są szybką siecią. Są one łatwiejsze w skalowaniu wraz ze zmieniającym się obciążeniem systemu, a także poprzez dublowanie się zapewniają wysoką dostępność. Linux Virtual Server umożliwia łączenie wielu serwerów w jeden klaster w celu zapewnienia wyżej postawonych wymagań. Jest to oprogramowanie, które umożliwia rozdzielanie żądań przychodzących z Internetu wśród puli serwerów obsługujących żądania.

1.2 Architektura LVS

Architektura LVS jest trójwarstwowa, składa się z następujących elementów:

- Load balancer (*dzielnik obciązenia*) - jego funkcją jest rozdzielanie połączeń przychodzących od klientów pomiędzy serwery pracujące w klastrze.
- Server pool (*pula serwerów*) - składa się z serwerów, które odostępniają klientom usługi charakteryzujące cały klaster - są to np. ftp, poczta, www, streaming multimediów.
- Backend storage (*system przechowywania danych*) - najczęściej rozproszony system plików taki jak Koda lub GFS zapewniający spójność danych.

Load balancer rozdziela przychodzące połączenia pomiędzy pulę serwerów korzystając z różnych technik dzielenia obciążenia, o których będzie mowa później. Jego zadaniem jest też obsługa stanu tych połączeń oraz przekazywanie pakietów do puli serwerów. Cała praca wykonywana przez dzielnik obciązenia jest wykonywana w jego jądrze, co ma pozytywny wpływ na wydajność - dzięki temu LVS jest w stanie obsłużyć dużo więcej połączeń niż źwykłyśerwer - dzięki temu nie stanowi wąskiego gardła dla całego systemu.

Węzły puli serwerów mogą być replikowane dla potrzeb skalowalności lub wysokiej dostępności.

- Skalowalność osiąga się poprzez przezroczyste dodawanie lub usuwanie węzłów z puli serwerów - w momencie, gdy obciążenie obecnej puli serwerów jest bliskie przeciążenia, można bez rekonfiguracji i zatrzymania systemu dodać nowy serwer. Ponieważ serwery w takiej puli w przeważającej liczbie przypadków nie są ze sobą ściśle powiązane, skalowalność takiego rozwiązania powinna być w większości przypadków liniowa - aż do momentu, gdy load balancer stanie się wąskim gardłem systemu.

- Wysoka dostępność jest osiągana jest razem ze skalowalnością - tak jakby 'automatycznie' - w klastrze pracuje wiele serwerów wykonujących tego samego typu zadanie. Po stronie dzielnika obciążenia działają demony nadzorujące pracę poszczegołnych węzłów z puli serwerów - w przypadku wykrycia awarii jednego z nich, demon dzielnika obciążenia odpowiedzialny za rozdzielanie połączeń zaprzestaje kierowania nowych połączeń do nieczynnego węzła, do momentu, gdy węzeł ten zacznie działać na nowo.

- W celu utrzymania niezawodności klastra w przypadku awarii dzielnika obciążenia stosuje się zapasowy dzielnik obciążenia, który nasłuchując "bicia serca"głównego dzielnika może wykryć jego awarię. W przypadku wykrycia awarii, serwer zapasowy przejmuje rolę głównego dzielnika połączeń, podszywając się pod adres IP głownego dzielnika. Po odzyskaniu sprawności dotychczasowy load balancer może przyjąć rolę serwera zapasowego, albo z powrotem stać się głównym serwerem.

System składowania danych jest zwykle redundantnym, odpornym na błędy, systemem plików, takim jak GFS, Coda albo Internezzo. Te systemy także obsługują zagadnienia wysokiej dostępności i skalowalności systemów. Węzły puli serwerów traktują rozproszony system plików jak lokalny system plików. Potencjalnym problemem związanym z takim rozwiązaniem jest możliwość powstawania wyścigów pomiedzy aplikacjami działającymi na różnych węzłach serwera - w niektórych przypadkach konieczne jest zastosowanie zarządcy rozproszonych blokad, który może być częścią rozproszonego systemu plików, albo działać poza systemem plików. Uwzględnienie zarządcy rozproszonych blokad ułatwia pracę programistów, piszących aplikacje działające w takim klastrze.

2 Techniki równoważenia obciążenia

W projekcie LVS stosuje się trzy różne techniki równoważenia obciążenia:

- LVS/NAT - (*tłumaczenie adresów*)
- LVS/TUN - Tunneling (*tunelowanie*)
- LVS/DR - Direct Routing (*bezpośrednie rutowanie*)

2.1 LVS/NAT

NAT jest szeroko stosowany w usługach związanych z internetem, w związku z ograniczoną adresacją w sieciach Ipv4 oraz z zagadanieniami bezpieczeństwa - NAT jest wykorzystywany przez firewalle ograniczające dostęp z zewnątrz do komputerów sieci lokalnej. NAT odbywa się poprzez modyfikację nagłówków pakietów - klienty sieci

zewnętrznej korzystający z serwerów w sieci wewnętrznej myślą, że komunikują się bezpośrednio z serwerami, a serwery sądzą, że udzielają usług bezpośrednio klientom. Ta cecha NAT jest wykorzystywana w LVS - pula serwerów może być widziana jako jeden serwer. Architektura LVS/NAT wygląda następująco: Kiedy klient żąda usługi od klastra serwerów, wysyła pakiet z żądaniem usługi na virtualny adres. Pakiet ten jest odbierany przez dzielnik obciążenia, z puli serwerów wybierany jest serwer, który obsłuży to połączenie, w tabeli połączeń zapisywane jest to połączenie wraz z informacją, przez który serwer jest ono osługiwane. Następnie pakiet jest przekazywany do serwera - modyfikowana jest część nagłówka pakietu odpowiedzialna za docelowy adres IP oraz port. W przypadku, gdy nadchodzący pakiet jest częścią połączenia już obsługiwanego przez jeden z serwerów, pakiet ten jest przekazywany do wybranego serwera, bez tworzenia nowego zapisu o połączeniu. Pakiety od serwerów do klientów przechodzą przez dzielnik obciążenia - tam zmieniany jest ich adres i port nadawcy tak, by klient myślał, że pochodzą od jednego wirtualnego serwera. Kiedy połączenie jest zrywane lub nastąpi jego time-out, połączenie jest usuwane z tabeli połączeń.

2.2 LVS/TUN

Tunelowanie IP jest techniką umieszczenia datagramu IP w innym datagramie IP. Umożliwia to przekazywanie datagramów adresowanych do jednego adresu IP komputerowi umieszczonemu pod innym adresem IP. Ta technika może być wykorzystana do budowy wirtualnego serwera - dzielnik obciążenia tuneluje pakiet do swybranego z puli serwerów, a następnie pakiet ten wysyłany jest z powrotem bezpośrednio przez serwer do nadawcy. W rozwiązaniu tym serwery korzystające z tego rodzaju przekazywania pakietów muszą obsługiwać tuneling.

2.3 LVS/DR

Bezpośrednie rutowanie jest możliwe wtedy, gdy dzielnik obciążenia i wszystkie serwery znajdują się w tym samym fizycznym segmencie sieci - mogą być połączone hubem lub switchem. Wirtualny adres IP klastra jest dzielony przez wszystkie serwery w sieci - wszystkie serwery mają swój zwrotny adres IP skonfigurowany na wirtualny adres IP klastra, a dzielnik obciążenia ma jeden ze swoich interfejsów sieciowych skojarzony z adresem wirtualnego serwera. W tej konfiguracji dzielnik obciążenia odbiera wszystkie pakiety kierując je do serwerów zmieniając w ich nagłówkach adres MAC odbiorcy pakietu. Kiedy serwer odbiera taki pakiet, stwierdza, że przeznaczony jest on dla jego adresu loobpack i odpowiada na pakiet bezpośrednio do jego nadawcy. Interfejsy serwerów muszą być tak skonfigurowane, aby nie odpowiadały protokołem ARP - w przeciwnym przypadku dojdzie do wyścigu pomiędzy serwerami.

3 Porównanie wydajności LVS/NAT, LVS/TUN i LVS/DR

Porównanie LVS/NAT, LVS/TUN i LVS/DR przedstawia poniższa tabela:

	LVS/NAT	**LVS/TUN**	**LVS/DR**
Serwer	**dowolny**	**tunneling**	**nie-arp**
Sieć	**prywatna**	**LAN/WAN**	**LAN**
Liczba serwerów	**niska**	**wysoka (10*niska)**	**wysoka (10*niska)**
Gateway	**load balancer**	**router**	**router**

4 Wady i zalety

4.1 LVS/NAT

- *Zalety* W konfiguracji tej serwery mogą działać pod dowolnym systemem operacyjnym obsługującym protokół TCP/IP, dzielnik obciążenia musi posiadać jeden publiczny adres IP, a serwery dowolne adresy wewnątrz sieci.
- *Wady* ograniczona skalowalność - ze względu na potrzebę filtrowania pakietów od klientów do serwerów i w drugim kierunku, dzielnik obciążenia szybko staje się ”wąskim gardłem”klastra.

4.2 LVS/TUN

- *Zalety:* większa wydajność niż w konfiguracji NAT - load balancer nie musi przerabiać pakietów zwrotnych, serwery mogą być rozproszone geograficznie
- *Wady:* węzły klastra muszą obsługiwać tunneling - mniejsza elastyczność

4.3 LVS/DR

- *Zalety:* największa wydajność dzielnika a co za tym idzie największa skalowalność klastra. Dzielnik obciążenia nie musi przerabiać nagłówków na poziomie ip, wystarczy, że zmodyfikuje adres MAC odbiorcy pakietu.
- *Wady:* system operacyjny musi umożliwiać wyłączenie odpowiedzi ARP na interfejsie zwrotnym, poza tym cały klaster musi byc podłączony do jednego switcha / huba.

5 Szeregowanie połączeń

LVS zakłada następujące algorytmy rozdzielania połączeń pomiędzy węzły klastra:

- *Round-Robin Scheduling* Algorytm Round Robin polega na przeglądaniu cyklicznej listy węzłów i przydzielaniu kolejnego połączenia kolejnemu serwerowi na liście.
- *Weighted Round-Robin Scheduling* algorytm Weighted Round Robin różni się od poprzednika tym, że każdemu serwerowi przypisana jest waga, która określa jego pojemność - im większa waga, tym więcej połączeń będzie przydzielanych danemu serwerowi w stosunku do innych serwerów.
- *Least-Connection Scheduling* w tym algorytmie najwięcej połączeń jest przydzielanych serwerowi, który ma ich najmniej w danej chwili.
- *Weighted Least-Connection Scheduling* algorytm ten różni się od poprzednika tym, że każdemu z serwerów możemy przypisać wagę - tak jak w Weighted Round Robin.
- *Locality-Based Least-Connection Scheduling* szeregowanie tego rodzaju wykorzystywane w przypadku klastra cache - algorytm szeregowania kieruje pakiety skierowane do jednego adresu IP do serwera cache obsługującego ten adres IP

- o ile serwer ten nie jest przeciążony i działa. W momencie, gdy serwer nie może przyjąć zgłoszenia, wybierany jest inny serwer wg algorytmu WLCS.

- *Locality-Based Least-Connection with Replication Scheduling* różni się od poprzednika tym, że load balancer posiada informacje na temat przypisania docelowego adresu IP do grupy serwerów obsługujący ten adres IP. Żądania skierowane do docelowego adresu IP są kierowane do serwera z grupy obsługującej ten adres IP, który ma najmniej połączeń. Jeżeli wszystkie serwery z grupy są przeciążone, dzielnik wybiera inny serwer spoza grupy i dodaje go do tej grupy. Po jakimś czasie, przez który serwer nie jest modyfikowany, węzeł o największym obciążeniu usuwany jest z grupy, aby uniknąć problemów z replikacją.

- *Destination Hashing Scheduling* algorytm ten rozdziela połączenia do serwerów na podstawie tablicy mieszanej adresów serwerów.

- *Source Hashing Scheduling* algorytm ten rozdziela połączenia do serwerów na podstawie tablicy mieszanej adresów źródłowych klientów.

- *Shortest Expected Delay Scheduling* algorytm ten szereguje połączenia wg najkrótszego oczekiwanego czasu obsługi połączenia - oczekiwana długość jest wprost proporcjonalna do ilości połączeń i wagi węzła.

- *Never Queue Scheduling* szeregowanie to przebiega w następujący sposób - jeżeli w puli serwerów jest jakiś serwer, który jest wolny - połączenie kierowane jest do niego w pierwszej kolejności. Jeżeli zaś nie ma wolnego serwera, połączenie kierowane jest wg. algorytmu SEDS.

- *Local Node Feature* jest to możliwość obsługiwania części połączeń przez dzielnik obciążenia - z uwagi na fakt, że w większości przypadków serwer ten nie jest obciążony, można wykorzystać jego moc obliczeniową do obsługi zadań, odciążając pulę serwerów.

6 Zachowanie stałości połączenia

Do tej pory zakładaliśmy, że kolejne połaczenia sa od siebie niezależne. Niektóre aplikacje, np. korzystające z sesji HTTP wymagają, aby połączenia wewnątrz jednej sesji były kierowane do tego samego serwera. Rozwiązaniem tego problemu jest kierowanie pakietów z jednego adresu IP na jeden serwer z puli. Chociaż zaburza to trochę równowagę pomiędzy serwerami, jest to dobre rozwiązanie. Czasami zdarza się, że połączenia od klientów za nieprzezroczystymi serwerami proxy są kierowane przez różne serwery - w tym przypadku pakiety należące do tej samej sesji idą przez różne serwery proxy, więc nie mogą byc rozpoznane jako należace do jednego klienta. Istnieje jednak i na to rozwiązanie - zakładając, że różne serwery proxy tego samego providera znajdują się w jednej domenie, możemy określić domeny, z których pakiety będą brane żazem"i kierowane do jednego węzła klastra.

7 System automatycznego dostosowywania wag

(ang. *Automated Weight Correcting System*)

7.1 Wstęp

Jak już wyżej wspomniano, w algortymach 'ważonych', w których podaj się współczynniki wagowe można dopasować wagi poszczególnych serwerów tak, aby obciążenie pomiędzy róznej klasy serwery było w optymalny sposób rozdzielane. To rozwiązanie nie jest jednak doskonałe - w przypadku, gdy jeden z serwerów dostaje nietypowe zadanie do obsłużenia, wymagające więcej mocy, rozłożenie obciążenia nie jest optymalne dla danej chwili. Aby ulepszyć działanie algorytmów ważonych potrzebny jest system, który na bieżąco monitoruje pracę serwerów i w zależności od ich obciążenia tak dobiera wagi, aby wszystkie były jednakowo obciążone. Taki system udało się mi zaimplementować, aczkolwiek muszę zaznaczyć, że jego pomysłodawcą był wykładowca prowadzący przedmiot, w ramach którego projekt był wykonany.

7.2 Architektura

AWCS składa się z load managera - procesu działającego na maszynie load balancera i agentów działających po stronie rzeczywistych serwerów. Zadaniem agentów jest monitorowanie dostępnych zasobów serwera, obliczanie na podstawie pobranych danych o zasobach wagi serwera oraz przesyłaniu do load managera poleceń zmiany wagi konkretnego serwera. Dane o obciążeniu każdego serwera zbierane są poprzez określony czas i wysyłane do load managera co określony odstęp czasu. Load manager odbiera komendy od agentów i dokonuje akutalizacji wag. Dane,

7.3 Algorytm obliczania wag

Dane, które zbierane są przez agenty i wchodzące w proces obliczania wagi dla wybranego serwera to:

- stosunek czasu bezczynności procesorów do ogółu czasu procesora - obliczany z pliku /proc/uptime
- stosunek wolnej pamięci RAM do całkowitej pamięci RAM obliczany z pliku /proc/meminfo
- stosunek wolnej pamięci swap do całkowitej pamięci swap obliczany z pliku /proc/meminfo

Wszystkie powyższe wartości wyrażone są w procentach. Aby konfiguracja serwera była elastyczna, dodałem możliwość określania współczynników modyfikujących wartość każdej z powyższych danych. Aby podstawą obliczania wagi był tylko dostępny czas procesora, należy współcznynnik tej danej ustawić na wartość większą od 0, a pozostałe współczynniki odpowiedzialne za pamięć ustawić na 0. Dodatkowo, jeżeli mamy w klastrze dwie maszyny o różnej wydajności, należy podać wyższe wartości współczynników na bardziej wydajnej maszynie niż na mmiej wydajnej.

7.4 Komunikacja w systemie

Komunikacja pomiędzy agentami a load managerem prezbiega w standardowy sposób - load manager nasłuchuje na wybranym porcie, odb ierając zgłoszenia od agentów. Dla każdego połączenia wybierany jest wątek z puli wolnych wątków. Za każdym

razem, kiedy agent wysyła komendę aktualizacji wagi, nawiązuje połączenie z load managerem. Pierwszym elementem jest przesłanie hasła zapewniającego autoryzację. Następnie w kolejnych paczkach przesyłane są dane niezbędne do aktualizacji wagi : są to m. in. dane identyfikujące usługę oraz serwer. Odebranie każdej paczki potwierdzane jest przez serwer komunikatem 'OK' w przypadku powodzenia lub 'ERR' w przypadku błędu. Jeżeli jakaś paczka zostanie źle wysłana połączenie jest zrywane przez serwer i komenda zostanie wysłana dopiero po upływanie określonego interwału. Zastanówmy się, czy nie lepiej byłoby ponawiać próbę wysłanie paczki. uważam że nie, dlatego że w przypadku np. podania złego hasła bądź innych niepoprawnych danych agent próbowałby się łączyć po kilka razy, co mogłoby doprowadzić do obniżenia wydajności całego systemu.

7.5 Aktualizacja wag

Wątki load managera, aktualizując wagi serwerów komunikują się z jądrem poprzez bibliotekę liblvs, której źródła zawarte są w programie ipvsadm, służącym do interaktywnego zarządzania LVS-em. Sama biblioteka, w jej obecnej implementacji korzysta z wywołań funkcji systemowych, komunikując się z modułem LVS dołączonym do jądra.

7.6 Eksperyment

W celu przetestowania działania system uruchomiłem klaster składający się z dwóch serwerów i jednego load balancera. Serwery miały różną wydajność, do obliczania wag przez agenty wykorzystałem tylko dostepny czas procesora. Do symulacji wykorzystałem algorytm Weighted Round Robin, load balancer działał w trybie NAT. System zachowywał się tak jak powinien - maszyna o większych możliwościach dostawała więcej zadań niż ta o mniejszych mołżliwościach. Ponadto, aby zasymulować inne programy obciążające system wykonywałem program, który wykonywał długą pętlę zasypiając po jej wykonaniu na 3 sekundy. Maszyna, na której uruchomiony był program obciążający serwer zmniejszył swoją wagę.

Oto przykładowe wydruki z ekranu komputera: W monecie, gdy system był wyłączony, obydwa serwery miały równe wagi. Wydruk statystyk LVS:

Prot	LocalAddress:Port	Conns	OutBytes	%Conns
TCP	**192.168.2.13:8080**	**232**	**101996**	**100%**
	192.168.3.194:8080	**116**	**51054**	**50%**
	192.168.3.11:8080	**116**	**50942**	**50%**

- obciążenie mocniejszego serwera: - ok 50 % CPU
- obciązenie słabszego - ok 100 % CPU

Wydruk z load managera po włączeniu systemu: (odświeżanie co 10s)
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 19 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 3 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 23 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 15 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 6 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 7 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 20 OK

Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 19 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 45 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 9 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 6 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 34 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 22 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 38 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 19 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 2 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 71 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 16 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 94 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 55 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 75 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 15 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 5 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 64 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 27 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK

Można zaobserwować dużą oscylację wag. Spójrzmy na statystyki:

Prot	LocalAddress:Port	Conns	OutBytes	% Conns
TCP	**192.168.2.13:8080**	**1118**	**488538**	**100%**
	192.168.3.194:8080	**884**	**383842**	**79%**
	192.168.3.11:8080	**234**	**104696**	**21%**

Serwer mocniejszy otrzymuje więcej zgłoszeń. Spróbujmy skrócić czas odświeżania:

Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 49 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 57 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 100 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 98 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 96 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 98 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 99 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 10 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 100 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 98 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 34 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 15 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK

Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 100 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 99 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 60 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 100 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 81 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.11:8080 0 OK
Aktualizacja wagi: 192.168.2.13:8080 192.168.3.194:8080 22 OK
Tutaj także występuje dość duża oscylacja. Spójrzmy jeszcze raz na statystyki:

Prot	LocalAddress:Port	Conns	OutBytes	% Conns
TCP	**192.168.2.13:8080**	**286**	**124196**	**100%**
	192.168.3.194:8080	**170**	**73926**	**59%**
	192.168.3.11:8080	**116**	**50270**	**31%**

7.7 Wnioski do eksperymentu

Jak widać, przy dłuższym okresie odświeżania system bardziej rozsądnie przydziela wagi, można uznać, że system działa zgodnie z przewidywaniami.

Aby osiągnąć optymalny rezultat należałoby zastosować system sterowania wraz z członem stałym, całkującym i różniczkującym, dzięki czemu system zachowywałby się bardziej stabilnie.

Literatura