Porównanie własności różnych dostępnych rozwiązań klastrowych zbudowanych w oparciu o system Linux

Projekt z przedmiotu RSO

Anna Felkner
afelkner@elka.pw.edu.pl

26 września 2005

Spis treści

1 Wstęp

Klaster to grupa niezależnych komputerów (węzłów), połączonych szybką siecią komunikacyjną. Z punktu widzenia użytkownika sprawia wrażenie pojedynczego systemu (komputera). Klastry pracują pod nadzorem specjalnego oprogramowania, które potrafi rozdzielać zadania między wszystkie pracujące w nim maszyny. Instalacje klastrowe można znaleźć w instytutach badawczych, firmach zajmujących się efektami specjalnymi w filmach, bankach, systemach bilingowych firm telekomunikacyjnych, itd.

Ze względu na szeroki zakres zastosowań, powstało wiele rozwiązań klastrowych, jednakże do najważniejszych należą Beowulf, Mosix, OpenMosix, OpenSSI, Kerrighed, DragonflyBSD, Genezy, Plurix.

Jednolity obraz systemu (ang. *SSI - Single System Image*) jest to fizyczny lub logiczny mechanizm dający złudzenie, że zestaw rozproszonych zasobów (pamięć, dysk, procesor) tworzy jednolity zasób. W obecnej chwili termin SSI nie ogranicza się do jednego zasobu, ale jest coraz bardziej rozszerzany do wszystkich zasobów klastra.

SSI może być wprowadzony w życie na kilku poziomach: sprzętu komputerowego, systemu operacyjnego, warstwy pośredniej (middleware) oraz aplikacji. Zostaną tu przedstawione rozwiązania klastrowe oparte na systemie operacyjnym Linux. Będą to systemy, które dojrzały już do tego, aby były używane poza ośrodkami naukowymi. Niektóre z nich są często używane w przemyśle.

2 Podział klastrów

Podstawowy podział klastrów ze względu na pełnioną funkcję:

- **klastry wysokiej wydajności** (ang. *High Performance Computing*), zwane też klastrami do przetwarzania równoległego. Służą do masowego przetwarzania danych jednego rodzaju (np. danych naukowych lub procesów wizualizacji). Wymagają specjalnie przygotowanych programów tworzonych przy użyciu wyspecjalizowanych bibliotek programistycznych. Przykładem takiego klastra jest Beowulf.

- **klastry równoważące obciążenie** (ang. *Load Balancing Cluster*), zwane też klastrami serwerowymi. Przeznaczone są do utrzymywania bardzo obciążonych usług sieciowych (np. serwerów WWW) lub prostych zadań obliczeniowych. Ich główne zadanie polega na równoważnym dystrybuowaniu obciążenia między poszczególne serwery - węzły klastra. Tego typu klaster instaluje się w systemach, w których bardzo istotny jest czas reakcji na żądanie klienta. Do tej grupy należy Mosix.

- **klastry dużej dostępności** (ang. *High Availability Computing*). Klastry tego typu nie zwiększają wydajności serwisów, a mają jedynie eliminować tzw. pojedynczy punkt awarii (ang. *Single Point of Failure*) - w razie uszkodzenia jednego z serwerów jego zadania są, w sposób niewidoczny dla użytkowników, przejmowane przez inny węzeł klastra. Przykładem takiego klastra jest oprogramowanie *Red Hat High Availability Server* oraz oprogramowanie opracowane w ramach projektu *Linux High Availability*.

W praktyce rozwiązania klastrowe mają charakter mieszany i wykonują dla pewnych aplikacji funkcje wydajnościowe, przy jednoczesnej niezawodności. Często taki tryb pracy klastra dotyczy serwerów WWW, pocztowych, itp., z racji sposobu aplikacji obsługujących tego typu serwisy.

3 Systemy plików używane w klastrach

Zostaną tu przedstawione popularne rozproszone systemy plików dla systemu Linux. Nie wszystkie one zostały zaprojektowane do wykorzystania w klastrach, ale ich wydajność umożliwia użycie ich w tego typu modelach. Należą do nich GFS (ang. *Global File System*), Lustre, NFS (ang. *Network File System*), Coda, OpenGFS (ang. *Open Global File System*).

3.1 GFS (ang. *Global File System*)

GFS jest stosunkowo zaawansowanym i dojrzałym systemem plików dla klastrów wysokiej dostępności. Umożliwia on bezpieczny jednoczesny dostęp do plików w trybie do zapisu i odczytu przez wiele węzłów klastra. Charakteryzuje się dobrą skalowalnością i szybkim doprowadzenie systemu plików po awarii do spójnego stanu. Architektura systemu plików umożliwia wydajne, skalowalne blokowanie plików oraz wybór mechanizmu blokowania. W odróżnieniu od centralnego serwera metadanych, GFS wyklucza powstawanie wąskich gardeł. Jest zgodny ze standardem POSIX. Jest to system plików o zamkniętym źródle.

3.2 Coda

Coda to zaawansowany sieciowy, rozproszony system plików. Jest rozwijany na uniwersytecie Carnegie Mellon od 1987 roku. Ten system plików powstał z AFS wersji 2. Celem systemu Coda jest wysoka dostępność. W przypadku uszkodzenia części sieci serwerów umożliwia dalszą pracę. Może on pracować bez połączenia z siecią. Można korzystać z zasobów będąc od nich odłączonym (korzystając z plików roboczych). Po podłączeniu do sieci aktualizacja danych jest wykonywana automatycznie. Coda cechuje się dobrą skalowalnością. Jest przechowywana na replikowanych serwerach danych.

3.3 NFS (ang. *Network File System*)

NFS jest sieciowym, rozproszonym systemem plików zorientowanym na obsługę plików. Został stworzony przez firmę Sun Microsystems, a obecnie jest otwartym protokołem internetowym. Pracuje w środowisku heterogenicznym. Umożliwia współdzielenie systemów plików pomiędzy dowolną liczbą komputerów. Każda maszyna

może jednocześnie pełnić rolę klienta i serwera, może jednocześnie być serwerem eksportującym część swoich plików oraz klientem sięgającym po pliki na innych maszynach.

NFS oferuje przezroczystość dostępu (moduł klienta systemu NFS dostarcza lokalnym procesom interfejsu programowania aplikacji, który jest identyczny jak interfejs lokalnego systemu operacyjnego, położenia (przez spójną konfigurację montowania zdalnych katalogów u klientów), awarii (większość usług jest idempotenta), wydajności (intensywne wykorzystanie pamięci podręcznych) i migracji.

Nie oferuje natomiast przezroczystości zwielokrotniania, przezroczystości współbieżności (elementy są w NFS 4), ma ograniczoną skalowalność.

Lokalne drzewo katalogów serwera jest udostępniane przy pomocy modułu eksportu. Klienty instalują wyeksportowane katalogi w dowolnym miejscu własnego systemu plików. Komputer-klient nie ma możliwości ponownego wyeksportowania zamontowanego systemu plików NFS.

3.4 Lustre

Lustre jest aktywnie rozwijanym systemem. Powstał w 2001 roku w firmie HP, później projekt został przekazany na licencji Open Source. Nazwa pochodzi od połączenia słów Linux i Clusters. Celem utworzenia tego systemu było doprowadzenie do możliwości bezpiecznego obsłużenia bardzo dużej ilości węzłów w klastrach o wysokiej przepustowości. Do konfiguracji i logowania Lustre wykorzystuje standardy LDAP (ang. *Lightweight Directory Access Protocol*) i XML (ang. *Extensible Markup Language*). Dane przechowywane w systemie (specjalne pliki i katalogi) są traktowane jak obiekty. Właściwości obiektów (rozmiar, czas utworzenia, wskaźniki dowiązań symbolicznych, flagi rezerwowe) przechowywane są na serwerach metadanych (MDS). Metadane przechowywane są oddzielnie w stosunku do rzeczywistej zawartości obiektów. Serwery metadanych obsługują tworzenie plików, zmiany ich właściwości i są odpowiedzialne za obsługę obszaru nazw: plik może być odnaleziony poprzez wysłanie zapytania do serwera metadanych. Wszystkie implementacje Lustre są zgodne ze standardem POSIX (ang. *Portable Operating System Interface*).

3.5 OGFS (ang. *Open Global File System*)

OGFS, znany też pod nazwą OpenGFS, stosuje raportowany system plików oparty na blokach, który zapewnia dostęp do zapisu i odczytu wielu węzłom. Obecnie w OGFS można korzystać z dowolnego menedżera dysków logicznych. Memexp zastąpiono modułem OpenDLM (ang. *Distributed Lock Manager*). Poprzednia wersja memexp wymagała sporych zasobów obliczeniowych. System OGFS umożliwia rozrastanie się systemów plików oraz dołączanie nowych dysków twardych (poprzez osobny LVM - *Logical Volume Manager*). Uszkodzenia węzłów obsługiwane są przy pomocy odzyskiwania rejestru i izolowania uszkodzonego węzła.

3.6 Podsumowanie

Żaden z prezentowanych tutaj systemów nie jest idealny. Dzięki odpowiedniemu oprogramowaniu istnieje możliwość wyboru spośród różnych wariantów systemów plików, które można zastosować w klastrze.

Klastrowe i rozproszone systemy plików uzupełniają się ze sobą. Klastrowe systemy plików mogą się różnić między sobą, ale mają wspólną cechę, która odróżnia

je od rozproszonych systemów plików: muszą one w każdej chwili zapewniać jeden wspólny widok na cały system plików oraz pełną zgodność pamięci podręcznych.

Właściwości, którymi charakteryzują się klastrowe systemy plików (rozszerzając rozproszone systemy plików) to:

- zwiększona skalowalność (w większości środowisk klastrowych istnieje możliwość dodania nowych elementów klastra bez wyłączania już używanego systemu),
- równoważenie obciążeń (tzw. *load balancing*, ruch sieciowy między elementami klastra i obciążenie węzłów jest równoważone przy pomocy algorytmów badających stan dostępnych zasobów systemowych),
- lepsze zarządzanie (zarządzanie systemem klastrowym jest ułatwione dzięki temu, że system jest widziany jako jedna całość; można zarządzać sprzętem i oprogramowaniem w trakcie normalnego użytkowania sieci),
- odporność na awarie serwera (jego pracę może przejąć inny serwer),
- odporność na awarie sprzętu (w razie utraty dostępu do części sieci, można wybrać inną, niezależną drogę, która doprowadzi do danego zasobu; wykrywanie awarii sprzętowych i systemowych oraz redundancja gwarantują nieprzerwaną pracę),
- przezroczystość (użytkownik postrzega klaster jako jeden serwer niezależnie od tego, na jakim elemencie klastra faktycznie pracuje, ile elementów klastra jest dostępnych oraz ile z nich uległo awarii),
- niezawodność (węzły są odizolowane od siebie, więc uszkodzenie jednego z nich nie powoduje wystąpienia awarii w innym).

Wraz z rozwojem i popularyzacją klastrów, coraz mniejsze wymagania są stawiane na ich warstwę sprzętową. Większa niezawodność i skalowalność jest zapewniana w dużym stopniu przez oprogramowanie, dlatego serwery w klastrze, które często są drogimi zaawansowanymi serwerami, mogą być z powodzeniem zastąpione przez zwyczajne komputery osobiste.

4 Opis wybranych systemów klastrowych

Kerrighed powstał, jako wynik projektu badawczego rozpoczętego w 1999 roku. Oferuje on konfigurowalne globalne szeregowanie procesów. Używając narzędzi systemu Kerrighed, dedykowane polityki szeregowania, mogą być łatwo napisane i dynamicznie dodawane do klastra. Domyślna polityka szeregowania pozwala dynamicznie równoważyć obciążenia procesorów w klastrze przez używanie planu migracji procesów z wywłaszczeniem inicjowanym przez odbiorcę. Kiedy węzeł jest niedostatecznie obciążony, system dostrzega niezrównoważenie i proces migruje z mocno do niedostatecznie obciążonego węzła.

Mechanizm migracji w Krrighed bazuje na kilku mechanizmach, takich jak: tzw. process ghosting, kontenery, strumienie migrowane i rozproszone systemy plików.

Process ghosting jest używany do uzyskania informacji o stanie procesu i przechowywania odpowiadających mu danych na określonym urządzeniu. Tym urządzeniem

może być dysk (zrzucenie migawki procesu), sieć (migracja procesu albo zdalne tworzenie procesu) lub pamięć (podwojenie procesu albo zrzucenie migawki pamięci).

Mechanizm kontenerów jest używany do dzielenia danych pomiędzy węzłami, zapewniając jednocześnie spójność danych. Ten mechanizm jest używany do implementacji pamięci dzielonej, współdzielonego bufora podręcznego i rozproszonego systemy plików zwanego KerFS.

Mechanizm migrowalnych strumieni używany jest do wydajnej obsługi migracji komunikujących się procesów.

System **OpenMosix** bazuje na projekcie badawczym Mosix (ang. *Multicomputer Operating System for unIX*), który rozpoczął się w 1981roku na Uniwersytecie Hebrajskim w Jerozolimie. Do równoważenia obciążenia procesorów Mosix używa procesu migracji z wywłaszczeniem inicjowanym przez nadawcę. Mechanizm migracji procesów jest oparty na mechanizmie zastępowania: kiedy proces jest przeniesiony z jednego węzła na inny, zależność jest zachowana na oryginalnym węźle przez proces zastępczy. To zastępowanie pozwala na rozwiązanie wywołań systemowych (komunikacja sieciowa, dostęp do pliku) migrującego procesu, którego Mosix nie może rozwiązać lokalnie po migracji.

W systemie Mosix jest dostarczany algorytm globalnego zarządzania pamięcią. Jest on aktywowany, kiedy ilość wolnej pamięci w węźle spada poniżej wartości progowej. Wtedy następuje próba przeniesienia procesów do innego węzła, który ma wystarczającą ilość wolnej pamięci.

Mosix posiada swój własny klastrowy system plików o nazwie MosixFS (nazywany też MFS), który pozwala na dostęp do odległych dysków. Aby zwiększyć wydajność systemu plików po migracji, jest używany algorytm *Direct File System Access* (DFSA). Ten algorytm pozwala na bezpośrednią komunikację z głównym węzłem.

Mosix początkowo był rozwijany na systemie BSD. Został przeniesiony na system Linux w 1999 roku, a w 2002 wydzielił się z niego OpenMosix.

OpenSSI ukazał się w 2001 roku bazując na projekcie UnixWare NonStop Cluster, który powstał na bazie systemu Locus.

Celem zaprojektowania OpenSSI jest dostarczenie platformy, w której mogą być zintegrowane inne technologie klastrowe oparte na otwartym oprogramowaniu. Aktualna wersja OpenSSI zawiera kilka systemów plików i systemów zarządzania dyskami opartych na otwartym oprogramowaniu (GFS, OpenGFS, Lustre, OCFS, DRSD), rozproszony mechanizm blokowania (OpenDLM - *Distributed Lock Manager*) i mechanizm migracji pochodzący z systemu Mosix.

OpenSSI pozwala dynamicznie równoważyć obciążenie procesorów w klastrze. Mechanizm migracji OpenSSI używa procesu delegacji do obsługi IPC i wywołań systemowych po migracji procesu oraz klastrowy system plików (OCFS) do obsługi dostępu do otwartych plików.

5 Porównanie klastrów

W tej części zostaną porównane własności najważniejszych dostępnych rozwiązań klastrowych zbudowanych w oparciu o system Linux.

Oto wybrane kryteria porównawcze:

- Widoczny jako jednolity obraz systemu
- Globalne zarządzanie procesami

- Globalna komunikacja międzyprocesorowa
- Odporność na uszkodzenia i mechanizm migawki
- Wsparcie sprzętowe

5.1 Widoczny jako jednolity obraz systemu

W systemie OpenMosix używając polecenia mps i mtop można wyświetlić wszystkie procesy rozpoczęte w danym węźle, nawet jeśli zostały one przeniesione do odległego węzła. Jednak nie zostaną wyświetlone procesy, które rozpoczęły się w innych węzłach klastra. Za pomocą polecenia mtop nie jest możliwe wyświetlenie globalnej statystyki pamięci, jak również obciążenia procesorów w klastrze. Zakres przestrzeni nazw procesów jest oferowany dla każdego procesu uruchomionego w danym węźle początkowym. Nie jest zachowana unikalna przestrzeń identyfikatorów procesów w obrębie całego klastra.

W systemie OpenSSI polecenie ps i top umożliwia wyświetlenie wszystkich procesów uruchomionych w klastrze: procesy uruchomione lokalnie i zdalnie, gdziekolwiek znajduje się ich węzeł początkowy. Polecenie top nie umożliwia wyświetlenia globalnej statystyki pamięci, jak również obciążenie procesorów w klastrze. Wszystkie urządzenia klastra są umieszczone w katalogu /dev i są dostępne z każdego węzła. Unikalna przestrzeń identyfikatorów procesów w obrębie całego klastra oraz jednolita lista procesów, są zachowane.

W systemie Kerrighed polecenia ps i top umożliwia wyświetlenie wszystkich procesów uruchomionych w klastrze: procesy uruchomione lokalnie i zdalnie, gdziekolwiek znajduje się ich węzeł początkowy. Polecenie top wyświetla globalną statystykę pamięci, jak również obciążenie procesorów w klastrze. Zachowana jest unikalna przestrzeń identyfikatorów procesów w obrębie całego klastra, jak również jednolita lista procesów.

	Kerrighed	**openMosix**	**OpenSSI**
Zakres przestrzeni nazw	Tak	Niekompletne	Tak
Jednolita lista procesów (ang. *global ps*)	Tak	Niekompletne	Tak
Globalna statystyka pamięci i obciążenie procesorów (ang. *global top*)	Tak	Nie	Nie
Globalny dostęp do urządzeń (ang. *global /dev*)	Nie	Nie	Tak
Jednolita przestrzeń nazw w systemie plików	Tak	Niekompletne	Tak

Rysunek 1: Jednolity obraz systemu.

5.2 Globalne zarządzanie procesami

OpenMosix pozwala na dynamiczne równoważenie obciążenia procesorów w klastrze. Wszystkie procesy, z wyjątkiem mocno związanych z węzłem i używających segmentów pamięci Systemu V, mogą migrować. Nie ma zapewnionej obsługi migracji pojedynczych wątków.

OpenSSI również pozwala na dynamiczne równoważenie obciążenia procesorów w klastrze. Wszystkie procesy, z wyjątkiem tych, które są mocno związane z węzłem, mogą migrować. Procesy używające segmentów pamięci Systemu V i grupy wątków również mogą migrować. Natomiast pojedyncze wątki nie mogą.

Kerrighed pozwala na dynamiczne równoważenie obciążenia procesorów w klastrze. Wszystkie procesy, poza mocno związanymi z węzłem, pojedyncze wątki i grupy wątków mogą migrować.

	Kerrighed	**openMosix**	**OpenSSI**
Migracja procesów	Tak	Tak	Tak
Migracja wątków	Tak	Nie	Nie
Migracja aplikacji wielowątkowych	Tak	Nie	Tak
Globalne szeregowanie procesów	Tak	Tak	Tak
Konfigurowalność globalnego szeregowania procesów (ang. *global top*)	Tak	Nie	Nie

Rysunek 2: Globalne zarządzanie procesami.

5.3 Globalna komunikacja międzyprocesorowa

OpenMosix wspiera migrację procesów używających gniazd, łącz i semaforów z Systemu V. Nie ma wsparcia dla migracji procesów używających segmentów pamięci z Systemu V.

OpenSSI wspiera migrację procesów używających gniazd, łącz, semaforów z Systemu V i segmentów pamięci Systemu V. Kerrighed wspiera migrację procesów używających gniazd, łącz i segmentów pamięci Systemu V. Nie ma wsparcia dla migracji procesów używających semaforów Systemu V.

	Kerrighed	**openMosix**	**OpenSSI**
Migracja procesów używając segmentów pamięci Systemu V	Tak	Nie	Tak
Migracja procesów używając semaforów Systemu V	Nie	Tak	Tak
Migracja procesów używając łączy	Tak	Tak	Tak
Migracja procesów używając gniazd UNIX	Tak	Tak	Tak
Migracja procesów używając gniazd INET	Tak	Tak	Tak

Rysunek 3: Globalna komunikacja międzyprocesorowa.

5.4 Odporność na uszkodzenia i mechanizm migawki

W systemie OpenMosix istnieje możliwość dodania lub usunięcia węzła podczas pracy klastra. Kiedy węzeł ulegnie uszkodzeniu, na pozostałe węzły nie ma to wpływu. Procesy, które zmierzały do uszkodzonego węzła zostały utracone. Procesy rozpoczynające się w uszkodzonym węźle (przed uszkodzeniem) i migrujące do innych zazwyczaj ulegną uszkodzeniu. Nie ma możliwości zrzucenia migawek.

W systemie OpenSSI również możliwe jest dodanie lub usunięcie węzła podczas pracy klastra. Uszkodzenie węzła nie ma wpływu na pozostałe węzły, z wyjątkiem sytuacji, gdy na uszkodzonym węźle mieścił się menadżer globalnego systemu plików. Procesy zmierzające do uszkodzonego węzła zostały utracone. Procesy rozpoczynające się w uszkodzonym węźle (przed uszkodzeniem) i migrujące do innych zazwyczaj ulegną uszkodzeniu. Nie ma możliwości zrzucenia migawek.

Kerrighed na razie nie oferuje dynamicznej rekonfiguracji. Uszkodzenie jednego węzła ma wpływ na pozostałe węzły i zazwyczaj prowadzi do zakleszczenia lub awarii. Procesy, które zmierzały w kierunku uszkodzonego węzła, jak również te, które w nim się rozpoczęły, zostają utracone. Jest obsługiwany mechanizm zrzucania migawek, który pozwala ponownie uruchomić proces po awarii węzła lub klastra.

	Kerrighed	**openMosix**	**OpenSSI**
Dynamiczne dodawanie węzła	Nie	Tak	Tak
Dynamiczne usunięcie węzła	Nie	Tak	Tak
Odporność na uszkodzenie węzła	Nie	Tak	Tak
Zrzucenie migawki procesu	Tak	Nie	Nie
Zrzucenie migawki aplikacji wielowątkowej	Nie	Nie	Nie
Zrzucenie migawki aplikacji równoległych	Nie	Nie	Nie

Rysunek 4: Odporność na uszkodzenia i mechanizm migawki.

5.5 Wsparcie sprzętowe

Procesory 64 bitowe są wspierane przez OpenMosix (Itanium) i OpenSSI (Alpha, Itanium). Wspierają one również tryb SMP na wieloprocesorowych maszynach. Kerrighed nie wspiera jeszcze architektury 64 bitowej, ani trybu SMP.

	Kerrighed	**openMosix**	**OpenSSI**
Procesory 64 bitowe	Nie	Tak	Tak
tryb SMP	Nie	Tak	Tak

Rysunek 5: Wsparcie sprzętowe.

5.6 Wnioski

OpenSSI jest najbardziej stabilnym systemem i oferuje najwięcej standardowych funkcji jednolitego obrazu systemu. OpenMosix, który jest prawdopodobnie najpopularniejszym systemem, oferuje bardzo dobry kompromis pod względem funkcjonalności i wydajności. Jednak jego stabilność nie jest na takim poziomie, jak w systemie OpenSSI. Kerrighed jest wciąż prototypem w fazie badań, mniej solidnym niż inne systemy. Nie pozwala na dynamiczne dodawanie i usuwanie węzłów. Uszkodzenie węzła często prowadzi do awarii całego klastra. Jednak oferuje on najlepsze wyniki w komunikacji międzyprocesorowej, wydajne współdzielenie pamięci w klastrze, pozwala na migrowanie pojedynczych wątków. Kerrighed jest wciąż w fazie rozwoju, więc prawdopodobnie problemy ze stabilnością zostaną rozwiązane w najbliższej przyszłości.

6 Rozwiązania wspomagające budowanie klastrów

Wraz z rozwojem klastrów i systemów klastrowych ciągle aktywnie powstają nowe rozwiązania, które wspomagają ich pracę, czyniąc ją coraz bardziej niezawodną i

wydajną. Zostaną tu przedstawione i krótko opisane projekty wspomagające budowę i działanie klastrów.

6.1 LVS (ang. *Linux Virtual Server*)

Istnieje wiele rozwiązań podnoszących wydajność poprzez rozkładanie obciążenia pomiędzy poszczególne klastry. Najbardziej popularnym jest Linux Virtual Server. Jest to projekt służący do stworzenia oprogramowania serwerowego do budowania klastra wydajnościowego. W jego skład wchodzi oprogramowanie oraz zestaw łat na jądro linuksa. Projekt ten jest rozwinięciem idei budowy klastra w oparciu o translację adresów sieciowych. Zawiera wyspecjalizowane algorytmy rozkładania obciążenia między poszczególne węzły i monitorowania stanu pracy węzłów. Klaster zbudowany jest z jednego serwera głównego (ang. *tzw. load balancer*), który jako jedyny jest "widziany" na zewnątrz sieci i pełni rolę nadrzędną nad pozostałymi węzłami klastra. Jego zadaniem jest przekazywanie żądań do kolejnych węzłów. Dzięki niemu pakiety są przekierowywane tak, żeby klient był przekonany, że łączy się bezpośrednio z wybranym serwerem. W rzeczywistości wygląda to w ten sposób, że każdy pakiet przechodzi przez ten właśnie serwer główny, który decyduje do jakiego rzeczywistego serwera go przekierować. Proces przekazywania zadań jest całkowicie przezroczysty dla klientów. Ważną rolą projektu jest monitorowanie stanu technicznego i obciążenia poszczególnych węzłów klastra oraz szybka reakcja na zachodzące zmiany.

6.2 DLM (ang. *Distributed Lock Manager*) - Rozproszone zarządzanie blokadami

DLM służy do utrzymywania aktualności i zgodności danych w pamięciach podręcznych (cache) poszczególnych procesorów. W momencie, gdy dwa węzły potrzebują dostępu do tych samych danych, ich odwołania muszą być zsynchronizowane i uszeregowane, w celu zachowania integralności danych. Właśnie tym zajmuje się tzw. Distributed Lock Manager. Działa on na poziomie wewnętrznych mechanizmów komunikacyjnych klastra. Jest udostępniany aplikacjom na poszczególnych węzłach jako serwis. Funkcjonowanie mechanizmu DLM obciąża klaster i może zmniejszać jego wydajność, szczególnie przy dużej liczbie węzłów. Wielodostęp węzłów do jednego zasobu zwiększa znacznie dostępność tego zasobu dla stacji roboczych w sieci, gdyż odwołania wysokopoziomowe nie zawsze muszą być kolejkowane.

6.3 LVM (ang. *Logical Volume Manager*) - Zarządca logicznych wolumenów

LVM jest to logiczna, abstrakcyjna płaszczyzna pomiędzy mediami fizycznymi, czyli dyskami, a systemem plików, który zawiera dane. Pozwala on na kontrolowanie dysków i ich partycji na wyższym poziomie abstrakcji niż urządzenia fizyczne. Zasada działania LVM polega na tym, że fizyczne dyski są dzielone na jednostki pamięci, które mogą pochodzić z tego samego lub z innego dysku w komputerze. Jednostki te można łączyć w tak zwane Logical Volumes. Ich pojemność przydziela się następnie poszczególnym partycjom. Do zadań zarządcy wolumenów należy: utrzymywanie informacji o zasobach, zarówno fizycznych (dyski, partycje), jak i logicznych (wirtualne wolumeny), zarządzanie konfiguracją systemu, monitorowanie zasobów w celu zmierzenia wydajności lub wykrycia potencjalnych awarii, udostępnianie zasobów

użytkownikowi, udostępnianie szeregu różnych funkcji administracyjnych (np. migracje, kopie, zmiany organizacji, zwiększanie pojemności wolumenów). Logical Volume Manager umożliwia zmianę wielkości partycji w trakcie pracy. Stosownie do zapotrzebowania na przestrzeń dyskową administrator może przydzielać partycjom więcej lub mniej pamięci fizycznej.

6.4 *Fake*

Fake służy do realizacji systemu niezawodnościowego. Jego działanie polega na tym, że zapasowy serwer monitoruje przez cały czas stan serwera głównego i przejmuje jego pracę w momencie wykrycia awarii serwera głównego. Takie rozwiązanie stosuje się przy budowie serwerów pocztowych i www.

6.5 *DRBD*

Projekt DRBD powstał w celu stworzenia systemu mirrorowania systemu plików. Umożliwia on replikację i synchronizację danych w sposób ciągły i automatyczny, tzn., że dane przesyłane są na drugi komputer w momencie ich nadejścia, a nie w założonych z góry odstępach czasowych. Pozwala to na wykonywanie kopii zapasowych w sposób natychmiastowy. Z utworzonej kopii można skorzystać od razu w wypadku awarii pierwszego komputera. Działanie DRBD funkcjonalnie podobne jest do działania macierzy dyskowej RAID-1, przy czym dyski rozproszone są na różnych maszynach.

6.6 *High - Availability Linux Project*

Projekt High - Availability Linux ma na celu stworzenie rozwiązania umożliwiającego wysoką dostępności dla systemów GNU/Linux w oparciu o klastering. Jest to projekt realizowany przez społeczność linuksową, wspierany przez takie firmy jak IBM, SGI, Intel, SuSE, Conectiva i inne. Projekt Linux-HA współpracuje także z projektem Linux Virtual Server. Działanie High Availability Linux Project polega na tym, że dodatkowy moduł, w wypadku awarii, przejmuje wszystkie otwarte połączenia. Jeśli podstawowy system przestaje pracować poprawnie obsługę wszystkich połączeń przejmuje zapasowy system bez potrzeby (ze strony użytkowników) ponownego nawiązywania połączenia czy też powtórnego uwierzytelniania. Użytkownicy nawet nie dostrzegają przełączenia obsługi do innego węzła. Co więcej wszystkie transakcje oraz transfery danych mogą odbywać się nieprzerwanie bez konieczności ich wznawiania. Konfiguracja sprzętowa i programowa obu węzłów nie musi być identyczna, choć im poszczególne węzły będą do siebie bardziej podobne, tym łatwiej będzie później takim klastrem zarządzać. Oczywiście możliwe jest stosowanie większej ilości węzłów, a także korzystanie z innego typu współdzielenia danych.

6.7 *NetFilter*

NetFilter jest odpowiedzialny za filtrowanie ruchu sieciowego. Filtrowanie pakietów w jądrze Linuxa ma długą historię. Istnieje tam kod, który potrafi śledzić i kontrolować pakiety jakie pojawiają się w jego zasięgu. Bardzo ważną cechą filtra pakietów jest możliwość dołączania dodatkowych modułów zwiększających jego możliwości. Jednym z nich jest *nth*. Moduł ten dodaje nowy test sprawdzający czy dany pakiet jest

n - tym pasującym do reguły. Umożliwia to w prosty sposób wykonanie rozkładania obciążenia dla połączeń wchodzących do serwera między komputery w klastrze. Netfilter jest ogólnym szkieletem, do którego można dodawać moduły, zmieniając tym samym jego właściwości i funkcjonalność. Na jego bazie zostało zbudowanych wiele modułów, które dodają nowe możliwości do zarządzania pakietami. Począwszy od podstawowego IP tables do bardziej złożonych próbujących np. zapobiec atakom DoS (Denial of Service). Zaletą tego rozwiązania jest łatwa implementacja nie wymagająca instalacji dodatkowego oprogramowania. Całe niezbędne oprogramowanie jest dostarczone wraz z każdą dystrybucją Linux'a. Rozwiązanie tego rodzaju nadaje się jedynie do rozkładania obciążeń w prostych zastosowaniach. Niestety to rozwiązanie nie jest niezawodne, z powodu braku zabezpieczenia przed awarią któregoś z węzłów. Rozwiązanie to nie posiada mechanizmów samodiagnozowania. Wadą jest brak narzędzi do monitorowania zajętości węzłów i tym samym wykrywania awarii. Nie ma również komunikacji między węzłami.

6.8 *Heartbeat*

Zapewnienie stałej komunikacji między węzłami umożliwia wykrycie awarii komputera wchodzącego w skład klastra. Połączenie gwarantujące komunikację nazywa się *heartbeat* czyli dosłownie „uderzenia serca" - węzły informują się nawzajem, że działają. Jeżeli któryś z węzłów przestaje odpowiadać, pozostałe przejmują jego funkcje do czasu usunięcia usterki. Pakiet *heartbeat* jest to publicznie dostępne oprogramowanie udostępniające podstawową funkcjonalność pozwalającą tworzyć i zarządzać klastrami wysokiej dostępności w systemie Linux. Do głównych zadań pakietu należy:

- monitorowanie dostępności węzłów klastra,
- uruchamianie i zatrzymywanie usług działających w trybie wysokiej dostępności,
- zarządzanie zasobami dzielonymi w obrębie całego klastra (np. adres IP, zasoby dyskowe).

Jeden z komputerów w klastrze pełni rolę węzła głównego serwując pewne usługi, drugi (węzeł zapasowy) oczekuje bezczynnie na awarię węzła głównego. Gdy do tego dojdzie przejmuje on jego funkcje, uruchamiając wszystkie te usługi, które udostępniał węzeł główny. Oprogramowanie hearbeat nie potrafi przenosić usług i aplikacji między węzłami wraz z ich stanem – wykrywa tylko awarię węzłów, przenosi zasoby na węzeł zapasowy i na nim od nowa uruchamia usługi zatrzymane w wyniku awarii. Nadaje się do prostych instalacji np. serwerów http.

7 Bezpieczeństwo

Dane przechowywane i przesyłane w sieci są narażone na wiele niebezpieczeństw. To samo dotyczy systemów klastrowych. Zadania, które należy wykonać w celu zabezpieczenia systemów klastrowych, to przede wszystkim ochrona zasobów danych przed zniszczeniem i zapewnienie bezpieczeństwa przed dostępem osób nieupoważnionych do tych danych. Poziom bezpieczeństwa zasobów sprzętowych i danych w klastrach zapewniany jest na poziomie sieci, systemów operacyjnych, aplikacji i usług oraz na poziomie proceduralnym (dodawanie nowych użytkowników, dołączanie nowych

klastrów). Przy tworzeniu systemów bezpieczeństwa wykorzystywane są zarówno gotowe narzędzia, jak i nowe, dedykowane mechanizmy. Systemy klastrowe posiadają mechanizmy, które pozwalają na zapewnienie poufności wykonywanych zadań. Stopień zaawansowania pracy tych mechanizmów jest zależny od potrzeb użytkowników systemu klastrowego. W systemach klastrów niezawodnościowych informacje są przechowywane w kilku niezależnych od siebie miejscach, a więc utrata, zniszczenie lub uszkodzenie informacji w jednym z miejsc nie powoduje jej całkowitej utraty. Nadmiarowość powoduje wzrost kosztów pracy klastrów, ale jednocześnie gwarantuje wysoki stopień bezpieczeństwa. W celu zablokowania dostępu do przetwarzanych danych przez osoby niepowołane, w systemach klastrowych istnieje możliwość zabezpieczenia kanałów transmisji pomiędzy poszczególnymi węzłami. Wiąże się to ze spowolnieniem działania klastra spowodowanym koniecznością szyfrowania komunikatów przepływających między węzłami klastra. Do szyfrowania używa się coraz dłuższych kluczy, co z kolei wydłuża cały proces.

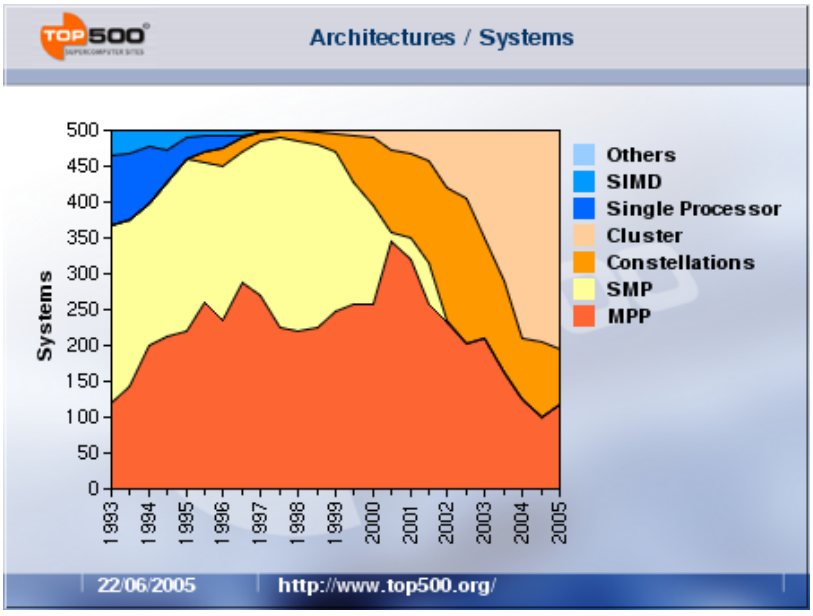
8 Podsumowanie

Łączenie w klastry stanowi alternatywę dla wieloprzetwarzania symetrycznego (SMP – symetric multiprocessing), jako technologia zapewniająca wysoką wydajność i dyspozycyjność. W związku z czym jest szczególnie interesującym rozwiązaniem dla aplikacji serwerowych. Duże systemy obliczeniowe dostępne na rynku są bardzo drogie. Zaletą klastrów jest wykorzystanie do ich konstrukcji ogólnie dostępnego i przez swoją popularność stosunkowo taniego sprzętu. Kolejnym czynnikiem, który wpływa na obniżenie kosztów budowy systemów klastrowych jest wzrost darmowego oprogramowania dostępnego w Internecie. Coraz częściej jest to oprogramowanie bardzo wysokiej jakości. Darmowe oprogramowanie jest elastyczniejsze, gdyż daje nam możliwość dostępu do kodu źródłowego systemu. Komputery klastrowe są skalowalne do bardzo dużych systemów. Można powiększyć liczbę komputerów w klastrze lub dołożyć kolejne procesory do istniejących już jednostek. Kolejną zaletą tworzenia klastra jest niskie ryzyko nieudanej inwestycji. Jeśli okaże się, że rozwiązanie nie nadaje się do realizacji postawionych celów, sprzęt można wykorzystać jako zwykłe samodzielne komputery.

Wszystkie te cechy przyczyniły się do spopularyzowania systemów klastrowych. Są one coraz częściej używane w zastosowaniach wymagających wysokiej wydajności. Na liście najszybszych komputerów świata (top 500) widzimy od kilku lat tendencję rosnącą. Dane pochodzące z listy umieszczone są na rysunku 6 i 7.

Rok	Miesiąc	Ilość klastrów
1996	Listopad	0
1997	Czerwiec	1
1997	Listopad	1
1998	Czerwiec	1
1998	Listopad	2
1999	Czerwiec	6
1999	Listopad	7
2000	Czerwiec	11
2000	Listopad	28
2001	Czerwiec	32
2001	Listopad	43
2002	Czerwiec	81
2002	Listopad	93
2003	Czerwiec	149
2003	Listopad	210
2004	Czerwiec	289
2004	Listopad	294
2005	Czerwiec	304

Rysunek 6: Ilość klastrów na liście top 500.



Rysunek 7: Klastry na liście top 500 (źródło[top]).

Literatura

[Beo] The beowulf cluster site - http://www.beowulf.org/.

[Cod] Coda file system - http://www.coda.cs.cmu.edu/.

[DRB] Drbd - http://www.drbd.org/.

[GFS] Red hat global file system - http://www.redhat.com/software/rha/gfs/.

[HAL] The high availability linux project - http://linux-ha.org/.

[Ker] Kerrighed - http://www.kerrighed.org/.

[LCI] The linux clustering information center - http://lcic.org.

[Lus] Lustre - http://www.lustre.org/.

[LVS] The linux virtual server project - http://www.linux-vs.org/.

[MOS] Mosix - cluster and grid management - http://www.mosix.org/.

[NET] Firewalling, nat and packet mangling for linux - http://www.netfilter.org/.

[NFS] Linux nfs - http://nfs.sourceforge.net/.

[OGF] The opengfs project - http://opengfs.sourceforge.net/.

[Opea] The openmosix project - http://openmosix.sourceforge.net/.

[Opeb] Single system image clusters (ssi) for linux - http://www.openssi.org/.

[RL04] Geoffroy Vallee Christine Morin Renaud Lottiaux, Pascal Gallard. Openmosix, openssi and kerrighed: A comparative study. Technical report, 2004.

[top] Lista najszybszych komputerów - http://www.top500.org/.