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

Projekt z przedmiotu RSO

Ewa Chachulska

17 czerwca 2005

Spis treści

1 Wstęp

Mimo wzrostu wydajności współczesnych komputerów PC, są zadania, które zawsze będą przekraczały możliwości pojedynczej maszyny. Dotyczy to wszelkich zadań obliczeniowych, począwszy od obliczeń matematycznych, kompilacji dużych programów, przez rendering grafiki, a na kompresji plików do formatu MP3 kończąc. Dotyczy to również wszelkich zadań przechowywania lub udostępniania dużej ilości danych. Do takich zadań tworzy się tzw. klastry - połączone grupy komputerów, pracujące pod nadzorem specjalnego oprogramowania, które potrafi rozdzielać pracę między wszystkie pracujące maszyny. Ponieważ uruchamianie klastrów wymagało dotąd zakupu specjalnego, kosztownego sprzętu i oprogramowania, ich budowa była zarezerwowana jedynie dla dużych ośrodków naukowych, firm i administracji państwowej. Obecnie, dzięki Linuksowi i bezpłatnemu oprogramowaniu Open Source klastry wykorzystywane są nie tylko w dużych korporacjach, ale także w małych prywatnych rozwiązaniach.

2 Typy klastrów

Jak myślimy o klastrach to pierwszym, co przychodzi do głowy to takie rozwiązania, które mają na celu podnieść wydajność obliczeniową jakiegoś systemu. Takie klastry nazywane są klastrami wydajnościowymi, (high-performance) i są stosowane wszędzie tam gdzie zależy nam na wykonaniu jakiś obliczeń w rozsądnym czasie. Poza rozwiązaniami podnoszącymi wydajność obliczeniową istnieją również systemy zapewniające niezawodną i nieprzerwaną pracę. Klastry wysokiej dostępności (high-availability) są stosowane tam, gdzie do czynienia mamy z aplikacjami, których nieprzerwana praca jest krytyczna dla jakiejś in-

stytucji czy systemu. Kluczem tego typu rozwiązań jest nadmiarowość. Klaster wysokiej dostępności zbudowany jest z wielu maszyn, a ich podzbiory mogą zapewnić określoną usługę. W najprostszym przypadku to jedno urządzenie lub serwer jest dostępny, pozostałe natomiast monitorują jego pracę, aby upewnić się, że wszystko jest poprawnie. Jeśli stwierdzone zostaną błędy to miejsce pierwszego serwera zajmuje jego następca. Istnieje jeszcze jeden typ klastrów. Koncepcja klastrów z równoważeniem obciążenia (load-balancing) jest taka by zapewnić wyższą wydajność dzięki rozproszeniu pracy na wiele komputerów. Na przykład, jeśli serwer WWW jest implementowany z użyciem klastra LB wówczas różne zapytania do serwera są dystrybuowane pomiędzy komputery w klastrze. Może to być realizowane przy użyciu prostego algorytmy Round-Robin. Przykładowo, gdy kierowane jest zapisanie do serwera DNS, ten zwraca adres następnej maszyny w klastrze. Oczywiście istnieją bardziej wyrafinowane algorytmy wykorzystujące sprzężenie zwrotne od każdej z maszyn tak by wybrać maszynę, która najlepiej poradzi sobie z zadaniem.

W tej chwili wszystkie przedstawione wyżej koncepcje przeplatają się tak by stworzyć rozwiązania spełniające konkretne wymagania.

3 Istniejące rozwiązania

W dalszej części zostaną przedstawione istniejące rozwiązania klastrowe. Są to rozwiązania zarówno zwiększające niezawodność jak i takie, których zadaniem jest zwiększenie wydajności.

3.1 BEOWULF

Beowulf to wielokomputerowa architektura, która może zostać użyta do obliczeń równoległych. Jest to system, który na ogół składa się z jednego węzła-serwera i jednego lub więcej węzła-klienta połączonego przez Ethernet lub jakąś inną sieć. Jest to system zbudowany w oparciu o powszechnie dostępne komponenty

sprzętowe, jak każdy PC zdolny do uruchomienia Linuxa oraz standardowe karty Ethernet i switch'e. Nie zawiera żadnych unikalnych komponentów sprzętowych i jest łatwy w tworzeniu. Beowulf korzysta również ze zwykłego oprogramowania, jak system operacyjny Linux, Parallel Virtual Machine (PVM) oraz Message Passing Interface (MPI). Węzeł-serwer kontroluje cały klaster i udostępnia pliki klientom. Pełni on także funkcję konsoli klastra i jest jego bramą do zewnętrznego świata. Duże maszyny Beowulf mogą posiadać więcej niż jeden węzeł-serwer, oraz inne węzły stworzone dla specyficznych zadań, na przykład konsole lub stacje monitorujące. W większości przypadków węzły-klienci w systemie Beowulf są głupie, im głupsze tym lepiej. Węzły są konfigurowane i kontrolowane przez węzeł-serwer, i robią tylko to, co kazano im robić. W konfiguracji bezdyskowej klienci nie znają nawet swojego adresu IP lub nazwy, dopóki serwer im ich nie poda. Jedną z podstawowych różnic pomiędzy Beowulf'em a architekturą COW (Cluster of Workstations) jest to, że Beowulf zachowuje się bardziej jak jedna maszyna, niż wiele stacji roboczych. W większości przypadków węzły-klienci nie posiadają klawiatur czy monitorów, a dostęp do nich jest możliwy jedynie przez odległe logowanie bądź opcjonalny terminal szeregowy. Węzły Beowulf można sobie wyobrazić jako pakiet CPU + pamięć, który może zostać podłączony do klastra, tak jak CPU czy moduł pamięci może zostać dołączony do płyty głównej. Beowulf nie jest specjalnym pakietem oprogramowania, nową topologię sieci czy nową nakładkę na jądro. Beowulf to technologia łączenia komputerów Linux, aby utworzyć równoległy, wirtualny superkomputer. Chociaż istnieje wiele pakietów oprogramowania, takich jak modyfikacje jądra, narzędzia konfiguracyjne, które przyspieszają architekturę Beowulf, ułatwiają konfigurację i zwiększają użyteczność, jednak możliwe jest zbudowanie maszyny Beowulf wykorzystując standardową dystrybucję Linux, bez żadnego dodatkowego oprogramowania.

Klasyfikacja

Systemy Beowulf są konstruowane z różnorodnych części. W celu zwiększenia możliwości obliczeniowych czasami korzysta się z pewnych niedostępnych powszechnie komponentów. W celu odróżnienia osiągów różnych typów systemów, i ułatwienia dyskusji na ich temat, proponuje się następującą klasyfikację:

Beowulf klasy I:
Maszyna tej klasy jest zbudowana wyłącznie z powszechnie dostępnych komponentów. W celu sprawdzenia powszechności elementów przeprowadza się test przy użyciu "Computer Shopper" (miesięcznika/katalogu systemów PC i komponentów). Test ten wygląda następująco:
Beowulf KLASY I to maszyna, która może zostać skonstruowana z części znalezionych przynajmniej w 3 krajowych/ogólnoświatowych katalogach reklamowych. Zalety systemów KLASY I to:

- sprzęt dostępny z wielu źródeł (niskie ceny, łatwa konserwacja)
- niezależność od konkretnego producenta
- wsparcie standardowych sterowników Linux
- najczęściej oparte o standardy (SCSI, Ethernet itp.)

Wady systemów KLASY I to:

- najlepsza wydajność może wymagać sprzętu KLASY II

Beowulf klasy II
Beowulf KLASY II to po prostu każda maszyna, która nie przejdzie testu z użyciem "Computer Shopper". To nie jest coś złego, jest to jedynie klasyfikacja maszyny.
Zalety systemów KLASY II to:

- całkiem dobra wydajność!

Wady systemów KLASY II to:

- zależność od konkretnego producenta
- może być droższy niż system klasy I
- problemy ze sterownikami

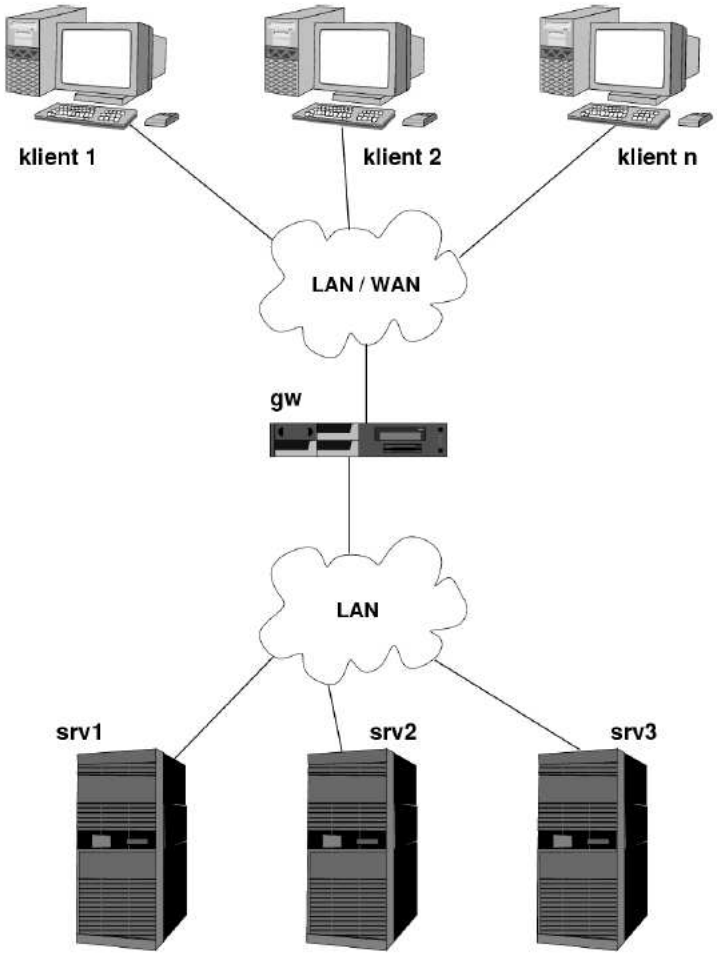
Żadna KLASA nie jest lepsza niż inna. Wszystko zależy wyłącznie od potrzeb i możliwości finansowych. Ta klasyfikacja została utworzona jedynie w celu ułatwienia i większej zwięzłości dyskusji na temat systemów Beowulf.

3.2 Linux Virtual Server

Misja projektu LVS jest stworzenie serwera Linux o wysokiej wydajności i dostępności, w oparciu o clustering zapewniający dobrą skalowalność i niezawodność. Projekt ten kładzie nacisk przede wszystkim na wysoka wydajność poprzez load-balancing, ale współpracuje także ściśle z projektem Linux-HA i uwzględnia w proponowanych rozwiązaniach kwestie wysokiej dostępności. Load-balancing można uzyskać na dwóch poziomach: aplikacji i IP. Programy takie jak, Reverse-proxy lub pWEB rozwiązują rozkładanie obciążenia na poziomie protokołu HTTP, przekazując poszczególne zadania do jednego z rzeczywistych serwerów. Przy dużej ilości węzłów może to jednak doprowadzić do sytuacji gdy sam load-balancer stanie się wąskim gardłem z powodu narzutu protokołu HTTP. Dlatego LVS oparty jest o load-balancing na poziomie protokołu IP, co umożliwia budowanie klastrów o liczbie węzłów dochodzących do kilkudziesięciu. LVS stoi za takimi adresami jak linux.com, sourceforge.net, themes.org, www.zope.org,www.real.com i wieloma innymi.

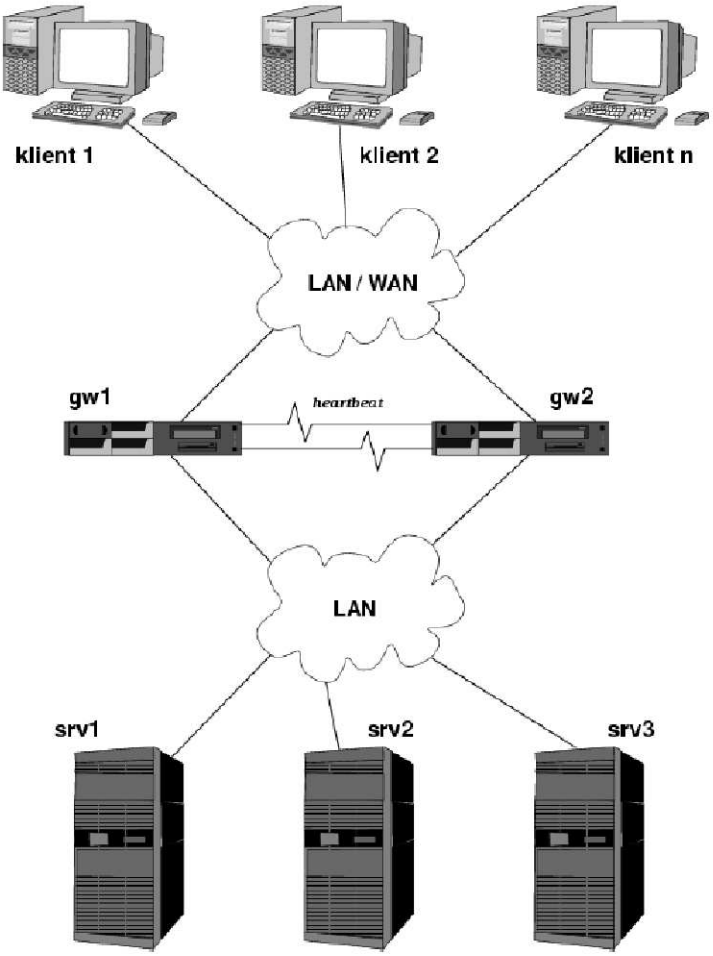
Proponowane rozwiązania

Rysunek przedstawia typowy system LVS, który zbudowany jest przede wszystkim w celu zwiększenia wydajności.



Maszyny srv1 do srv3 noszą w nomenklaturze LVS nazwę serwerów rzeczywistych. To na nich uruchomione są usługi. Mogą one w ogólności pracować pod

dowolnym systemem operacyjnym i z dowolnym oprogramowaniem serwerów usług. Maszyna gw nosi w nomenklaturze LVS nazwę “directora“, ponieważ kieruje zadania klientów do wybranych serwerów rzeczywistych, realizując w ten sposób rozkładanie obciążenia na poszczególne maszyny. Oprogramowanie LVS oferuje różne algorytmy wyboru serwera wirtualnego, do którego zostanie przekazane zadanie. Cały klaster dzięki directorowi widziany jest w konsekwencji jako “serwer wirtualny“. Jak wspomniano wcześniej, przekazywanie zadan wykonywane jest na poziomie protokołu IP w oparciu o NAT, IP tunneling lub direct routing. Oferowana “przy okazji“ wysoka dostępność jest niewiele większa niż na przykład ta, jaka oferuje round-robin DNS, ponieważ w przypadku uszkodzenia serwera rzeczywistego kierowane do niego zadania nie będą obsługiwane. Jednak nawet w takim przypadku posiadana redundancja zapewnia obsługę przynajmniej części klientów (tych, których zadania zostały przekazane do sprawnych serwerów wirtualnych). Stosując jednak odpowiednie oprogramowanie, kontrolujące dostępność poszczególnych serwerów wirtualnych, można w bardzo prosty sposób uzyskać całkiem niezła dostępność. Drugim mankamentem przedstawionego wyżej rozwiązania jest to, ze sam director stanowi SPOF (Single Point Of Failure), ponieważ w przypadku jego awarii żaden z serwerów rzeczywistych, a w konsekwencji cały klaster, będzie niedostępny. Można jednak w łatwy sposób zapewnić wysoka dostępność wyżej pokazanego systemu, stosując redundantny director, jak przedstawiono na rysunku, oraz kilka narzędzi opisanych niżej.



Wyżej przedstawiona konfiguracje można w łatwy sposób uzyskać stosując na przykład program heartbeat. LVS dostarcza kilka elementów programowych, które w połączeniu umożliwiają zbudowanie wysoko-dostepnego systemu zapewniającego jednocześnie wysoka wydajność. Instaluje się je na maszynie (maszynach) działającej jako director, pracującej pod kontrola systemu GNU/Linux.

Moduły jądra

Umożliwiają one wydajny, load-balancing, bo implementowany jest on na poziomie jądra:

- łata virtual server dla jadra 2.2
- moduł netfilter IPVS dla jądra 2.4
- moduł netfilter IPVS dla jądra 2.5 (eksperymentalny)

ipvsadm - Jest to interfejs użytkownika do LVS. Został zaimplementowany jako narzędzie linii poleceń służące do konfiguracji struktur danych jądra uzupełnionego o wyżej wymieniona łatę lub moduł. Używa się go podobnie do iptables, ipchains czy ipfwadm.

Mon - Jest to program przeznaczony do monitorowania usług. W łatwy sposób można skonfigurować go tak, aby w razie uszkodzenia któregoś z serwerów rzeczywistych, serwer ten został usunięty z puli dostępnych maszyn. Dzięki temu można uzyskać failover serwerów rzeczywistych realizowany przez "trzecia" maszynę - director.

Heartbeat - Jest to program świadczący usługi uczestnictwa i komunikacji w klastrach wysokiej dostępności. Można go użyć w celu zapewnienia przejęcia kontroli nad LVS przez zapasowy director w razie awarii aktywnego directora.

ldirectord - Linux Director Daemon Jest to program przeznaczony do monitorowania działania usług na rzeczywistych serwerach. Obecnie jest zaimplementowane monitorowanie usług HTTPS i HTTPS. Spełnia on podobna funkcje jak "mon", ale został napisany specjalnie do użytku z LVS - na przykład automatycznie czyta odpowiednie pliki konfiguracyjne LVS i samodzielnie dokonuje odpowiednich modyfikacji w tablicach routingu LVS w przypadku "padania" i "powstawania" poszczególnych serwerów rzeczywistych. W przypadku uszkodzenia wszystkich serwerów rzeczywistych możliwe jest automatyczne ustawienie routingu na

serwer “fall-back“, który może na przykład zwracać tylko komunikat o tymczasowej niedostępności danej usługi. Ponadto bardzo dobrze integruje się on z programem heartbeat. directord jest dystrybuowany razem z LVS.

keepalived Jest to projekt którego głównym celem jest dodanie sprawnego mechanizmu monitorowania serwerów rzeczywistych. Keepalived jest napisany w C i potrafi monitorować serwery rzeczywiste wchodzące w skład systemu LVS jednocześnie w trzech warstwach modelu OSI/ISO: trzeciej, czwartej i piątej. W przypadku “padu“ którejś z usług keepalived informuje o tym fakcie jądro wykonując odpowiednie wywołanie setsockopt, dzięki czemu uszkodzony serwer zostaje usunięty z topologii LVS. Keepalived zawiera tez stos VRRPv2, który umożliwia monitorowanie stanu samego directora i ewentualny jego failover. Jest wiec on w stanie zastąpić zarówno heartbeat jak i mon, czy ldirectord.

fake Fake (ang. nieprawdziwy, podrabiany) służy do odbierania uszkodzonemu serwerowi adresu IP, dzięki czemu zapasowy serwer może udostępniać usługi. Jest to przydatne zwłaszcza, gdy uszkodzenie wystąpi w taki sposób, ze maszyna i system operacyjny nadal będą pracować (zajmując adres IP), ale dana usługa będzie niedostępną. W tym celu fake konfiguruje odpowiedni interfejs z danym adresem IP (wykorzystując IP aliasing), a następnie podszywa się pod dana maszynę wykorzystując ARP spoofing.

3.3 Klaster OpenMosix

Klaster OpenMosix pozwala utworzyć superkomputer z rozproszonych zasobów wielu stacji roboczych i serwerów, dzięki czemu otrzymujemy moc przetwarzania dostępną zazwyczaj na dużych wieloprocesorowych maszynach pracujących w technologii SMP. Podstawowa różnica do SMP polega na szybkości wymiany danych między jednostkami obliczeniowymi (w klastrze ta prędkość ogranicza

się do prędkości sieci typu LAN).

OpenMosix dynamicznie rozkłada obciążenie, przesuwając je na mniej obciążone maszyny - migracja ta z perspektywy użytkownika jest całkowicie przezroczysta, a dzielenie obciążenia pomiędzy poszczególne węzły klastra odbywa się na poziomie jądra systemu. Architektura tego klastra wykorzystuje technologię SSI (Single System Image), w której skład wchodzą: ARS (Adaptive Resource Sharing), PPM (preemptive proces migration), oraz klastrowy system plików wykorzystujący DFSA. Dzięki tym mechanizmom nie jest potrzebne specjalnie kodowanie aplikacji dla środowiska klastrowego a OpenMosix dokona wyboru najlepszej maszyny dla wykonania danego zadania wykorzystują swoje mechanizmy równoważenia obciążenia. OpenMosix potrafi dokonać migracji większości procesów w systemie Linux. Na przykład jeśli aplikacja używa mechanizmu fork() do uruchomienia procesów potomnych, procesy te mogą zostać przesunięte na inne węzły klastra w sposób niezauważalny dla użytkownika. Dla potrzeb identyfikacji procesów każdy z nich skojarzony jest z parametrem UHN (Unique Home Node) przydzielany w momencie uruchamiania procesu. UHN to zazwyczaj węzeł na którym powstał proces. UHN identyfikuje proces nawet po jego migracji do innego węzła. Pozwala to w openMosixie na zastosowanie mechanizmu CC w którym zasoby korzystają z zasobów lokalnego węzła w klastrze ale są identyfikowane i komunikują się z użytkownikiem.
Tym co zdecydowanie utrudnia taką migrację procesów to:

- bezpośrednia manipulacja urządzeniami wejścia/wyjęcia
- dzielona pamięć do zapisu
- planowanie w czasie rzeczywistym

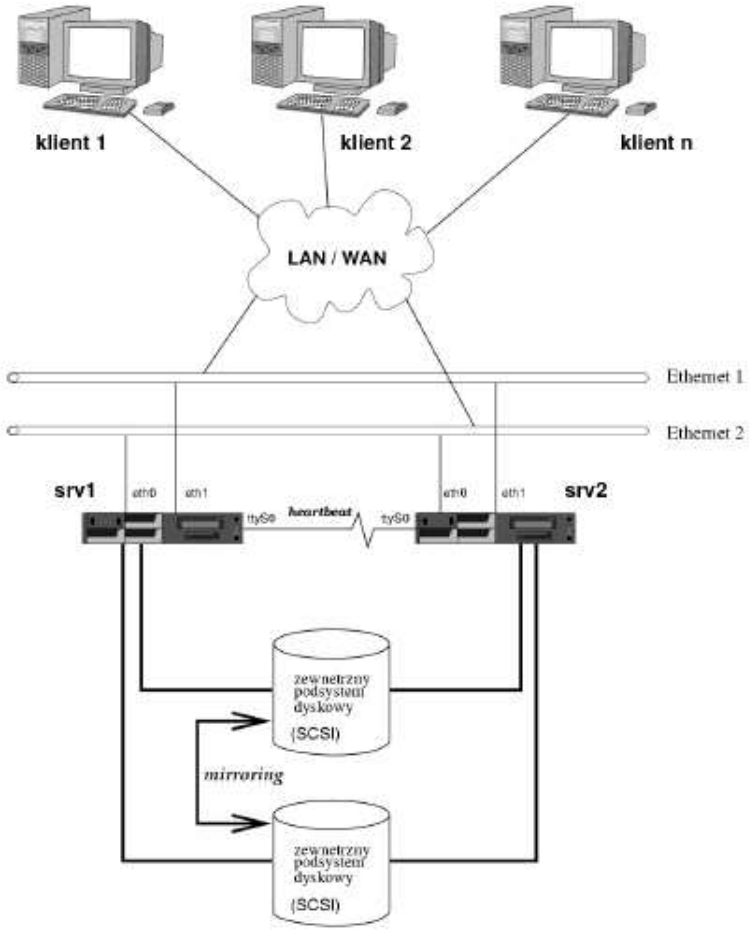
Wszystkie procesy które korzystają z takich funkcji nie mają możliwości migrować i zostają zamknięte w środowisku swojego UHN.

3.4 Linux HA- High Availability Linux Project

Projekt ten ma na celu stworzenie rozwiązania wysokiej dostępności dla systemów GNU/Linux w oparciu o clustering. Podstawowym mechanizmem projektu Linux-HA jest program heartbeat, realizujący usługi uczestnictwa i komunikacji. Procesy heartbeat działające na serwerach wymieniają miedzy sobą komunikaty świadczące o działaniu każdego z węzłów.
Komunikaty te heartbeat potrafi transmitować przez następujące media:

- UDP (z wykorzystaniem multicastu, co ma znaczenie zwłaszcza w przypadku większych klastrów)
- linie szeregowa “null-modem“
- połączenie PPP
- magistrale SCSI

Choć dzięki wykorzystaniu protokołu UDP poszczególne węzły klastra mogą teoretycznie być umieszczone w różnych podsieciach i nie mieć fizycznego połączenia w postaci kabla szeregowego, autorzy heartbeat jednak bardzo zalecają użycie kabla null-modem. Ze względu na nieskomplikowana budowę, stanowi on praktycznie niezawodny kanał komunikacyjny dla sygnału. Ponieważ heartbeat umożliwia kontrolowanie “pulsu“ węzłów przez kilka mediów jednocześnie, bardzo zalecane jest wykorzystanie do tego celu wszystkich dostępnych niezależnych mediów łączących poszczególne węzły, gdyż zapewni to maksymalna pewność, ze węzły nie utracą ze sobą kontaktu.



W przypadku przedstawionym na powyższym rysunku można skonfigurować programy heartbeat działające na obu węzłach, by wymieniały komunikaty przez obie sieci Ethernet, połączenie null-modem, a także przez interfejs SCSI.

- Przejmowanie usług

Interfejsy eth0 (podstawowy) i eth1 (zapasowy) na obu węzłach są przeznaczone do świadczenia usług przez klaster, oraz do administracji. Załóżmy, ze w powyższej sytuacji mamy zamiar skonfigurować usługi WWW (z wykorzystaniem serwera Apache) i SMB (z wykorzystaniem pakietu Samba) które powinny być widoczne pod adresem IP 1.2.3.4. W takim przypadku konfiguracja klastra z wykorzystaniem programu heartbeat przebiega następująco:

1. instalujemy na obu węzłach wybrana dystrybucje Linuksa

2. konfigurujemy na obu węzłach interfejsy sieciowe (każdy powinien mieć swój unikalny adres - adres “usługowy“ 1.2.3.4 będzie konfigurowany automatycznie przez heartbeat). Konfigurujemy również połączenie szeregowe.

3. konfigurujemy podsystem przechowywania danych. Nie każemy jej jednak automatycznie montować przy starcie systemu.

4. konfigurujemy na obu węzłach serwery Apache i Samba w taki sposób, aby były gotowe do świadczenia usług na adresie IP 1.2.3.4, jednak w taki sposób, aby nie były one uruchamiane automatycznie przy starcie węzła

5. instalujemy na obu węzłach heartbeat i konfigurujemy go tak, aby sprawował kontrole nad usługami “apache“ i “samba“ pod adresem IP 1.2.3.4, oraz montował w odpowiednim momencie system plików. Polega to na wpisaniu dosłownie jednej linijki, takiej jak: srv1 1.2.3.4 Filesystem::/dev/sda1::/data::ext2 apache samba do pliku haresources na każdym z węzłów.

Heartbeat będzie od tego momentu prowadził wymianę komunikatów “pulsu“ pomiędzy klastrami, wykrywał odejścia węzłów z klastra i automatycznie migrował adres IP serwisów oraz same usługi w odpowiedni sposób. Domyślnie pierwszy z włączonych węzłów przejmuje role aktywnego i heartbeat uruchamia usługi konfigurując na podstawowym interfejsie adres IP 1.2.3.4, oraz uruchamiając wymienione wyżej skrypty z parametrem start. Gdy do klastra dołącza kolejny węzeł, wykrywa on dzięki komunikatom “heartbeat“ przesyłanym przez łącze szeregowe oraz przez UDP (multicast), ze klaster świadczy już usługi i przechodzi w tryb “standby“. Węzły wymieniają komunikaty na temat zdarzeń dotyczących węzłów opuszczających klaster nieoczekiwanie lub wyłączanych administracyjnie oraz przyłączających się do niego, występujących błędów itp. Ze względów bezpieczeństwa, aby uniemożliwić wprowadzenie klastra w błąd,

wymiana komunikatów podlega autentykacji. W zależności od tego, jak bardzo narażona na atak jest siec przez która porozumiewają się węzły, można wybrać jeden z następujących mechanizmów autentykacji: crc, md5 oraz sha1. Tym samym heartbeat uruchomiony w odpowiedniej konfiguracji sprzętowej (chodzi zwłaszcza o redundantne połączenia, jak wyżej) implementuje dwa pierwsze podsystemy klastra wysokiej dostępności:

Usługi uczestnictwa (membership services)

- wykrywanie maszyn dołaczajacych do klastra
- wykrywanie maszyn odłączających się od klastra
- wykrywanie uszkodzonych połączeń
- wykrywanie naprawianych połączeń
- powiadamianie zainteresowanych

Usługi komunikacji (communication services)

- niezawodny multicast
- autentykacja i autoryzacja
- nadmiarowe łącza

3.5 Motorola High-Availability Linux 3.011

High-Availability Linux 3.011 to produkt, którym motorola chce rozpocząć nowa erę wysokiej dostępności - 6NINES (sześć dziewiątek), czyli 99,9999dostępności, co odpowiada trzydziestu sekundom planowanej i nieplanowanej niedostępności systemu. HA Linux to rozwiązanie, w którego skład wchodzi zarówno odpowiednie oprogramowanie, jak tez odpowiedni sprzęt. Obecnie Motorola

oferuje HA Linuksa w połączeniu z platforma MXP, a w przyszłości także platformami CXP i HXP.

Sprzęt

MXP Multi-Service Packet Transport Platform to specjalna “szafa“ dedykowana dla rozwiązań wysokiej dostępności, zawierająca 18 slotów połączonych magistrala PICMG 2.16 oraz siecią połączeń szeregowych o wysokiej przepustowości. Sloty umożliwiają montowanie na platformie serwery oparte na procesorach Intel oraz PowerPC.

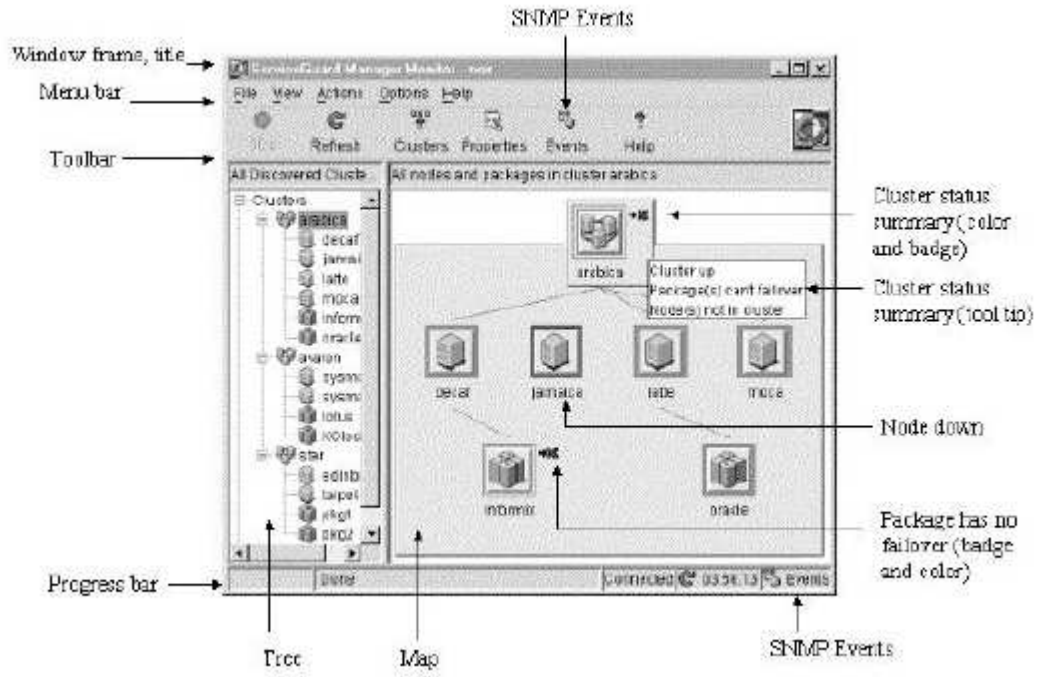
Oprogramowanie

Dwie podstawowe koncepcje na jakich zbudowano HA Linuksa to “Managed object model“ oraz “Distributed event management“. Pierwsza z nich polega na tym, ze każdy element systemu (chłodzenie, wentylator, procesor, system operacyjny itp.) jest traktowany jako obiekt. Z każdym obiektem jest związana nazwa, atrybuty oraz polityka obsługi (ang. policy). Gdy atrybut obiektu zmienia stan (pojawia się zdarzenie), powoduje to odpowiednia obsługę według polityki. Rezultatem może być przeniesienie usługi na zapasowy obiekt, wyłączenie obiektu lub jego restart. Jeśli na przykład w systemie MXP zepsują się trzy wentylatory, polityka obsługi takiego zdarzenia może nakazać ustawienie pozostałych dwóch na maksymalne obroty. Polityki obsługi to zwykłe pliki konfiguracyjne, które można w łatwy sposób modyfikować. Rozproszone zarządzanie zdarzeniami pozwala na przesyłanie zdarzeń przez siec i ich obsługę (na przykład przejecie usług) na zdalnych systemach. Distributed Inter-System Communication System (DISCS), to podsystem dostarczający środowiska niezawodnego transportu zdarzeń przez IP. W systemach MXP wykorzystuje on fizyczne połączenia

PICMG 2.16. Checkpoint services umożliwia znaczne przyspieszenie czasu przełączeń (failover). W systemie typu 6NINES niedopuszczalna jest taka sytuacja, ze w przypadku awarii węzła węzeł zapasowy jest inicjalizowany, a następnie przejmuje role uszkodzonego. Dlatego stworzono mechanizm polegający na częstej wymianie stanu (checkpointing) miedzy węzłem aktywnym a zapasowymi. Dzięki temu węzeł zapasowy może zacząć prace od razu. Sterowniki sprzętu w HA Linuksie pozwalają na efektywna diagnostykę stanu urządzeń i szybkie zgłaszanie ewentualnych awarii do managera zdarzeń, który następnie podejmuje odpowiednie działanie. Autorzy systemu twierdza, ze HA Linux pozwoli na osiągnięcie czasów przełączeń (w wypadku uszkodzenia systemu) rzędu 1-2 sekund, co rzeczywiście umożliwiłoby osiągnięcie dostępności rzędu 6NINES.

3.6 Hewlett-Packard

HP kontynuując swoja strategie rozwijania możliwości Linuksa w dziedzinie wysokiej niezawodności przeniosło z HP-UX-a oprogramowanie MC/Serviceguard.



Zdjecie ekranu ServiceGuard manager HP Multi-Computer/Serviceguard MC/ Serviceguard zapewnia wysoka dostępność systemu, a ponadto potrafi równoważyć obciążenie na poszczególne węzły. Współpracuje z HP StorageWorks Cluster Extension, co pozwala na geograficzne rozproszenie części składowych klastra. Razem z pakietem /rozprowadzane są przykładowe skrypty ułatwiające uruchomienie na klastrze serwerów Apache, NFS, Samba oraz SendMail. Zawiera narzędzie HP serviceguard manager, które jest graficznym interfejsem użytkownika umożliwiającym zarządzanie klastrem oraz monitorowanie jego stanu. Potrafi on obsłużyć serviceguard'a działającego zarówno pod Linuksem jak i w systemie HP-UX. Przykładowy ekran tego interfejsu jest przedstawiony na rysunku. HP MC/ServiceGuard działa na serwerach HP ProLiant DL380 G2 oraz DL580 G2, z dystrybucja Red Hat Professional 7.3 (w przypadku interfejsu SCSI) oraz Red Hat Advanced Server 2.1 (w przypadku użycia Fibre Channel).

3.7 Ultra Monkey

Projekt Ultra Monkey ma na celu stworzenie serwera wysokiej dostępności i wykorzystującego load-balancing, w oparciu o komponenty Open Source działające pod systemem Linux. W tym momencie uwaga jest skoncentrowana na skalowanych "web farms" wysokiej dostępności, choć można tego rozwiązania używać także do takich usług jak e-mail i FTP. Sercem Ultra monkey jest LVS , a failover zapewnia pakiet heartbeat. Usługi są monitorowane przy pomocy daemona ldirectord. W obecnym stadium rozwoju Ultra Monkey obsługuje następujące usługi:

- HTTP
- HTTPS
- FTP
- POP3
- IMAP
- SMTP
- LDAP

W skład pakietu wchodzą przykładowe, przetestowane konfiguracje gotowych topologii. Można dzięki nim szybko zaimplementować swój system dostosowując przykłady do swoich potrzeb.

3.8 DRBD

DRBD to moduł jądra służący do mirrorowania systemów plików przez siec. Działanie DRBD polega na tym, ze każdy blok danych zapisywany lokalnie

na dysku jest także przesyłany do zdalnego węzła, na którym również zostaje zapisany. Odczyty są zawsze wykonywane lokalnie. Jest wiec to mechanizm ciągłej replikacji danych. W chwili obecnej tylko jeden węzeł w danej chwili może mieć dane urządzenie zamontowane do zapisu i odczytu, choć możliwe byłoby rozwiniecie tego oprogramowania tak, aby więcej niż jeden węzeł miał taki dostęp. Na DRBD zrealizowano całkiem poważne systemy produkcyjne, w tym dwuterabajtowy system plików na uniwersytecie Utah, czy największe centrum danych w Japonii (ponad 4000 komputerów).

4 Bibliografia

- “Zaawansowane programowanie w systemie Linux” *Neil Mathew, Richard Stones*
- “ Klastry OpenMosix“ *Linux Magazine*
- “Linux Clustering. Building and Maintaining Linux Clusters“
- “High Performance Linux Clusters with OSCAR, Rocks, OpenMosix and MPI“
- “Linux Clustering with CSM and GPFS“
- http://linux-ha.org
- http://www.linuxvirtualserver.org
- http://www.ultramonkey.org
- http://www.mcg.motorola.com
- http://www.hp.com/products1/unix/highavailability