Wysoka dostępność w systemie Linux

Marcin Owsiany <marcin@owsiany.pl>

11 stycznia 2003 roku

Streszczenie

Referat ten przedstawia rozwiązania z dziedziny systemów wysokiej dostępności dostępne w systemie Linux. Główny nacisk położono na najpopularniejsze ze względów ekonomicznych klastry wysokiej dostępności.

Przedstawione zostały rodzaje rozwiązań proponowane przez projekty i firmy zajmujące się wysoką dostępnością systemów GNU/Linux, a także samo oprogramowanie oferowane przez nie.

Spis treści

1 Podstawowe pojęcia

Prawie wszystkie rozwiązania wysokiej dostępności oparte na systemach GNU/Linux to klastry wysokiej dostępności, zbudowane z kilku stosunkowo tanich maszyn.

W zasadzie jedynymi systemami, które są skonstruowane inaczej, są Advanced HA Motoroli (patrz rozdział 2.8.1) oraz Linux na systemach zSeries i S/390 firmy IBM (`http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp0220.html`).

W tym rozdziale znajdują się podstawowe definicje związane z klastrami wysokiej dostępności, które są wykorzystywane niżej.

1.1 Podsystemy w klastrze wysokiej dostępności

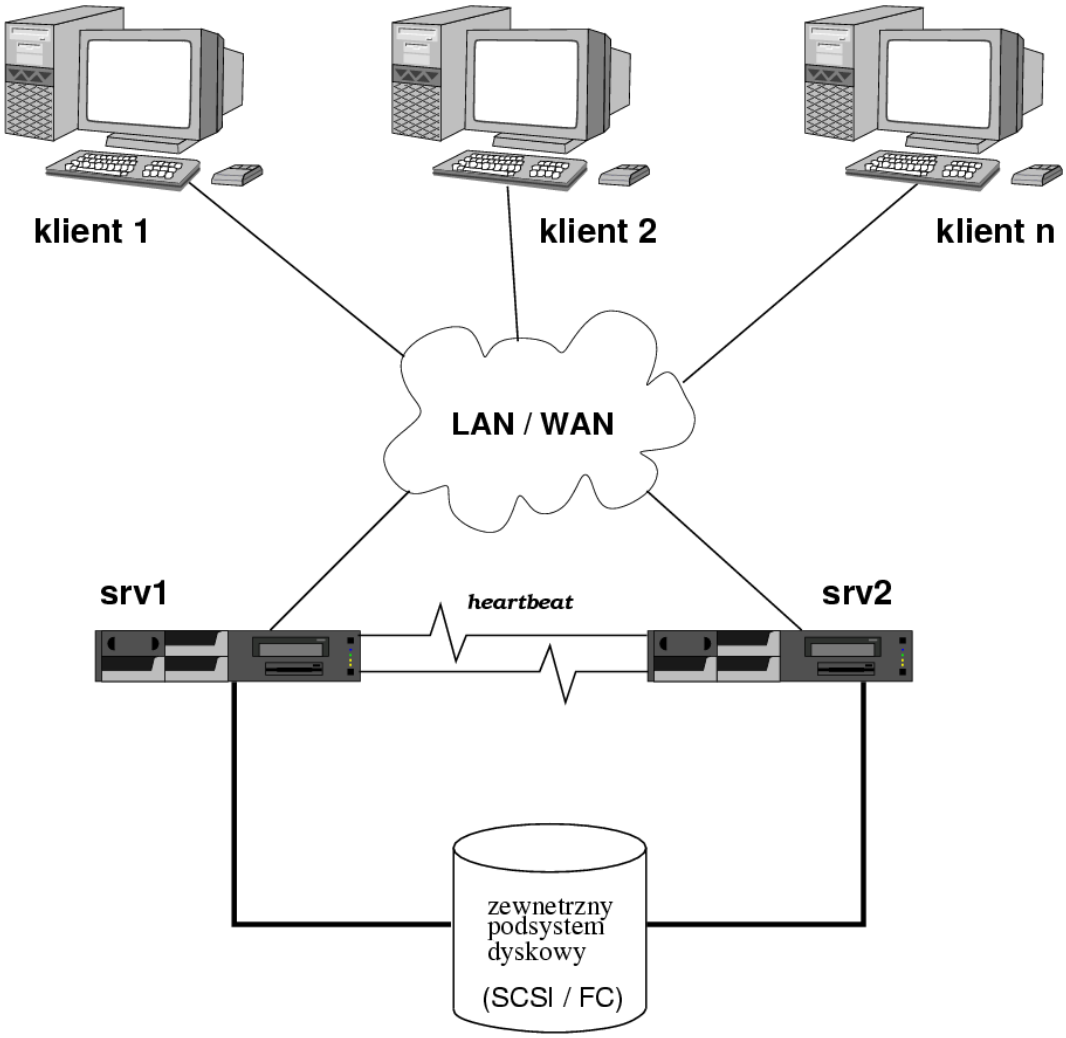
W klastrze takim można wyróżnić następujące podsystemy:

- Usługi uczestnictwa (ang. *membership services*)
- Usługi komunikacji (ang. *communication services*)
- Zarządzanie klastrem (ang. *cluster management*)
- Odgradzanie od zasobów (ang. *resource fencing*)
- Monitorowanie zasobów (ang *resource monitoring*)
- Współdzielenie/replikacja pamięci masowych

1.2 Działanie klastra — usługi uczestnictwa i komunikacji

Aby wyjaśnić role wyżej wymienionych podsystemów rozważmy prosty klaster wysokiej dostępności przedstawiony na rysunku 1.

Klaster składa się w naszym przypadku z dwóch węzłów, oraz zewnętrznej pamięci masowej podłączonej przez interfejs SCSI lub Fibre Channel. Udostępnia on przez sieć pewne usługi klientom (są one udostępnione pod pewnym adresem IP, będącym adresem klastra).



Rysunek 1: Schemat prostego klastra wysokiej dostępności

W klastrze tylko jeden z węzłów jest aktywny w danym momencie — to on ma zamontowany zewnętrzny system plików, na jego interfejsie sieciowym jest skonfigurowany adres IP klastra, i to na nim uruchomione są usługi. Pozostałe węzły (w tym przypadku jeden) pozostaje w trybie „standby".

Aby węzeł zapasowy mógł w przypadku awarii węzła aktywnego przejąć jego rolę, musi znać jego stan. W tym celu stosuje się usługi uczestnictwa i monitorowanie zasobów:

usługi uczestnictwa polegają na (najczęściej częstej, okresowej) wymianie komunikatów typu „I am alive", zwanej „pulsem" (ang. *heartbeat*). Oprócz tego węzły mogą wymieniać komunikaty informujące na przykład o administracyjnym wyłączeniu jednego z węzłów.

monitorowanie zasobów polega na okresowym sprawdzaniu dostępności usług na węźle aktywnym

W przypadku konieczności zmiany aktywnego węzła (przejęcie roli nosi nazwę „fail-over") czy to na skutek wydłużonego braku „odczuwalnego pulsu", czy zgłoszonej wprost rezygnacji z aktywności, jeden z węzłów zapasowych przejmuje adres IP klastra, uzyskuje dostęp do współdzielonych danych i uruchamia usługi.

1.3 Resource fencing

Z tym krokiem związane są pewne zagrożenia, które powodują, że klastry zawierające współdzielone elementy (zwłaszcza pamięci masowe) muszą przedsięwziąć środki ochrony integralności tych elementów na wypadek tymczasowej lub wyjątkowo „perfidnej" awarii któregoś z węzłów.

Technikę tą określa się mianem „resource fencing" (odgradzanie od zasobów).

Zasada ta dotyczy prawie wszystkich zasobów. Na przykład w przypadku adresów IP, gdy uszkodzony węzeł nie zwolni adresu, wystąpi konflikt i żaden z węzłów nie będzie mógł skorzystać z adresu.

Szczególne znaczenie ma to jednak w przypadku dysków, gdy stosowane są zwykłe (nie-klastrowe) systemy plików. W takim przypadku system plików nie może być zamontowany do zapisu i odczytu przez więcej niż jeden węzeł.

Oto przykładowy scenariusz obrazujący konieczność stosowania tej techniki:

- aktywny jest węzeł srv1
- w węźle srv1 występuje awaria, która powoduje, że węzeł przestaje chwilowo działać, ale nie umiera. W tym przypadku system plików jest w nieokreślonym stanie — na węźle srv1 są brudne bufory.
- srv2 traci puls srv1, więc montuje dysk do zapisu i odczytu, a następnie przejmuje funkcje węzła aktywnego
- srv1 nagle czuje się lepiej i zapisuje bufory na dysk

Podobna sytuacja może wystąpić nawet w większych klastrach, gdzie dostęp do zasobów jest kontrolowany przez konieczność uzyskania quorum — wystarczy, że w przypadku uszkodzenia połączeń na tranmitujących puls węzły uznają, że reszta węzłów umarła i że każdy z nich musi się aktywować. (Taka sytuacja nazywana jest „split-brain".)

Rozwiązaniem jest właśnie „resource fencing". Wyróżnia się dwa jego rodzaje:

- hard fencing, zwane też „resource-based fencing" umożliwiające zatrzymanie I/O na żądanie dzięki odpowieniemu wsparciu ze strony sprzętu. W tym celu wykorzystuje się na przykład:
  - komendy SCSI reserve/release
  - odpowiednie komendy Fibre Channel

  Metoda ta jest dobra z punktu widzenia wydajności, lecz w praktyce niemożliwa do zaimplementowania w heterogenicznych systemach, na jakich zazwyczaj działa Linux. Dlatego dla systemów innych niż firmowe, gdzie rodzaj urządzeń jest ściśle kontrolowany, stosuje się drugie podejście:
- STONITH (Shoot The Other Node In The Head)
  - polega na fizycznym wyłączeniu węzła uznanego za uszkodzony (failed) przy pomocy usług zarządzania klastrem
  - wykonywane jest przy pomocy specjalnych urządzeń do resetowania lub odcinania zasilania od maszyny
  - jest proste, tanie, uniwersalne

  STONITH został opisany w rozdziale 2.1.5.

1.4 Współdzielenie danych

Funkcję tą można realizować na kilka sposobów, w zależności od charakteru działających usług oraz sposobu modyfikacji danych:

- współdzielenie dzięki wsparciu sprzętowemu — gdy dostępne są takie dedykowane rozwiązania jak shared SCSI czy Fibre Channel. W tym przypadku przydaje się klastrowy system plików taki jak Global FileSystem, lub przynajmniej system plików obsługujący journalling
- automatyczna replikacja danych:
  - ciągła, jaką oferuje na przykład DRBD, albo NBD w połączeniu z jednym z systemów: RAID (moduł md), LVM lub EVMS
  - okresowa — wykonywana co pewien czas
  - „na żądanie" — na przykład po aktualizacji stron WWW są one rozsyłane automatycznie na innych węzłach. W tym i poprzednim przypadku pomocny może okazać się program rsync
  - wykorzystująca własne mechanizmy replikacji danej aplikacji czy usługi — na przykład replikacja MySQL, OpenLDAP, DNS, NIS itp...
- sieciowe systemy plików, takie jak NFS, codafs czy InterMezzo

2 Główne projekty i firmy oraz oferowane rozwiązania

Ten rozdział zawiera dostępne dla systemu Linux rozwiązania implementujące poszczególne z wyżej wymienionych podsystemów. Gdzie tylko to możliwe, są one pogrupowane według projektów, z którymi są związane.

2.1 Linux-HA — High-Availability Linux Project

2.1.1 Ogólne informacje o projekcie

Projekt ten[1] ma na celu stworzenie rozwiązania wysokiej dostępności dla systemów GNU/Linux w oparciu o clustering. Jest to projekt realizowany przez społeczność linuksową, wspierany przez takie firmy jak IBM, SGI, Intel, SuSE, Conectiva i inne.

Projekt Linux-HA współpracuje on także z projektem LVS (Linux Virtual Server, patrz rozdział 2.2), który ma na celu stworzenie klastra typu „load-balancing".

Należy także zaznaczyć, że twórcy projektu dążą do tego, aby Linux-HA stał się referencyjną implementacją Open Cluster Framework (OCF) (patrz rozdział 2.3).

2.1.2 Heartbeat

Podstawowym produktem projektu Linux-HA jest program `heartbeat`, realizujący pierwsze dwa z wymienionych w rozdziale 1 podsystemów klastrów wysokiej dostępności, czyli usługi uczestnictwa i komunikacji.

Szacuje sie, że istnieje kilka tysięcy instalacji tego systemu na systemach produkcyjnych. Strona projektu Linux-HA zawiera liczne wypowiedzi użytkowników programu heartbeat, którzy bardzo chwalą stabilność i niezawodność oprogramowania oraz łatwość jego instalacji i konfiguracji.[2]

Wchodzi on ponadto w skład dystrybucji SuSE Linux, Conectiva Linux, Mandrake Linux, MSC Linux oraz Debian GNU/Linux. Opierają się na nim Mission Critical Linux (rozdział 2.7), Ultramonkey (rozdział 2.6), a także systemy wbudowane kilku firm.

System, w jakim autorzy `heartbeat`-a proponują jego instalację, przedstawiono na rysunku 2.

Jest to właściwie typowy schemat klastra wysokiej dostępności (jak na rysunku 1), przy czym duży nacisk jest położony na unikanie „Single Points Of Failure".

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 (trzeba wtedy stosować specjalne przełącznice łączy szeregowych), a także korzystanie z innego typu współdzielenia danych. Chodzi jedynie o przedstawienie przykładowego środowiska działania programu heartbeat.

- Puls

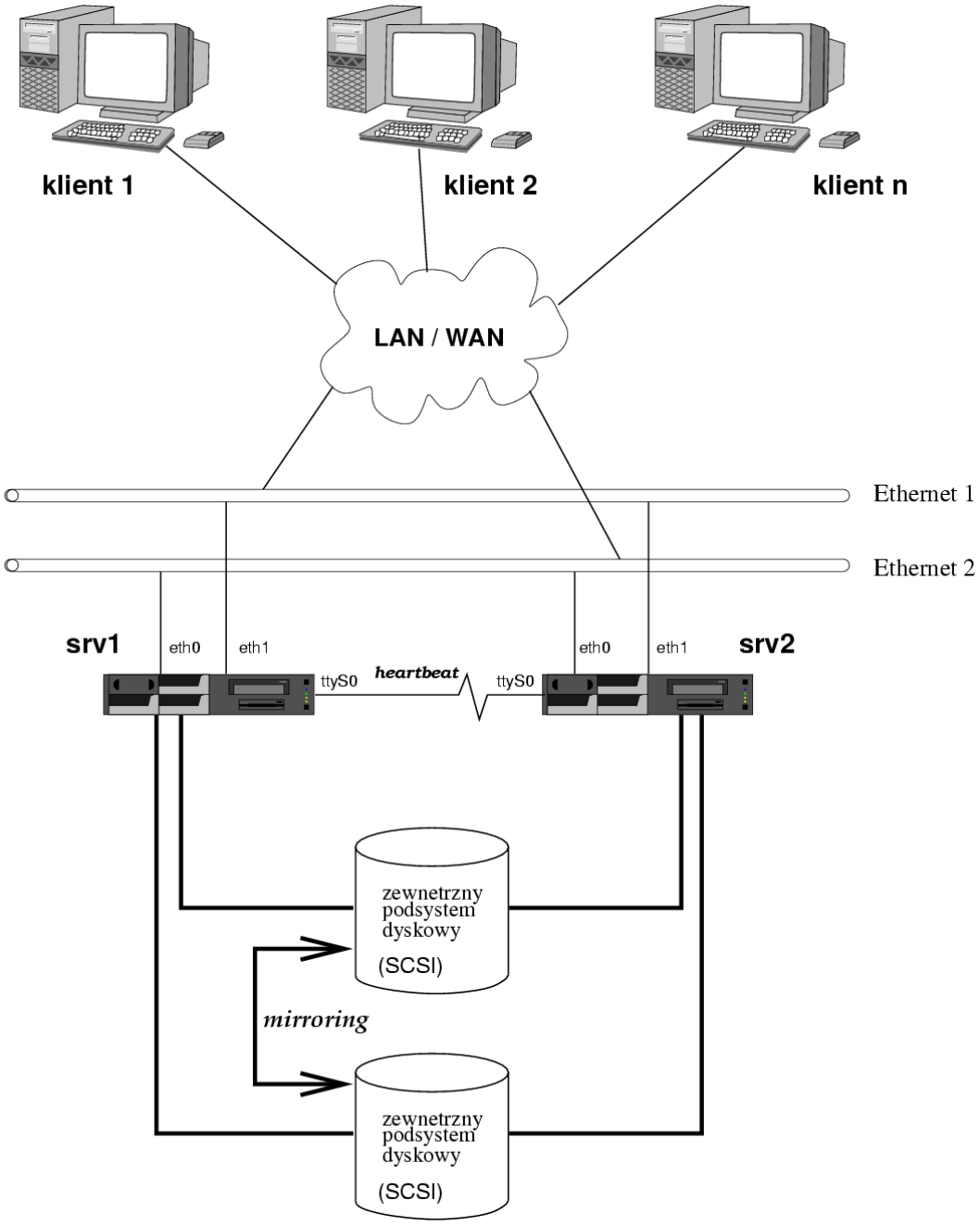
  Procesy heartbeat działające na maszynach srv1 i srv2 wymieniają między sobą komunikaty świadczące o działaniu każdego z węzłów.

  Komunikaty te `heartbeat` potrafi transmitować przez następujące media:

  – UDP (z wykorzystaniem multicastu, co ma znaczenie zwłaszcza w przypadku większych klastrów)

[1] Strona projektu: `http://linux-ha.org/`

[2] Uwagi użytkowników można przeczytać m.in. na stronie `http://linux-ha.org/heartbeat/users.html`



Rysunek 2: Schemat prostego systemu w jakim pracuje `heartbeat`

– linię szeregową „null-modem”
– połączenie PPP
– magistralę 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 nieskomplikowaną budowę, stanowi on praktycznie niezawodny kanał komunikacyjny dla sygnału „pulsu”.

Ponieważ heartbeat umożliwia kontrolowanie „pulsu” węzłów przez kilka mediów jednocześnie, bardzo zalecene jest wykorzystanie do tego celu wszystkich dostępnych niezależnych mediów łączących poszczególne węzły, gdyż zapewni to maksymalną pewność, że węzły nie utracą ze sobą kontaktu (jest to bardzo niekorzystne ze względów opisanych w rozdziale 1.3).

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, że 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 wybraną dystrybucję 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. kofigurujemy podsystem przechowywania danych (w tym przypadku system plików na zewnętrznej macierzy SCSI — `/dev/sda1`). 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 kontrolę nad usługami „apache” i „samba” pod adresem IP `1.2.3.4`, oraz montował w odpowienim 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

  Jeśli wszystko zostało wykonane poprawnie, mamy działający klaster!

  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 odpowieni sposób.

Jest to możliwe, o ile w katalogu `/etc/init.d` (czy innym, w zależności od dystrybucji) będą znajdować się skrypty `apache` i `samba` umożliwiające przez odpowiednie wywołanie (z parametrem `start` lub `stop`) wyłączenie lub włączenie poszczególnych usług.

Domyślnie pierwszy z włączonych węzłów przejmuje rolę 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), że 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 sieć przez którą 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 dołączających 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

Z projektem powiązane są także następujące programy:

2.1.3 Heart

Jest to program napisany przez Toma Vogta.

Implementuje podobną funkcjonalność co „heartbeat", jednak w ograniczonym zakresie i nie jest on już aktywnie rozwijany, więc nie został opisany szerzej.

2.1.4 Fake

Program ten został opisany w rozdziale 2.11.

Umożliwia on przejmowanie adresów IP (ang. *IP address takeover*). Zaktualizowane części tego kodu są wcielone w heartbeat, więc przydaje się on w zasadzie tylko przy tworzeniu systemu wysokiej dostępności nie wykorzystującego programu „heartbeat".

2.1.5 STONITH

STONITH[3] został wybrany przez projekt Linux-HA jako implementacja „resource-fencing".

Jest on zaimplementowany w formie biblioteki, która jest w stanie zresetować węzeł dzięki załadowaniu odpowiedniej wtyczki obsługującej dane urządzenie. Dostępnych jest obecnie 10 różnych wtyczek STONITH, z których każda obsługuje inne urządzenie resetujące:

- jedna obsługuje UPS-y,
- pięć obsługuje różne rodzaje switchy zasilających,
- jedna jest interfejsem dla oprogramowania zarządzającego klastrem „VACM" (patrz rozdział 2.13),
- jedna zgłasza potrzebę ręcznego zresetowania maszyny operatorowi i odbiera od niego potwierdzenie zresetowania,
- dwie inne używane są przeważnie do testów.

2.2 LVS — Linux Virtual Server

2.2.1 Ogólne informacje o projekcie

Misją projektu LVS (`http://www.linuxvirtualserver.org/`) 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 wysoką wydajność poprzez load-balancing, ale współpracuje także ściśle z projektem Linux-HA (patrz rozdział 2.1) i uwzględnia w proponowanych rozwiązaniach kwestię wysokiej dostępności.

Load-balancing można uzyskać na dwóch poziomach: aplikacji i IP.

Programy takie jak , Reverse-proxy[4] lub pWEB[5] rozwiązują rozkładanie obciążenia na poziomie protokołu HTTP, przekazując poszczególne żądania 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.

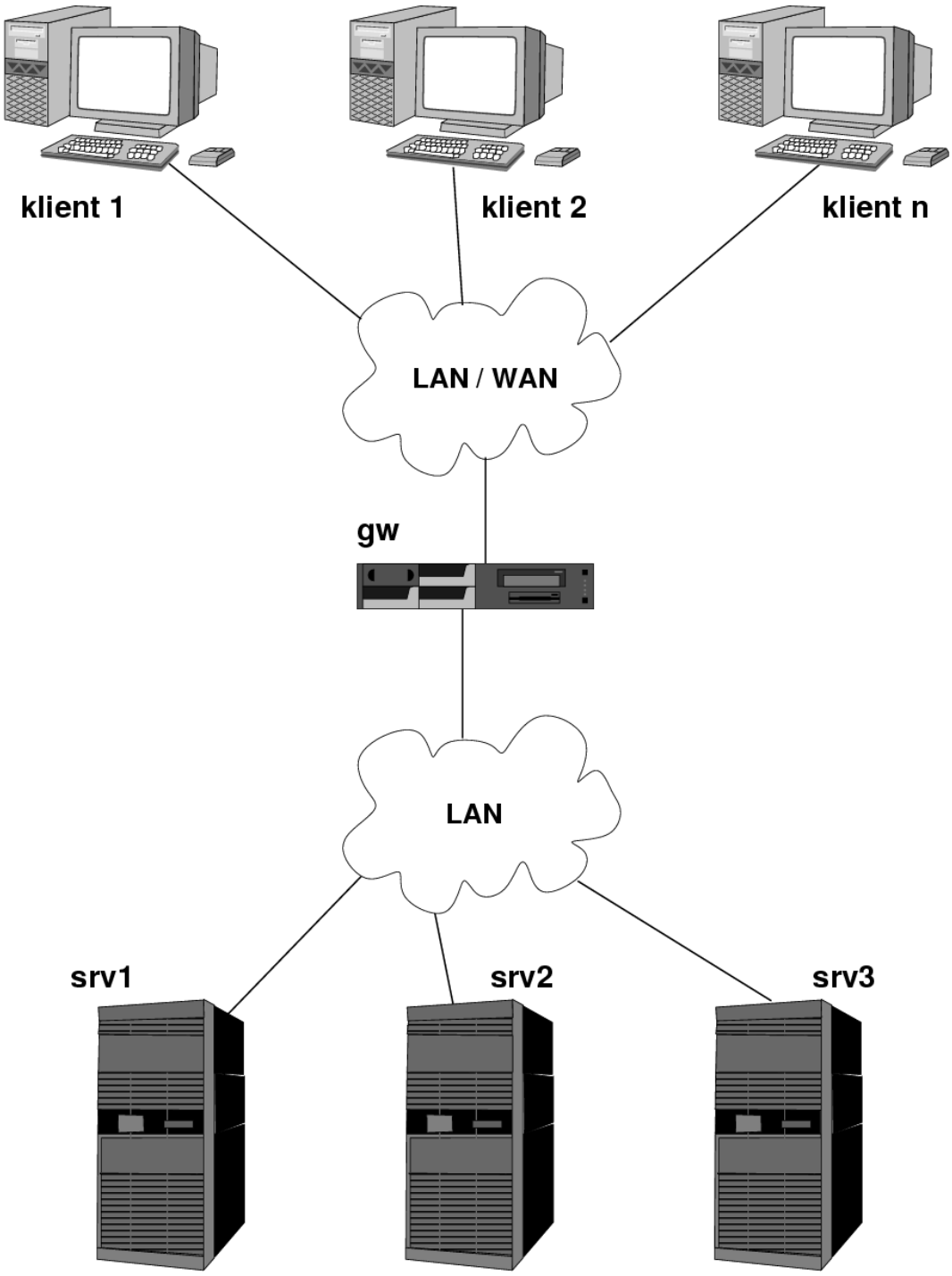
LVS stoi za takimi adresami jak linux.com, sourceforge.net, themes.org, www.zope.org, www.real.com i wieloma innymi[6].

2.2.2 Proponowane rozwiązania

Rysunek 3 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.

[3]Strona podprojektu: `http://linux-ha.org/stonith.html`
[4]Strona programu reverse-proxy: `http://www.engelschall.com/pw/wt/loadbalance/`
[5]Strona programu pWEB: `http://www.nsrc.nus.sg/STAFF/edward/`
[6]Inne przykłady zastosowania LVS: `http://www.linuxvirtualserver.org/deployment.html`



Rysunek 3: Schemat najprostszego systemu LVS

Maszyna `gw` nosi w nomenklaturze LVS nazwę „directora", ponieważ kieruje żądania 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 żądanie.

Cały klaster dzięki directorowi widziany jest w konsekwencji jako „serwer wirtualny".

Jak wspomniano wcześniej, przekazywanie żądań 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, jaką oferuje round-robin DNS, ponieważ w przypadku uszkodzenia serwera rzeczywistego kierowane do niego żądania nie będą obsługiwane. Jednak nawet w takim przypadku posiadana redundancja zapewnia obsługę przynajmniej części klientów (tych, których żądania zostały przekazane do sprawnych serwerów wirtualnych).

Stosując jednak odpowiednie oprogramowanie (na przykład `mon` — patrz niżej), kontrolujące dostępność poszczególnych serwerów wirtualnych, można w bardzo prosty sposób uzyskać całkiem niezłą dostępność.

Drugim mankamentem przedstawionego wyżej rozwiązania jest to, że sam director stanowi „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 wysoką dostępność wyżej pokazanego systemu, stosując redundantny director, jak przedstawiono na rysunku 4, oraz kilka narzędzi opisanych niżej.

Wyżej przedstawioną konfigurację można w łatwy sposób uzyskać stosując na przykład program `heartbeat` (patrz rozdział 2.1.2).

2.2.3 Narzędzia

LVS dostarcza kilka elemtentów programowych, które w połaczeniu umożliwiają zbudowanie wysoko-dostępnego systemu zapewniającego jednocześnie wysoką wydajnoś. Instaluje się je na maszynie (maszynach) działającej (-ych) jako director, pracującej (-ych) pod kontrolą systemu GNU/Linux.

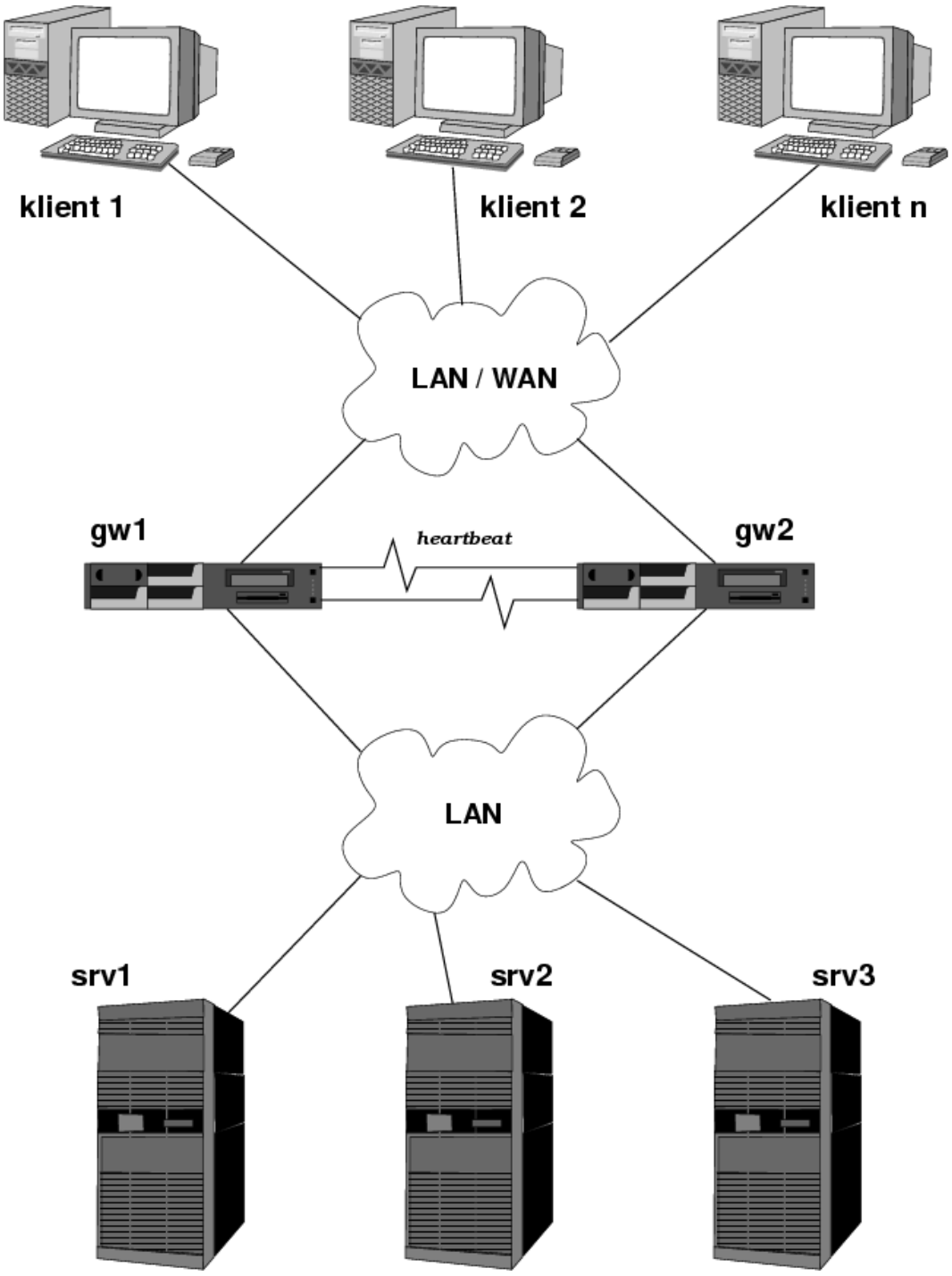
Moduły jądra Umożliwiają one wydajny load-balancing, bo implementowany jest on na poziomie jądra:

- łata virtual server dla jądra 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 wymienioną łatę lub moduł. Używa się go podobnie do iptables, ipchains czy ipfwadm.

mon Jest to program przeznaczony do monitorowania usług (opisany w rozdziale 2.10). W łatwy sposób [7] można skonfigurować go tak, aby w razie uszkodzenia które-

[7] Przykład konfiguracji opisany jest na przykład pod adresem `http://www.linuxvirtualserver.org/HighAvailability.html`



Rysunek 4: Schemat systemu LVS z redundantnym load-balancerem

goś z serwerów rzeczywistych, serwer ten został usunięty z puli dostępnych maszyn.

Dzięki temu można uzyskać failover serwerów rzeczywistych realizowany przez „trzecią" maszynę — director.

heartbeat Jest to program świadczący usługi uczestnictwa i komunikacji w klastrach wysokiej dostępności (opisany w rozdziale 2.1.2). 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 autorstwa Jacoba Riefa przeznaczony do monitorowania działania usług na rzeczywistych serwerach. Obecnie jest zaimplementowane monitorowanie usług HTTPS i HTTPS.

Spełnia on podobną funkcję jak „mon", ale został napisany specjalnie do użytku z LVS — na przykład automatycznie czyta odpowienie pliki konfiguracyjne LVS i samodzielnie dokonuje odpowienich 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.

ldirectord jest dystrybuowany razem z LVS.

keepalived Jest to również monitor usług, posiadający kilka ciekawych cech. Został opisany w rozdziale 2.12.

fake Jest to program umożliwiający odebranie adresu IP innej maszynie (patrz rozdział 2.11). Można użyć go w celu przejęcia kontroli nad load-balancingiem, gdy nie wykorzystujemy programu `heartbeat` (który samodzielnie wykonuje odbieranie adresu IP).

2.3 Open Cluster Framework

Open Cluster Framework (OCF)[8] to projekt mający na celu zdefiniowanie API udostępniającego podstawowe funkcje dostępne w klastrze, a także stworzenie implementacji tego API.

Projekt zwraca uwagę zarówno na klastry HA (wysokiej dostępności), jak i HPC (klastry wysokiej wydajności).

Choć projekt jest na razie we wczesnej fazie rozwoju, jest on wart odnotowania choćby z tego powodu, że wspierają go takie potęgi jak IBM, SGI czy COMPAQ.

2.4 Red Hat, Inc.

2.4.1 Piranha

Ta firma oferuje pakiet „piranha", będący gotowym rozwiązaniem High-Availability, korzystających z dostępnych powszechnie a także napisanych specjalnie komponen-

[8]Strona projektu: `http://opencf.org/`

tów, którego rdzeniem jest LVS. Dlatego wszystko co powiedziano wyżej na temat LVS (w rozdziale 2.2), odnosi się też do piranhii.

Dodatkowo piranha ma też prymitywną funkcję dostosowywania w czasie rzeczywistym algorytmu dystrybucji żądań w zależności od średniego obciążenia poszczególnych serwerów, o ile możliwe jest pobranie jego wartości przy pomocy komendy `rsh`.

W skład tego pakietu wchodzą:

LVS Sercem piranhii jest opisany wyżej (patrz rozdział 2.2) pakiet do tworzenia wirtualnych serwerów, czyli kod w jądrze oraz narzędzie ipvsadm.

lvs Tablicą routingu LVS zarządza przy pomocy ipvsadm daemon lvs.

nanny Deamon monitorujący usługi rzeczywistych serwerów — odpowiednik „mon" (rozdział 2.10) lub „ldirectord" (rozdział 2.2.3).

pulse Jest to odpowiednik opisanego wyżej daemona heartbeat (opisany w rozdziale 2.1.2). Umożliwia on „failover" aktywnego routera w przypadku jego awarii.

piranha Graficzny interfejs użytkownika służący do konfiguracji całego pakietu. Mimo, iż Red Hat dostarcza GUI, cały serwer da się równie łatwo skonfigurować i uruchomić na maszynie nie posiadającej karty grafiki czy monitora.

2.5 TurboLinux

Firma TurboLinux, Inc. stworzyła komercyjny produkt do tworzenia klastrów wysokiej dostępności nazwany TurboCluster.

2.5.1 TurboCluster

Strona produktu: http://www.turbocluster.com/

Rozwiązanie to składa się z modułów jądra, udostępnionych na zasadzie licencji GNU GPL, oraz z aplikacji trybu użytkownika (w tym GUI do konfiguracji), o zamkniętych źródłach. Jest dystrybuowany jako część zmodyfikowanej dystrybucji TurboLinux.

TurboCluster umożliwia przełączanie w warstwie czwartej w podobny sposób jak umożliwiają to LVS i heartbeat.

2.6 Ultra Monkey

Projekt Ultra monkey[9] 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 tworzeniu skalowalnych „web farms" wysokiej dostępności, choć można tego rozwiązania używać także do takich usług jak e-mail i FTP.

[9]Strona projektu: `http://www.ultramonkey.org/`

Sercem Ultra monkey jest LVS (patrz rozdział 2.2), a failover zapewnia pakiet heartbeat (rozdział 2.1.2). Usługi są monitorowane przy pomocy daemona ldirectord (rozdział 2.2.3). 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.

2.7 Mission critical Linux

Mission critical Linux[10] to firma utworzona w 1999 roku przez byłych pracowników Compaqa i Digitala. Misją firmy jest tworzenie rozwiązań wysokiej dostępności w oparciu o oprogramowanie Open Source dla systemu Linux.

2.7.1 Convolo cluster — NetGuard edition

Sztandarowy produkt MCLX, NetGuard zbudowany jest na opisanych wyżej komponentach: LVS (rozdział 2.2) oraz heartbeat (rozdział 2.1.2) i jest przeznaczony do działania na tanim sprzęcie, bez dodatkowych urządzeń typu zewnętrzne macierze dyskowe. Jest on obecnie w fazie beta.

2.7.2 Kimberlite

Drugi z produktów MCLX jest przeznaczony do działania na sprzęcie specjalnie zaprojektowanym dla klastrów wysokiej dostępności. Jest w stanie wykorzystać takie urządzenia jak zewnętrzne macierze SCSI czy Fibre Channel.

2.8 Motorola, Inc

2.8.1 Advanced HA

High-Availability Linux 3.0[11] to produkt, którym motorola chce rozpocząć nową erę wysokiej dostępności — 6NINES (sześć dziewiątek), czyli 99,9999% dostępności, co odpowiada trzydziestu sekundom planowanej i nieplanowanej niedostępności systemu.

[10] Strona firmy: `http://www.missioncriticallinux.com/`

[11] Strona projektu: `http://mcg.motorola.com/cfm/templates/swdetail.cfm?PageID=682\&PageTypeID=10\&SoftwareID=6\&ProductID=202`

HA Linux to rozwiązanie, w którego skład wchodzi zarówno odpowiednie oprogramowanie (patrz niżej), jak też odpowiedni sprzęt. Obecnie Motorola oferuje HA Linuksa w połaczeniu z platformą 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 magistralą 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, że 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 odpowiednią 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 sieć i ich obsługę (na przykład przejęcie 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.

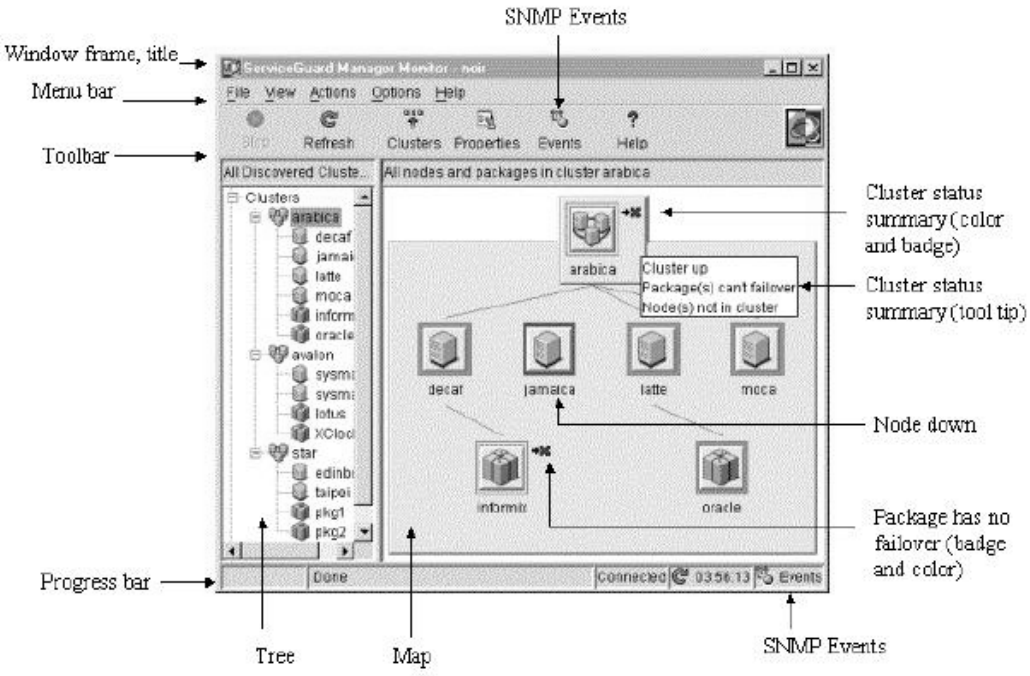
Checkpoint services umożliwia znaczne przyspieszenie czasu przełączeń (failover). W systemie typu 6NINES niedopuszczalna jest taka sytuacja, że w przypadku awarii węzła węzeł zapasowy jest inicjalizowany, a następnie przejmuje rolę uszkodzonego. Dlatego stworzono mechanizm polegający na częstej wymianie stanu (checkpointing) między węzłem aktywnym a zapasowymi. Dzięki temu węzeł zapasowy może zacząć pracę od razu.

Sterowniki sprzętu w HA Linuksie pozwalają na efektywną diagnostykę stanu urządzeń i szybkie zgłaszanie ewentualnych awarii do managera zdarzeń, który następnie podejmuje odpowiednie działanie.

Autorzy systemu twierdzą, że HA Linux pozwoli na osiągnięcie czasów przełączeń (w wypadku uszkodzenia systemu) rzędu 1-2 sekund, co rzeczywiście umożliwiłoby osiągnięcie dostępności rzędu 6NINES.

2.9 Hewlett-Packard

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



Rysunek 5: Zdjęcie ekranu ServiceGuard manager

2.9.1 HP Multi-Computer/Serviceguard

MC/Serviceguard[12] zapewnia wysoką 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 5.

HP MC/ServiceGuard działa na serwerach HP ProLiant DL380 G2 oraz DL580 G2, z dystrybucją Red Hat Professional 7.3 (w przypdku interfejsu SCSI) oraz Red Hat Advanced Server 2.1 (w przypadku użycia Fibre Channel).

2.10 mon — service monitoring daemon

mon[13] to narzędzie przeznaczone ogólnie do monitorowania dostępności usług i alarmowania w razie wykrycia ich niedostępności. Został on zaprojektowany tak, aby być otwartym i rozszerzalnym — obsługuje on monitorowanie dowolnych usług

[12]Strona produktu: `http://www.hp.com/products1/unix/highavailability/`
[13]Strona projektu: `http://www.kernel.org/software/mon/`

i alarmowanie w dowolny sposób dzięki wspólnemu interfejsowi, który można zaimplementować w łatwy sposób w C, Perlu, shellu itp.

mon traktuje monitorowanie zasobów jako dwa oddzielne zadania:

- sprawdzanie stanu
- wyzwalanie jakiegoś działania w przypadku wykrycia uszkodzenia

Każde z tych działań jest realizowane przez zewnętrzny program, mon jest tylko schedulerem, wykonującym monitory (każdy z nich sprawdza jakiś warunek) i wywołującym alarmy w przypadku, gdy monitor zawodzi. Logika rządząca działaniem mon-a podlega konfiguracji przez użytkownika.

Same monitory i alarmy nie są częścią serwera, choć w pakiecie jest dostarczonych kilka najbardziej przydatnych. Taka architektura umożliwia łatwą rozbudowę i dostosowanie programu do indywidualnych potrzeb.

2.11 Fake

Jest to program[14] autorstwa Simona Hormana.

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, że maszyna i system operacyjny nadal będą pracować (zajmując adres IP), ale dana usługa będzie niedostępna.

W tym celu fake konfiguruje odpowiedni interfejs z danym adresem IP (wykorzystując IP aliasing), a następnie podszywa się pod daną maszynę wykorzystując ARP spoofing.

2.12 keepalived

Jest to projekt[15] ściśle związany z LVS. Głównym jego 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 też stos VRRPv2, który umożliwia monitorowanie stanu samego directora i ewentualny jego failover. Jest więc on w stanie zastąpić zarówno heartbeat jak i mon, czy ldirectord.

2.13 VA Cluster Manager

VACM[16] to oprogramowanie do zarządzania klastrem, stworzone przy współpracy VA Linux Systems, znanego producenta serwerów dedykowanych dla Linuksa.

Oto jego najważniejsze cechy:

[14]Strona projektu: `http://www.vergenet.net/linux/fake/`
[15]Strona projektu: `http://keepalived.sourceforge.net/`
[16]Strona projektu: `http://vacm.sourceforge.net/`

- zawiera moduł stanu systemu umożliwiający odczytywanie takich statystyk systemu operacyjnego jak użycie pamięci, dysku, obciążenie, ilość procesów itp...
- stosunkowo rozbudowany system przywilejów, umożliwiający tworzenie klas użytkowników posiadających różne uprawnienia.
- obsługa SSL
- API umożliwiające pisania modułów i klientów VACM w językach innych niż C
- klient linii poleceń (VASH)
- obsługa sieciowych koncentratorów szeregowych (CISCO, Digi Etherlite)

2.14 Sistina

Firma ta[17] stworzyła między innymi LVM (Logical Volume Manager) oraz GFS (Global File System).

2.14.1 GFS — Global File System

Jest to stosunkowo zaawansowany i dojrzały system plików, jeśli chodzi o systemy plików dla klastrów wysokiej dostępności. Używają go takie organizacje jak NASA — centrum lotów kosmicznych Goddarda, czy Fermilab.

Umożliwia on bepieczny jednoczesny dostęp do systemu plików w trybie do zapisu i odczytu przez wiele węzłów klastra.

Jego najważniejsze zalety:

bezpośrednie I/O zwiększa wydajność — ponieważ omija buffer cache, nadaje się do użytku przez zaawansowane systemy baz danych takie jak Oracle

dobra skalowalność producent twierdzi, że jest ona do czterech razy lepsza od tej, jakie oferują inne klastrowe systemy plików

architektura OmiLock umożliwia wydajne, skalowalne blokowanie plików oraz wybór mechanizmu blokowania

quoty obsługuje mechanizm miękkich i twardych quot dyskowych

journaling umożliwia szybkie doprowadzenie systemu plików do spójnego stanu

rozproszone metadane wyklucza powstawanie wąskich gardeł jak w przypadku centralnego serwera metadanych

zgodność z POSIX

Niestety jest to system plików o zamkniętym źródle, za darmo można go ewaluować tylko przez 30 dni.

[17]Strona firmy: `http://www.sistina.com/`

2.15 Coda filesystem

Coda[18] to zaawansowany sieciowy, rozproszony system plików. Jest rozwijany na uniwersytecie Carnegie Mellon od 1987 roku przez grupę systemów M. Satyanarayanana. Ten system plików wyrósł z AFSv2.

Oto kilka ciekawych cech tego systemu plików:

- praca bez połączenia z siecią
- wolnodostępny — liberalna licencja
- cache po stronie klienta
- replikacja po stronie serwera
- model bezpieczeństwa umożliwiający autoryzację, szyfrowanie i kontrolę dostępu
- umożliwia dalszą pracę nawet w przypadku uszkodzenia części sieci serwerów
- adaptacja do przepustowości sieci
- dobra skalowalność
- dobrze zdefiniowana semantyka, nawet w przypadku uszkodzenia sieci

2.16 Inter Mezzo

InterMezzo[19] to rozproszony to system plików przeznaczony przede wszystkim dla rozwiązań wysokiej dostępności, objęty licencją GPL. Ten system plików był zainspirowany wymienionym wyżej systemem plików codafs (rozdział 2.15), ale został kompletnie przeprojektowany i zaimplementowany od zera.

Nadaje się do replikacji serwerów, do komputerów przenośnych, zarządzania oprogramowaniem systemowym na dużych klastrach, a także do stosowania w klastrach wysokiej dostępności.

InterMezzo umożliwia na przykład pracę bez połączenia, automatyczne przywracanie stanu po awarii sieci.

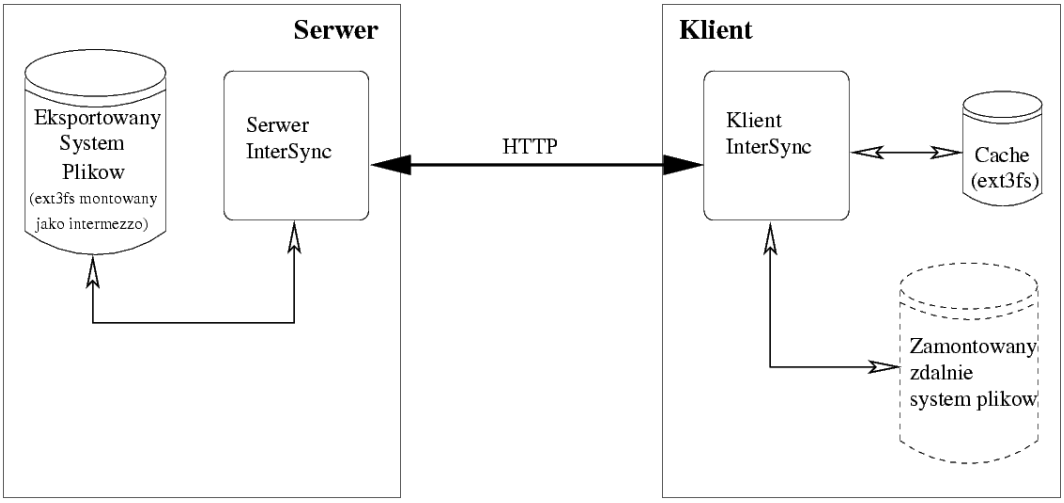
InterMezzo jest zaimplementowane jako „nakładka" na prawdziwy (najlepiej obsługujący journalling) system plików, w którym InterMezzo przechowuje (w ukrytych katalogach) informacje na temat stanu (patrz rysunek 6.

System składa się z modułów jądra, oraz daemona InterSync działającego w trybie użytkownika, zarówno po stronie serwera, jak i po stronie klienta. Deamon po stronie serwera śledzi modyfikacje plików na eksportowanej partycji, a daemon klienta co pewien czas łączy się z serwerem i pobiera modyfikacje. Komunikacja między procesami InterSync odbywa się w protokole HTTP co ma zapewnić rozszerzalność tego systemu plików o nowe funkcje, o które można rozszerzyć zwykły serwer HTTP (na przykład szyfrowanie).

2.17 Linux Network Block Device

Oryginalna implementacja NBD została napisana przez Pavla Machka w 1997 roku i dołączona do jądra w wersji 2.1.101.[20]

[18]Strona projektu: `http://www.coda.cs.cmu.edu/`
[19]Strona projektu: `http://inter-mezzo.org/`
[20]Strona projektu: `http://nbd.sourceforge.net/`



Rysunek 6: Schemat architektury intermezzo

Obecnie przez Petera Breuera rozwijany jest projekt Enhanced NBD, mający na celu usprawnianie implementacji NBD.[21].

Network Block Device (NBD) to moduł jądra umożliwiający reprezentację połączenia TCP jako urządzenia blokowego (`/dev/nd0`). Urządzenie takie można traktować jako zwykłą partycję, a co za tym idzie na przykład włączyć jako jeden z woluminów przy konstruowaniu programowej macierzy RAID.

W szczególności po drugiej stronie połączenia może działać specjalny serwer, który udostępnia faktyczną partycję na zdalnym hoście. Jest to przedstawione na rysunku 7.

Przy każdej próbie odczytu z `/dev/nd0` na maszynie klienckiej zostanie wysłane przez TCP do zdalnego serwera żądanie przesłania potrzebnych bloków.

Ponieważ NBD działa na dużo niższym poziomie jądra niż na przykład NFS, na partycji zamontowanej przez NBD można umieścić dowolny system plików.

Z drugiej strony, w danej chwili tylko jeden klient może bezpiecznie montować system plików NBD do zapisu i odczytu.

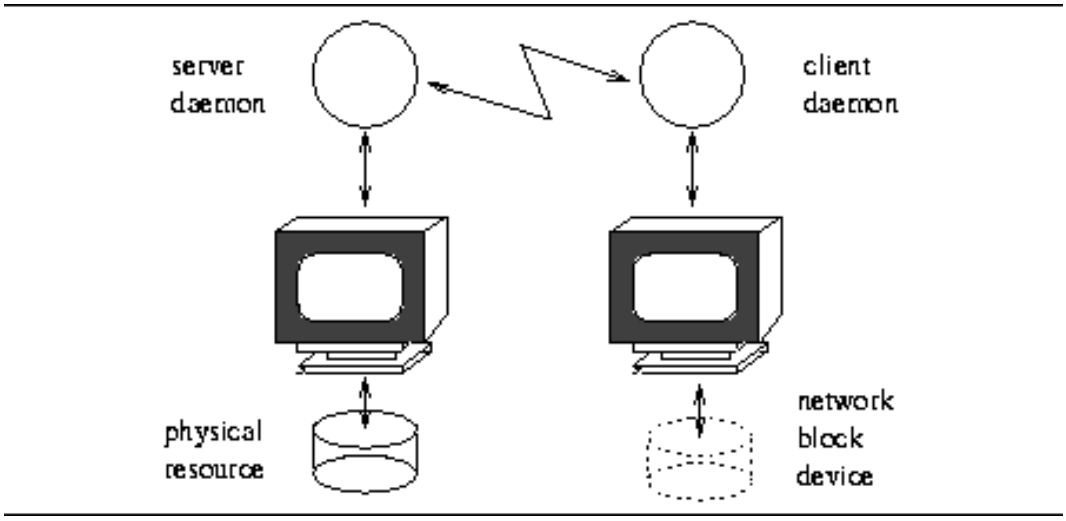
2.18 DRBD

DRBD[22] to moduł jądra autorstwa Philipa Reisnera, służący do mirrorowania systemów plików przez sieć.

Działanie DRBD polega na tym, że każdy blok danych zapisywany lokalnie na dysku jest także przesyłany do zdalnego węzła, na którym również zostaje zapisany. Odczyty są zawsze wykonywane lokalnie. Jest więc 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 rozwinięcie tego oprogramowania tak, aby więcej niż jeden węzeł miał taki dostęp.

[21]Strona projektu: `http://www.xss.co.at/linux/NBD/`
[22]Strona projektu: `http://www.complang.tuwien.ac.at/reisner/drbd/`



Rysunek 7: Schemat architektury NBD

Na DRBD zrealizowano całkiem poważne systemy produkcyjne, w tym dwuterabajtowy system plików na uniwersytecie Utah, czy największe centrum danych w Japonii (ponad 4000 komputerów) — patrz `http://www.complang.tuwien.ac.at/reisner/drbd/deploy.html`.

Literatura

[1] High-availability linux project, październik 2002. `http://linux-ha.org/`.

[2] Harald Milz (hm@seneca.muc.de). Linux high availability HOWTO, grudzień 1998. `http://www.ibiblio.org/pub/Linux/ALPHA/linux-ha/High-Availability-HOWTO.html`.

[3] Rudy Pawul (rpawul@iso ne.com). Getting started with Linux-HA (heartbeat), 2000. `http://linux-ha.org/download/GettingStarted.html`.

[4] Alan Robertson (alanr@unix.sh). Linux-ha APIs. Talk given at LWCE/NYC in February, 2001. `http://linux-ha.org/heartbeat/LWCE-NYC-2001/index.html`.

[5] Alan Robertson (alanr@unix.sh). Implementing HA servers on Linux — a brief tutorial on the Linux-HA heartbeat software. `http://linux-ha.org/heartbeat/DevDen2002.pdf`.

[6] Steve Blackmon (steve.blackmon@transtech.cc). High-availability file server with heartbeat, 2001. `http://www.samag.com/documents/s=1146/sam0109c/0109c.htm`.

[7] Ram Pai. Heartbeat API. `http://linux-ha.org/heartbeat/heartbeat_api.html`.

[8] Horms (Simon Horman) (horms@verge.net.au). Fake home page, 2002. `http://www.vergenet.net/linux/fake/`.

[9] Alan Robertson (alanr@suse.com). Linux-HA heartbeat system design, 2000. http://www.linuxshowcase.org/2000/2000papers/papers/robertson/.

[10] Richard Ferri (rcferri@us.ibm.com). Conversations: Introducing the open cluster framework, wrzesień 2002. http://www.linuxjournal.com/article.php?sid=6143.

[11] Ip load balancing (piranha), 2002. http://www.redhat.com/software/advancedserver/technical/piranha.html.

[12] Linux virtual server home page. http://www.linuxvirtualserver.org/.

[13] Joseph Mack (jmack@wm7d.net). LVS-mini-HOWTO, listopad 2002. http://www.linuxvirtualserver.org/Joseph.Mack/mini-HOWTO/LVS-mini-HOWTO.html.

[14] mon home page, 2002. http://www.kernel.org/software/mon/.

[15] Keepalived home page, 2002. http://keepalived.sourceforge.net/.

[16] RFC2338 — virtual router redundancy protocol. http://www.ietf.org/rfc/rfc2338.txt.

[17] Alexandre Cassen (acassen@linux vs.org). Keepalived user guide, 2002. http://keepalived.sourceforge.net/pdf/UserGuide.pdf.

[18] Horms (Simon Horman) (horms@verge.net.au). Creating linux web farms (linux high availability and scalability), listopad 2000. http://verge.net.au/linux/has/.

[19] Horms (Simon Horman) (horms@verge.net.au). Ultra monkey project home page, 2002. http://www.ultramonkey.org/.

[20] Inc Mission Critical Linux. Mission critical linux website, 2002. http://www.missioncriticallinux.com/.

[21] Mission critical linux to deliver the first clustering solution specifically developed for e-commerce, marzec 2002. http://linuxpr.com/releases/1488.html.

[22] Motorola Computer Group. Advanced high availability services for linux, 2002. http://mcg.motorola.com/cfm/templates/swdetail.cfm?PageID=682&PageTypeID=10&SoftwareID=6&ProductID=202.

[23] Charles de Tranaltes. The road to six nines (6NINES) availability, luty 2002. http://mcg.motorola.com/wp/index.cfm?pagetypeid=35&source=6.

[24] HP high-availability software, 2002. http://www.hp.com/products1/unix/highavailability/.

[25] Global filesystem home page. http://www.globalfilesystem.org/.

[26] Alan Robertson (alanr@us.ibm.com). Resource fencing using STONITH. http://linux-ha.org/heartbeat/ResourceFencing_Stonith.html.

[27] Non-stop authentication with linux clusters. http://www-1.ibm.com/servers/esdd/articles/linux_clust/index.html.

[28] Coda filesystem home page, 2002. http://www.coda.cs.cmu.edu/.

[29] Inter Mezzo filesystem home page, 2002. http://inter-mezzo.org/.

[30] Bill von Hagen (vonhagen@vonhagen.org). Using the InterMezzo distributed filesystem — getting connected in a disconnected world, 2002. http://www.linuxplanet.com/linuxplanet/reports/4368/1/.

[31] OCF. Open Cluster Framework project home page, 2002. http://opencf.org/.

[32] VA Cluster Manager project home page, 2002. http://vacm.sourceforge.net/.

[33] Philipp Reisner (philipp.reisner@gmx.at). DRBD home page, 2002. http://www.complang.tuwien.ac.at/reisner/drbd/.

[34] Pavel Machek. NBD project home page. http://nbd.sourceforge.net/.

[35] Peter Breuer. Enhanced NBD project home page. http://www.xss.co.at/linux/NBD/.