# Realizacja niezawodnych usług w rozwiązaniu SSI

Projekt z przedmiotu RSO

Wojciech Maziarz
wmaziarz@mion.elka.pw.edu.pl

16 czerwca 2005

## Spis treści



# 1 Wstęp

Celem projektu jest opracowanie serwera świadczącego niezawodną usługę, działającego w klastrze OpenSSI. Serwer ma być niezawodny, co oznacza, że w przypadku awarii jednego z węzłów klastra, ma działać i kontynuować komunikację z klientem. Serwer ma być stanowy, a jego stan (stan komunikacji z klientem) nie może ulec zmianie w przypadku awarii jednego z węzłów.

# 2 Niezawodność serwera w OpenSSI

Klaster OpenSSI jest ”widziany” przez klienta jak jedna maszyna. Taka pzezroczystość jest osiągana dzięki zastosowaniu CVIP (Cluster Virtual IP). W ten sposób, łącząc się z klastrem, np przez ssh, nie wiemy (ale możemy to sprawdzić), na którym węźle jesteśmy zalogowani, na którym działają nasze procesy. Mamy do dyspozycji zasoby z wszystkich węzłów składających się na klaster. Jednakże jeśli nastąpi awaria węzła, na którym działają nasze procesy, przestaną działać, mimo, że reszta systemu

nadal będzie dostępna. Nie mamy tu więc do czynienia z niezawodnością rozwiązania (High Availability).

Celem projektu jest zaproponowanie rozwiązania serwera, które będzie niewrażliwe na awarie pojedynczych węzłów. Jak powinien wyglądać taki serwer? OpenSSI zapewnia pewne mechanizmy, które można wykorzystać do stworzenia HA. Najważniejszym z nich wydaje się być demon **keepalive** . Jest to demon, który monitoruje zarejestrowane procesy. Jeśli któryś z takich procesów ulegnie awarii, jest on uruchamiany na nowo przez keepalive.

## 2.1 Keepalive

Keepalive jest demonem monitorującym procesy i inne demony zarejestrowane w nim przy użyciu interfejsu **spawndaemon** . Kiedy zerejestrowany proces lub demon ulegnie awarii, keepalive uruchamia skrypt restartujący ten proces. Restartowany proces może zostać uruchomiony na dowolnym, wskazanym przez użytkownika węźle.

Keepalive przy starcie bądź restartowaniu procesów korzysta ze standardowych skryptów shellowych. Skrypty te, napisane przez użytkownika mogą mieć dowolne nazwy, ale muszą być umieszczone w katalogu /etc/keepalive.d. Muszą być własnością root'a i mieć ustawione prawa dostępu na 0755, by zapobiec możliwości nadpisania ich przez innych użytkowników.

Jeśli program wykonywalny dla monitorowanego procesu jest umieszczony w zdalnym systemie, potrzebne są prawa root'a by uruchomić lub zrestartować ten proces.

Każdy proces lub demon uruchomiony przez keepalive ma przekierowane strumienie standardowy wyjściowy oraz błedów do pliku z logami
/var/log/keepalive/daemon_basename.process_id .
Strumienie wejściowe są przekierowane na partycję / (root). Logi nie są gromadzone na stałe. Wyrejestrowanie procesu z demona keepalive powoduje wykasowanie jego logów. Również w przypadku gdy proces jest restartowany, jego poprzednie logi sa usuwane. Keepalive uruchomiony z opcją -i kasuje wszystkie nieaktualne logi.

Opcje polecenia keepalive

- **-i** usuwa plik /etc/keepalive.d/keepalive.data przed uruchomieniem demona. Keepalive uruchomiony jest z wyczyszczoną tablicą monitorowanych procesów.
- **-t interval** Definiuje interwał (w sekundach) odpytywania procesów. Wartością domyślną jest 5. Przepytywanie jest stosowane na przykład przy wywoałaniu fork(2).
- **-P node** Definiuje węzeł, na którym ma być uruchomiony demon keepalive w pierwszej kolejności. Proces keepalive jest przywiązany do konkretnego węzła w klastrze i nie może być migrowany przy użyciu load_leveld().
- **-S node** Definiuje węzeł, na którym ma być uruchomiony demon keepalive w drugiej kolejności w przypadku gdy pierwszy węzeł jest niedostępny. Proces keepalive jest przywiązany do konkretnego węzła w klastrze i nie może być migrowany przy użyciu load_leveld().

Jeśli jeden z wyspecyfikowanych węzłów jest niedostępny, keepalive generuje ostrzeżenie i kontynuuje wykonywanie na innym węźle.

Keepalive jest zawsze przypisany do węzła, na którym jest uruchomiony niezależnie od tego, czy węzeł ten został jawnie zdefiniowany.

**Pliki używane przez keepalive**

- **/dev/keepalivecfg** Potok nazwany do przyjmowania poleceń
- **/etc/keepalive.d** Katalog, w którym umieszczone są skrypty restartowe monitorowanych procesów bądź demonów.
- **/etc/keepalive.d/keepalive.data** Plik z tablicą monitorowanych procesów.
- **/var/log/keepalive** Katalog z logami procesów monitorowanych przez keepalive

## 2.2 Spawndaemon

Spawndaemon jest interfejsem lini poleceń dla demona Keepalive.

Aby przekazać proces lub demon pod kontrolę keepalive, należy wykonać następujące czynności

1. Utworzyć plik konfiguracyjny dla procesu w katalogu /etc/spawndaemon.d
2. Utworzyć skrypt startowy w jednym z katalogów /etc/rc*.d
3. Utworzyć skrypt startowy procesu w katalogu /etc/keepalive.d. Jest on wykorzystywany do uruchomienia i restartowania monitorowanego procesu.
4. Opcjonalnie utworzyć skrypty w katalogu /etc/keepalive.d. Mogą one zostać użyte do restartu procesu w przypadku awarii węzła lub procesu, w przypadku przekroczenia ustalonej liczby monitorowanych procesów. Mogą być także użyte do zamykania procesu lub demona.
5. Uruchomić spawndaemon c lini poleceń lub poprzez restart systemu.
6. Sprawdzić czy proces został poprawnie zarejestrowany. Można do tego celu wykorzystać komendę **spawndaemon -L** lub z opcją **-v** .

Aby zarejestrować grupę procesów lub demonów, każdy z nich musi zostać zarejestrowany i mieć swoje skrypty startowe i restartowe tak jak w przypadku nizależnych procesów. Dodatkowo grupa musi mieć swój plik konfiguracyjny.

Polecenie **spawndaemon** dostępne jest z wieloma opcjami. Najważniejsze z nich to

- **-d** *nazwa_procesu* wyrejestrowanie procesu o podanej nazwie.
- **-F** *numer_wezla* uruchomienie procesu na wskazanym węźle. Proces nie może być migrowany przy użyciu programu load_leveld.
- **-k** wywołanie skryptu kończącego proces, określonego w pliku konfiguracyjnym. Jeśli skrypt nie został określony, wysłany jest sygnał SIGTERM, a po upływie czasu określonego w pliku konfiguracyjnym w atrybucie termwait, wysyłany jest sygnał SIGKILL(jeśli proces nadal się nie zakończył).
- **-L** wyświetlenie listy monitorowanych procesów.
- **-o** proces zarejestrowany z tą opcją nie może się zdemonizować.

- **-x** jeśli proces jest w stanie *down* , to jest zerowany jego licznik błędów i proces jest restartowany.

  Jeśli proces działa, to opcja ta zeruje jego licznik błędów.

  Jeśli opcja ta użyta zostanie z opcją -k, to proces jest restartowany. Jest to użyteczne polecenie, gdy proces się zawiesił.

Bardziej szczegółowy opis polecenia spawndaemon znajduje się w dokumentacji openssi, oraz na stronie
http://ci-linux.sourceforge.net/spawndaemon.shtml

### 2.2.1 Pliki konfiguracyjne

Oprócz plików wymienionych w opisie keepalive, należy jeszcze umieścić w katalogu */dev/keepalivecfg* pliki konfiguracyjne dla spawndaemon.

Nazwa pliku dla każdego monitorowanego procesu lub grupy procesów musi zaczynać się na *ka_* . Właścicielem pliku musi być root i musi należeć do grupy root. Prawa dostępu do każdego pliku konfiguracyjnego muszą być ustawione na rw- r– r– .

Dla pojedynczych procesów lub demonów używane są pliki w innym formacie niż dla grup procesów lub demonów.

Dla **pojedynczego procesu lub demona** linia w pliku konfiguracyjnym jest w formacie format:
[group_name]:full_path_to_executable:[arg_list]:[termwait]:uid:
gid:[max_errors]:[probation_period]:[minrespawn]:
startup_script:[shutdown_script]:[process_failure_recovery_script]
:[node_failure_recovery_script]:[down_script]:[down_script_policy]

Pola oddzielone są dwukropkami, a ich znaczenie opisane jest poniżej.

- **group_name** Nazwa grupy procesów. Pole jest wymagane, gdy proces lub demon należy do grupy procesów. Pole musi pozostać puste, gdy proces nie należy do grupy. Wartość tego pola musi być taka sama jak wartość pola group_name w pliku konfiguracyjnym dla odpowiedniej grupy. Maksymalna długość nazwy to 16 znaków.
- **full_path_to_executable** Pełna nazwa (wraz ze ścieżką dostępu) pliku wykonywalnego monitorowanego procesu.
- **arg_list** lista argumentów wywołania procesu. Liczba argumentów musi być niewiększa niż dopuszczalna liczba argumentów wywołania procesu lub demona z lini poleceń.
- **termwait** Czas, mierzony w sekundach, przez który może trwać kończenie procesu lub demona. Jeśli ten okres zostanie przekroczony, keepalive wysyła do procesu sygnał SIGKILL. Wartość domyślna to 2.
- **uid** Identyfikator użytkownika, z którego prawami jest uruchamiany proces.
- **gid** Identyfikator grupy, z której prawami jest uruchamiany proces.

- **max_errors** Maksymalna dopuszczalna liczba błędów wykonania procesu, które mogą wystąpić w okresie probation_period . Po przekroczeniu tej liczby, keepalive przestaje wznawiać proces. Błędy muszą wystąpić w okresie określonym przez pole probation_period. Wartością domyślną jest 10.

- **probation_period** Przedział czasu, w którym jeśli wystąpią błędy w liczbie większej niż max_errors, proces nie będzie wznowiony. Jeśli w tym okresie wystąpi więcej niż *max_errors* błędów, to proces przechodzi w stan *down* (w rozumieniu keepalive). Domyślną wartością jest 300 sekund.

- **minrespawn** Czas (w sekundach) po jakim najwcześniej zostanie wznowiony proces. Czasomierz jest zerowany przy starcie procesu. Jeśli przed upłynięciem minrespawn sekund proces ten zakończy się, to nie będzie wznawiany aż do momentu gdy czas ten upłynie. Domyślną wartością jest 0.

- **startup_script** Nazwa skryptu wykorzystywanego przez keepalive do uruchomienia monitorowanego procesu. Skrypt ten jest uruchamiany także w przypadku gdy proces ulegnie awarii i nie ma dla niego określonej wartości w polu *process_failure_recovery_script* . Jeśli awarii ulegnie węzeł klastra OpenSSI, na którym działa monitorowany proces i nie ma określonego *process_failure_recovery_script* ani *node_failure_recovery_script* , to jest wykorzystywany skrypt *startup_script* . Skrypt ten musi znajdować się w katalogu /etc/keepalive.d .

- **shutdown_script** Skrypt uruchamiany gdy użytkownik zakończy proces lub demon z wykorzystaniem polecenia spawndaemon z opcją -k . Jeśli nie ma określonego takiego skryptu, to keepalive wysyła monitorowanemu procesowi sygnał SIGTERM. Jeśli przed upływem czasu *termwait* proces nie zakończy się, wysyłany jest sygnał SIGKILL. Skrypt ten musi znajdować się w katalogu /etc/keepalive.d .

- **process_failure_recovery_script** Skrypt uruchamiany w przypadku awarii monitorowanego procesu. Jest także wykorzystywany w przpadku awarii węzła, na którym działa proces, jeśli nie ma określonego *node_failure_recovery_script* . Skrypt ten musi znajdować się w katalogu /etc/keepalive.d .

- **node_failure_recovery_script** Skrypt wykonywany w przypadku awarii węzła, na którym działa monitorowany proces. Skrypt ten musi znajdować się w katalogu /etc/keepalive.d .

- **down_script** Skrypt wykonywany, gdy monitorowany proces przejdzie w stan *down* w rozumieniu keepalive. Proces przechodzi w ten stan, gdy ulegnie awarii więcej razy niż wartość *max_errors* w okresie *probation_period* sekund. Skrypt ten powinien zwalniać zasoby przydzielone procesowi. Keepalive próbuje uruchomić ten skrypt na węźle, na którym ostatnio działał proces. Jeśli węzeł ten nie jest dostępny i pole *down_script_policy* ma wartość 1, to keepalive próbuje wykonać ten skrypt na innym węźle. Skrypt ten musi znajdować się w katalogu /etc/keepalive.d .

- **down_script_policy** Dopuszczalne wartości to 0 albo 1.

  Jeśli 0 to keepalive nie próbuje wykonania skryptu *down_script* w przypadku awarii procesu i węzła, na którym działał proces.

Jeśli 1 to keepalive próbuje uruchomić skrypt *down_script* w przypadku awarii procesu i węzła, na którym działał proces.

Wartością domyślną jest 1.

Przykładowa linijka w pliku konfiguracyjnym */etc/spawndaemon.d/ka_cron*, dla demona *cron* :
:/usr/sbin/cron:::root:sys::::cron_startup::cron_restart:::

Dla **grupy procesów** linia w pliku konfiguracyjnym jest w formacie format:
<keepalive_group>:group_name
member_file:[wait_time]:[critical]
member_file:[wait_time]:[critical]

Pierwsza linijka musi zaczynać się od ciągu znaków *keepalive_group* ujętego w znaki <>.

Każda nazwa pliku konfiguracyjnego procesu (*member_file* ) należącego do grupy musi zaczynać nowy wiersz.

- **group_name** Nazwa grupy. Nazwa musi być taka sama jak w pliku konfiguracyjnym dla procesu należącego do tej grupy. Długość nazwy nie może przekraczać 16 znaków.
- **member_file** Nazwa pliku konfiguracyjnego procesu należącego do tej grupy.
- **wait_time** Opóźnienie (w sekundach) pomiędzy uruchomieniem kolejnego procesu z grupy. Wartością domyślną jest 0. Poprawne wartości to nieujemne liczby całkowite. Określenie tego atrybutu dla ostatniego procesu na liście procesów grupy nie ma sensu, bo będzie zignorowane.
- **critical** Dopuszczalne wartości to 0 albo 1.

  Jeśli 0, to w przypadku awarii procesu, keepalive restartuje tylko ten proces.

  Jeśli 1, to w przypadku awarii procesu, keepalive restartuje całą grupę.

  Wartością domyślną jest 0.

### 2.2.2 Stany procesów lub demonów

Proces lub demon monitorowany przez keepalive zawsze znajduje się w określonym stanie.

- **start** Proces jest uruchamiany. Przechodzi do stanu **ok** , jeśli zostanie uruchomiony pomyślnie.

  W przeciwnym wypadku, jeśli jego uruchomienie nie powiedzie się na żadnym z dostępnych węzłów, przechodzi do stanu **dead** .

  Jeśli nie ma dostępnych żadnych węzłów w klastrze, proces pozostaje w tym stanie.

- **ok** Proces jest wykonywany.

  Jeśli proces zostanie uszkodzony i został zarejestrowany przy użyciu polecenia spawndaemon bez opcji -o, to przechodzi do stanu **daemonize** .

Jeśli proces zostanie uszkodzony i został zarejestrowany przy użyciu polecenia spawndaemon z opcją -o, lub opuścił węzeł, to przechodzi do stanu **dead** .

Jeśli spawndaemon został użyty do zakończenia procesu, lub gdy krytyczny proces grupy (*critical* 1) uległ awarii, to proces przechodzi do stanu **shutdown** .

Jeśli inny proces należący do tej samej grupy przeszedł do stanu **down** lub jeśli zakończył działanie ze statusem **down** , to ten proces przechodzi do stanu **down** .

- **dead** Proces został uszkodzony i nie jest wykonywany.

  Jeśli maksymalna dopuszczalna liczba błędów (*max_errors* ) została przekroczona, przechodzi do stanu **down** .

  Jeśli maksymalna dopuszczalna liczba błędów (*max_errors* ) nie została przekroczona, przechodzi do stanu **respawn** .

- **down** Proces nie jest wykonywany. Może przejść do stanu **respawn** jeśli zostanie wykonane polecenie spawndaemon z opcją -x.

- **respawn** Proces jest restartowany. Jeśli zostanie zrestartowany pomyślnie, przechodzi do stanu **ok** .

  Jesli restart się nie powiódł na żadnym z dostępnych węzłów w klastrze, przechodzi do stanu **dead** .

  Jeśli nie jest dostępny żaden węzeł w klastrze, proces pozostaje w tym stanie.

- **shutdown** Proces nie jest wykonywany. Proces może przejść do stanu **respawn** , jeśli zostanie wykonane polecenie spawndaemon z opcją -x -k, lub zostaje wyrejestrowany.

- **daemonize** Proces uległ awarii i mógł się zdemonizować. Keepalive wyonuje przywracanie procesu. Jeśli zakończy się powodzeniem, to proces przechodzi do stanu **ok** , w przeciwnym razie przechodzi do stanu **dead** .

# 3 Opis implementacji

Pomysł na rozwiązanie, który **nie został zrealizowany.**

Z powodu ograniczenia wynikającego z możliwości korzystania z demona keepaliwe tylko przez root'a, zdecydowałem się na rozwiązanie polegające na stworzeniu własnego programu, który po uruchomieniu będzie monitorował wskazany proces. Rozwiązanie to nie jest zapewne tak dopracowane jak demon keepalive, niemniej jednak stwarza okazję do zaprezentowania innych dostępnych w OpenSSI mechanizmów. Chodzi tu funkcje **rexecv** , która uruchamia proces na wskazanym węźle klastra. Funkcja ta została wykorzystana w programie monitorującym proces serwera (ha_server). Program ten nazwany został "watchdog".

## 3.1 Funkcjonalność serwera i klienta

Serwer realizuje zlecenia klienta dotyczące stanu jego konta. Do przetestowania serwera używany jest program telnet.

- klient łączy się z serwerem: *tenet adres_serwera nr_portu*
- klient powinien się zautoryzować wpisując *cid xxx* , gdzie *xxx* jest identyfikatorem klienta.
- klient może wpisać wartość liczbową - liczby zmiennoprzecinkowe dodatnie i ujemne. Po wysłaniu takiego komunikatu zostanie wykonana odpowiednia operacja na koncie klienta. Jeśli zostanie wysłana liczba ujemna, to stan konta zostanie zmniejszony o wartość bezwzględną tej liczby. Jeśli dodatnia, to wartość konta wzrośnie. W odpowiedzi serwer wysyła aktualny stan konta klienta.
- klient może zakończyć komunikację wpisując polecenie *end* .

Serwer jest stanowy. Stan konta klienta przechowywany jest w pliku tekstowym. Po przerwaniu i następnie wznowieniu komunikacji, odczytywany jest stan konta z pliku. Po każdej transakcji wpis w pliku jest uaktualniany.

## 3.2 Watchdog

Program monitorujący dostępność procesu serweru komunikuje się z nim poprzez kolejki nazwane. W OpenSSI obiokty IPC są tworzone na węźle, na którym działa proces tworzący je. Dlatego w razie awarii procesu serwera, kolejki FIFO będą nadal dostępne do komunikacji z procesem powołanym przez watchdoga.

Watchdog wykonuje następujące czynności.

- uruchamia serwer ha_server.
- co określony czas wysyła do monitorowanego procesu komunikat z kolejną liczbą całkowitą.
- jeśli nie otrzyma w odpowiedzi od serwera liczby, którą mu wysłał, to będzie oznaczać, że serwer uległ awarii.
- jeśli w odpowiedzi otrzyma liczbę, którą wysłał do serwera, watchdog kontynuuje monitorowanie jego dostępności.
- W przypadku awarii restartuje ha_server wywołując funkcję **rexecv()** .

Czas niedostępności serwera jest równy czasowi potrzebnemu do jego ponownego uruchomienia. Może to trwać ok. 60 sekund w związku z zablokowaniem przez system socketa, na którym działał poprzedni proces serwera.

Wadą tego rozwiązania jest niewątpliwie nieodporność na awarię watchdoga. Kiedy ten proces zostanie uszkodzony, cały serwer utraci właściwość HA. Można poprawić to rozwiązanie stosując większą liczbę procesów watchdoga na różnych węzłach.

Zastosowany algorytm elekcji węzła, na którym ma zostać uruchomiony proces serwera jest bardzo prosty. Po prostu zostaje wybrany pierwszy dostępny węzeł. Dostępność węzłów jest sprawdzana w kolejności ich numeracji od najmniejszego.

Funkcja **clusternode_avail(i)** zwraca wartość dodatnią, gdy węzeł i jest dostępny.

### 3.3 ha_server

Proces ten prowadzi komunikację z klientami poprzez sockety TCP.

Nawiązanie komunikacji następuje w sposób standardowy, poprzez wywołanie kolejno funkcji socket(), bind(), listen(), accept(). Po wykonaniu accept() tworzony jest nowy proces komunikujący się z klientem poprzez socket, którego deskryptor został zwrócony przez accept().

Dialog z klientem został opisany w rozdziale 3.1.

Zanim proces zacznie oczekiwać na połączenia od klientów, uruchamia wątek odpowiadający na pytania watchdoga. Na każdy komunikat od klienta odpowiada komunikatem o takiej samej treści. Treść komunikatów stanowią kolejne liczby całkowite generowane przez watchdoga. Jeśli monitorowany serwer nie odpowie wartością otrzymaną od watchdoga, oznacza to, że uległ awarii.

### 3.4 DRBD

DRBD jest systemem plików, w kórym wykorzystywana jest replikacja. Dzięki temu można go wykorzystać do zapewnienia dostępności zapisywanych na jednym węźle plików, po awarii tego węzła.

Na klastrze w laboratorium DRBD nie jest niestety zainstalowany. Jednakże przystosowanie aplikacji serwera do tego systemu plików nie stwarza żadnych trudności. Wystarczy pliki zapisywać na partycji, o której wiadomo, że jest DRBD.

## Literatura