VMware ESXi,
Microsoft Hyper-V,
VirtualBox, OpenVZ

Marek **Serafin**

Wirtualizacja w praktyce

Helion

WIRTUALIZACJA — TO PROSTE!

Poznaj zalety wirtualizacji i sytuacje, w których sprawdza się ona najlepiej
Naucz się instalować najpopularniejsze rozwiązania wirtualizacyjne
Dowiedz się, jak korzystać z wirtualnych serwerów i jak nimi zarządzać

Redaktor prowadzący: Michał Mrowiec

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Fotografia na okładce została wykorzystana za zgodą Shutterstock.com.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie?virtua
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-3724-9

Printed in Poland.

Spis treści

Przedmowa

Celem książki jest zapoznanie Czytelnika z ideą wirtualizacji oraz przedstawienie kilku praktycznych przykładów z wykorzystaniem najpopularniejszych narzędzi. W poszczególnych rozdziałach opisałem różne implementacje maszyn wirtualnych.

Dla kogo jest ta książka?

Książka adresowana jest głównie do administratorów systemów, którzy nie mieli jeszcze styczności z wirtualizacją lub z pewnych względów jej unikali. Mam nadzieję, że w tych kilku rozdziałach przekonam Cię do wirtualizacji albo przynajmniej do przeprowadzenia własnych testów. Z drugiej strony, jeśli wdrożyłeś już produkcyjnie któreś z komercyjnych rozwiązań, np. VMware ESXi, to ta książka nie jest dla Ciebie — nie dowiesz się z niej prawdopodobnie niczego, czego byś już nie wiedział.

O autorze

Nazywam się Marek Serafin, z wykształcenia jestem inżynierem informatykiem. Pracuję jako administrator sieci w Grupie Wydawniczej Helion SA. Swoją przygodę z administrowaniem sieci zacząłem w połowie lat 90., współtworząc jedną z pierwszych na Śląsku dużych sieci osiedlowych (nieistniejąca już RojcaNet w Radzionkowie).

W chwilach wolnych jeżdżę konno.

Rozdział 11.
OpenVZ

Do tej pory każdy z omawianych programów — czy też produktów wirtualizacyjnych — pozwalał na instalację i uruchomienie w wirtualnym środowisku praktycznie dowolnego systemu operacyjnego (spośród systemów dostępnych dla PC). Niezależnie od tego, czy procesor wspierał sprzętowo wirtualizację, czy nie (pełna wirtualizacja ze wsparciem sprzętowym lub bez), dla systemu gościa dostępny był cały wirtualny komputer PC. Dzięki temu goszczony system „nie wiedział" nawet o fakcie instalacji w wirtualnej maszynie. Takie rozwiązanie jest bardzo wygodne i elastyczne, gdyż nie wymaga żadnej modyfikacji systemu gościa (np. jądra systemu lub sterowników). Wiążę się jednak z pewnym narzutem obliczeniowym dla fizycznego komputera i siłą rzeczy wirtualny komputer jest mniej wydajny od fizycznej maszyny. Niemniej dzięki sprzętowym rozszerzeniom współczesnych procesorów, a także parawirtualizacyjnym sterownikom urządzeń narzut ten jest obecnie niewielki, a wydajność często porównywalna z fizyczną maszyną.

W tym rozdziale omówię inne podejście do tematu — wirtualizację opartą na współdzielonym jądrze systemu. Jak się domyślasz, zaletą tej metody jest bardzo wysoka wydajność, gdyż uruchomione jest tylko jedno jądro, które „obsługuje" wiele systemów. Oczywistą wadą jest ograniczenie w postaci uruchomienia instancji tych samych systemów operacyjnych, a konkretnie systemu Linux. Wirtualne systemy w tym przypadku nazywane są „kontenerami" (ang. *container*) lub VPS-ami (Virtual Private Server). Przy czym istotne jest to, że nie muszą one być wierną kopią macierzystego systemu. W kontenerach możesz, co zostanie zaraz pokazane, używać różnych dystrybucji systemu Linux.

Ten rodzaj wirtualizacji zainteresuje zapewne tych administratorów, którzy wymagają wysokiej wydajności, a jednocześnie separacji systemów, np. do świadczenia usług hostingowych opartych na systemie Linux. OpenVZ dostępny jest za darmo (GPL v.2). W jego rozwoju pomaga firma Parallels, która wykorzystuje go w swym komercyjnym produkcie Virtuozzo.

11.1. Instalacja środowiska w systemie Linux Debian

Opis instalacji przedstawię na podstawie systemu Linux Debian, gdyż szczegółowy opis dla dystrybucji Fedora, RedHat i CentOS można znaleźć na stronie projektu:

http://wiki.openvz.org/Quick_installation

Tak naprawdę cały proces sprowadza się do zainstalowania nowego jądra systemu i paru dodatkowych pakietów. W systemach opartych na dystrybucji RedHat zainstalujesz kilka pakietów RPM, w Debianie natomiast ich odpowiedniki DEB.

Konfiguracja zaś wygląda tak samo lub bardzo podobnie we wszystkich dystrybucjach.

Instalację w Debianie przeprowadzimy z gotowych paczek. W tym celu musimy zainstalować specjalnie przygotowane jądro systemu Linux. Zaloguj się jako `root` i wpisz polecenie:

```
root@debian:/# apt-cache search openvz |grep linux-image
linux-image-2.6-openvz-amd64 - Linux 2.6 for 64-bit PCs (meta-package), OpenVZ support
linux-image-openvz-amd64 - Linux for 64-bit PCs (meta-package), OpenVZ support
linux-image-2.6.32-5-openvz-amd64-dbg - Debugging infos for Linux 2.6.32-5-openvz-amd64
linux-image-2.6.32-5-openvz-amd64 - Linux 2.6.32 for 64-bit PCs, OpenVZ support
```

Skopiuj do schowka nazwę pakietu — w moim przypadku jest to `linux-image-2.6.32-5-openvz-amd64` — a następnie zainstaluj go. W tym celu wpisz polecenie:

```
root@debian:/# apt-get install linux-image-2.6.32-5-openvz-amd64
```

Nie przejmuj się skrótem „AMD" w nazwie pakietu. Jądro działa także na procesorze Intel (AMD określa tu architekturę PC-64bit, a nie producenta procesora). Oprócz jądra systemu zostaną zainstalowane dodatkowe pakiety (m.in. `vzctl`, `vzquota`), służące go kontroli pracy maszyn VPS.

Niestety, nie obejdzie się bez ponownego uruchomienia komputera. Zauważ podczas niego, że zostało dodane nowe — domyślne jądro, z którego system Linux zostanie uruchomiony (w moim przypadku: `Linux 2.6.32-5-openvz-amd64`).

W pierwszej kolejności po uruchomieniu systemu z „nowego" jądra musimy zmodyfikować kilka jego parametrów. W tym celu musimy nanieść zmiany w pliku */etc/sysctl.conf* (wszystkie dystrybucje) — patrz listing 11.1.

Listing 11.1. *Zawartość pliku /etc/sysctl.conf*

```
# On Hardware Node we generally need
# packet forwarding enabled and proxy arp disabled
net.ipv4.ip_forward = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.default.proxy_arp = 0
```

```
# Enables source route verification
net.ipv4.conf.all.rp_filter = 1

# Enables the magic-sysrq key
kernel.sysrq = 1

# We do not want all our interfaces to send redirects
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
```

Jak widzisz, chodzi głównie o załączenie przekazywania pakietów (załączenie routingu) i wyłączenie Proxy ARP.

Gdybyś chciał przekleić sobie wszystkie potrzebne linie pliku *sysctl.conf* (aby uniknąć literówki), wejdź na stronę internetową:

http://wiki.openvz.org/Quick_installation#Configuring

Po naniesieniu zmian w pliku musisz je „aktywować", wpisując polecenie (jednorazowo):

```
root@debian:~# sysctl -p
```

Tyle, jeśli chodzi o konfigurację środowiska. Dla pewności możesz się upewnić, czy po wpisaniu komendy:

```
lsmod | grep vz
```

zobaczysz listę załadowanych modułów jądra, takich jak: `vzethdev`, `vznetdev`, `vzrst`, `vzcpt`, `vzmon`, `vzdquota`, `vzdev`, `vzevent`.

11.2. Tworzenie nowego VPS-a

W tym momencie powinniśmy utworzyć nową maszynę, uruchomić ją i zainstalować system z nośnika. W przypadku OpenVZ sprawa wygląda jednak trochę inaczej. Zamiast instalować cały system z nośnika, instalujemy tylko jego „szablon" (ang. *template*). Odpowiednie szablony są dostępne na stronie projektu — każdy dla osobnej dystrybucji.

Załóżmy, że drugim (wirtualnym) systemem ma być 64-bitowy CentOS. Wejdź zatem na poniższą stronę internetową:

http://download.openvz.org/template/precreated/

i odszukaj paczkę odpowiadającą systemowi, który chcesz zainstalować — w opisywanym przypadku będzie to plik o nazwie *centos-6-x86_64.tar.gz*. Pobierz plik i umieść go w katalogu */var/lib/vz/template/cache* systemu macierzystego (katalog powinien już istnieć).

```
root@debian:~# cd /var/lib/vz/template/cache && wget
http://download.openvz.org/template/precreated/centos-6-x86_64.tar.gz
```

Następnie wpisz polecenie:

```
vzctl create 101 --ostemplate centos-6-x86_64
```

gdzie `101` to numeryczny identyfikator systemu wirtualnego, a `centos-6-x86_64` to nazwa szablonu. W wyniku powinieneś po paru sekundach zobaczyć komunikat informujący o utworzeniu przestrzeni dla nowego systemu (kontenera) — patrz listing 11.2.

Listing 11.2. *Tworzenie nowego kontenera*

```
root@debian:~# vzctl create 101 --ostemplate centos-6-x86_64
Creating container private area (centos-6-x86_64)
Performing postcreate actions
Container private area was created
```

Następnie nadaj nazwę systemowi. W tym celu wpisz polecenie:

```
vzctl set 101 --hostname centos64 --save
```

Przypisz także systemowi adres IP:

```
vzctl set 101 --ipadd 10.6.8.114 -save
```

Adres IP podajemy bez maski podsieci, nawet gdy ta jest „nietypowa" dla danej klasy adresowej. Ewentualne problemy z routingiem rozwiążemy później. Po prostu parametr `--ipadd` nie rozpoznaje parametru maski podsieci.

Możesz zdefiniować od razu serwer DNS. W tym celu wpisz polecenie:

```
vzctl set 101 --nameserver 1.2.3.4 -save
```

Jeśli chcesz, aby „wirtualny" system uruchamiany był automatycznie wraz ze startem macierzystego systemu, wpisz polecenie:

```
vzctl set 101 --onboot yes -save
```

11.3. Uruchomienie systemu

Nadszedł czas, aby uruchomić nasz wirtualny system (VPS). Wpisz więc polecenie:

```
vzctl start 101
```

Na ekranie powinieneś zobaczyć komunikaty takie, jak na listingu 11.3.

Listing 11.3. *Uruchomienie wirtualnego systemu o ID 101*

```
root@debian:~# vzctl start 101
Starting container ...
Container is mounted
Adding IP address(es): 10.6.8.114
Setting CPU units: 1000
Set hostname: centos64
File resolv.conf was modified
Container start in progress...
```

Aby „połączyć" się z naszym wirtualnym systemem o identyfikatorze 101, skorzystamy z programu `vzctl`. Wpisz polecenie `vzctl enter 101`. Spójrz na listing 11.4.

Listing 11.4. *Podłączanie się do wirtualnego systemu*

```
root@debian:~# vzctl enter 101
entered into CT 101
root@centos64:/#
root@centos64:/# ping onet.pl
PING onet.pl (213.180.146.27) 56(84) bytes of data.
64 bytes from s4.m1r2.onet.pl (213.180.146.27): icmp_req=1 ttl=51 time=10.8 ms
64 bytes from s4.m1r2.onet.pl (213.180.146.27): icmp_req=2 ttl=51 time=10.6 ms
```

Zwróć uwagę (patrz listing 11.4) na zmianę nazwy systemu z *debian* na *centos64* (tzw. „login promt"). Nastąpiło przełączenie konsoli — do systemu wirtualnego. Jak widać na listingu, w systemie sieć działa poprawnie. Aby „wyjść" z wirtualnego systemu i powrócić do macierzystego, naciśnij kombinację klawiszy *Ctrl+D*. Przedtem jednak warto zdefiniować hasło użytkownika `root`, używając standardowego polecenia `passwd`.

Zauważ, że domyślnie w nowym systemie działa już usługa serwera SSH oraz serwer http Apache 2.

W przeciwieństwie do wcześniej omawianych rozwiązań wirtualizacyjnych system plików wirtualnych systemów nie jest przechowywany w jednym pliku (obrazie dysku), lecz dostępny bezpośrednio w strukturze katalogów systemu macierzystego. Domyślnie pliki VPS-ów przechowywane są w katalogu:

```
/var/lib/vz/private/{VPS_ID}/
```

Przy czym w momencie uruchomienia maszyny cała struktura pojawia się także w katalogu:

```
/var/lib/vz/root/{VPS_ID}/
```

Dzieje się tak dlatego, że w momencie uruchomienia VPS-a tworzone są „twarde linki" do plików w katalogu */var/lib/vz/private/*. Idea jest taka, żeby nie ruszać ręcznie plików w katalogu *private*. Jeśli chcesz zmodyfikować zawartość systemu wirtualnego z poziomu systemu macierzystego, należy zmodyfikować zawartość katalogu */var/lib/vz/root/{VPS_ID}/*. Skoro jesteśmy przy omawianiu systemu plików, wspomnę o możliwości „współdzielenia" katalogów pomiędzy systemem nadrzędnym a VPS-em. Przykładowo gdybyś chciał „podmontować" katalog */data/shared* systemu macierzystego w kontenerze o identyfikatorze 101, należy w systemie macierzystym wpisać polecenie:

```
mount --bind /data/shared/ /var/lib/vz/root/101/mnt/shared
```

W tym momencie w kontenerze 101 zobaczysz zamontowany zasób w katalogu */mnt/shared* (patrz listing 11.5).

Listing 11.5. *Zamontowany katalog gospodarza w systemie wirtualnym*

```
[root@centos64 /]# mount | grep shared
/dev/disk/by-uuid/a670d536-c84a-417c-bc47-e6e1b36c93ce on /mnt/shared
```

Oczywiście równie dobrze mógłbyś wykorzystać w tym celu protokół NFS, ale wiązałoby się to z pewnym spadkiem wydajności wynikającym z narzutu obsługi NFS.

Kontrolowanie stanu pracy kontenerów

Zamykanie wirtualnej maszyny

```
vzctl stop ID_maszyny
```

Restart maszyny

```
vzctl restart ID_maszyny
```

Usuwanie maszyny

```
vzctl destroy ID_maszyny
```

Listę dostępnych maszyn (kontenerów) możesz sprawdzić za pomocą polecenia `vzlist -a`.

Pomijając parametr `-a`, uzyskasz listę aktualnie uruchomionych systemów. Wpisz polecenie `vzlist -L`, aby uzyskać listę dostępnych argumentów, które możesz pobrać przy użyciu polecenia `vzlist -o`. Przykładowo aby pobrać status (`running` lub `stopped`) konkretnej maszyny, wpisz: `vzlist 101 -Ho status`. W wyniku otrzymasz sam ciąg znaków `running` lub `stopped`, bez dodatkowych informacji tekstowych (przydatne podczas tworzenia skryptów shellowych).

Uruchamianie programu w maszynie

Aby uruchomić program bez „wchodzenia" do maszyny, wystarczy wpisać polecenie:

```
vzctl exec 101 ps aux
```

gdzie `101` to oczywiście identyfikator kontenera. Polecenie to możesz wykorzystać podczas tworzenia różnego rodzaju skryptów automatyzujących pewne zadania.

Pliki konfiguracyjne

/etc/vz/vz.conf — globalny plik konfiguracyjny

/etc/vz/conf/.conf* — pliki konfiguracyjne kontenerów

W plikach konfiguracyjnych każdego z kontenerów możesz zdefiniować przede wszystkim ograniczenia zasobów wirtualnego środowiska, takie jak dostępna pamięć, liczba możliwych do uruchomienia procesów czy ograniczenia pojemności dysku. Składnia pliku konfiguracyjnego najczęściej przybiera postać `PARAMETR=bariera:limit`, co przez analogię do limitów nakładanych na dysk można porównać do soft quota i hard quota.

Składnię pliku konfiguracyjnego możesz sprawdzić, używając polecenia `vzcfgvalidate`. Poniżej przedstawiam opis najważniejszych parametrów, które wpływają na dostępne zasoby kontenera VPS:

- `KMEMSIZE` — limit wielkości pamięci jądra wyrażonej w bajtach. Na wielkość zaalokowanej pamięci jądra wpływa liczba uruchomionych procesów, gdyż każdy utworzony proces implikuje alokację 40 – 60 KB (przeciętnie) pamięci jądra. Weźmy dla przykładu domyślną wartość `KMEMSIZE="14372700:14790164"`. Zwróć uwagę na różnicę pomiędzy barierą a limitem. Dzięki istnieniu bariery jądro „zawczasu" nie pozwoli na utworzenie nowego procesu (gdy zajętość osiągnie wartość bariery). Gdyby wyrównać te parametry, mogłoby dojść do sytuacji, w której jądro musiałoby „ubijać" istniejące procesy uruchomione w kontenerze. Nie należy mylić pamięci jądra z pamięcią dostępną dla aplikacji (patrz poniżej).
- `PRIVVMPAGES` — maksymalna ilość pamięci RAM dostępna dla danego VPS. Wielkość wyrażona jest w stronach (ang. *pages*), dlatego wartość stron należy pomnożyć przez 4096, aby uzyskać wynik w bajtach (lub odwrotnie — oczekiwany przydział w bajtach podzielić przez 4096, aby uzyskać liczbę stron). Przykładowo jeśli chcesz, aby dany kontener nie przekroczył 1 GB RAM-u, musisz przypisać barierę na 262 144 strony. Przy większej liczbie kontenerów należy to dokładnie przemyśleć. Zakładając, że nie każdy z kontenerów wykorzystuje w danej chwili dostępny dla niego RAM w 100%, możesz sumarycznie zadeklarować więcej pamięci RAM, niż posiada komputer. W praktyce wszystko zależy od rzeczywistego obciążenia VPS-ów. Twórcy OpenVZ zalecają, aby poszczególne kontenery nie mogły zaalokować więcej niż 0,6 całkowitej pamięci RAM. Tzn. dla każdego z kontenerów spełniony winien być warunek:

  ```
  PRIVVMPAGES * 4096 < ilość 0.6 * RAM
  ```

  W przeciwnym razie już pojedynczy kontener mógłby doprowadzić do „swapowania", co oczywiście wpłynie negatywnie na wydajność całego systemu.
- `VMGUARPAGES` — gwarantowana wielkość pamięci RAM dostępnej dla danego VPS-a; także wyrażona w stronach (ang. *pages*). Bariera `VMGUARPAGES` musi być mniejsza od bariery `PRIVVMPAGES`, która jest nieprzekraczalną granicą dla VPS-a. Z technicznych względów składnia dla `VMGUARPAGES` powinna mieć wartość:

  ```
  VMGUARPAGES="BARIERA:9223372036854775807"
  ```

  tzn. limit powinien być ustawiany „na sztywno" — na maksymalną wartość. Jak wspominałem, i tak nie ma możliwości, aby VPS przekroczył wartości bariery `PRIVVMPAGES`.
- `NUMPROC` — maksymalna liczba możliwych do utworzenia procesów w kontenerze. Jak wspomniałem, na liczbę możliwych do uruchomienia procesów ma także wpływ wielkość dostępnej pamięci jądra (patrz `KMEMSIZE`). Innymi słowy, przydzielenie zbyt małej pamięci jądra ograniczy też liczbę możliwych do uruchomienia procesów.

W praktyce limity pamięci działają tak, że dopóki ilość zaalokowanej przez kontener pamięci nie przekracza bariery `VMGUARPAGES`, aplikacje uruchomione w kontenerze mogą dalej alokować pamięć (i mają gwarancję jej przydzielenia). Jeśli bieżące zużycie

RAM-u przez kontener przekracza barierę gwarantowaną, ale nie przekracza bariery maksymalnego zużycia (PRIVVMPAGES), operacja alokacji może, ale nie musi się powieść. Zależy to od ilości dostępnej w danym momencie pamięci wirtualnej całego komputera (RAM + SWAP), czyli od „aktywności" systemu macierzystego i pozostałych kontenerów. Parametr VMGUARPAGES jest o tyle istotny, że jest odpowiednikiem parametru CIR znanego z terminologii sieciowej, czyli pasma gwarantowanego. W szczególności dostawcy usług hostingowych winni zwrócić na niego uwagę.

Twórcy OpenVZ zalecają, aby wielkość systemowego „swapa" stanowiła mniej więcej dwukrotność fizycznej wielkości pamięci RAM. Nie zaleca się przesadzać ze swapem, np. mającym wielkość odpowiadającą 10-krotności pamięci RAM, gdyż wpłynie to negatywnie na wydajność systemu.

Skoro już jesteśmy przy ograniczaniu zasobów, muszę wspomnieć o specjalnym pliku */proc/user_beancounters*, który zawiera informacje oraz statystyki (liczniki) związane z bieżącym działaniem uruchomionych kontenerów. Jeśli przeglądasz plik z poziomu wirtualnego systemu, zobaczysz statystyki tylko dla tego kontenera. Z poziomu systemu macierzystego dostępne są statystyki dotyczące wszystkich systemów. W kolumnie uid odnajdziesz identyfikator każdego z kontenerów. W następnych kolumnach możesz znaleźć bieżące oraz maksymalnie użyte zasoby (dotyczące kmem, privvmpages, numproc, physpages itd.), a także informacje o bieżących limitach. Szczególnie istotna jest kolumna failcnt, w której możesz odczytać, ile razy żądana operacja nie powiodła się (np. alokacja pamięci, utworzenie nowego procesu czy gniazda sieciowego).

Zróbmy dla przykładu prosty test. Ograniczmy parametr KMEMSIZE maszyny o identyfikatorze 101 do wartości 3 593 174 (około 3,5 MB przydzielonej pamięci jądra). Następnie napiszmy prosty skrypt shellowy, który w pętli spróbuje uruchomić 300 procesów (np. 300 razy uruchomić program top). Jak wspomniałem, utworzenie nowego procesu alokuje pewną ilość pamięci jądra, stąd ograniczenie tej pamięci powinno skutkować niemożnością utworzenia nieskończonej liczby procesów. Ustaw wspomniane ograniczenie dla maszyny o numerze 101 i uruchom ją ponownie. Następnie przełącz się do konsoli wspomnianej maszyny i utwórz w niej skrypt pokazany na listingu 11.6.

Listing 11.6. *Skrypt, którego zadaniem jest uruchomienie w tle stu procesów programu top*

```
#!/bin/sh
for i in `seq 1 100` ; do
    echo $i && top&
done
```

Następnie uruchom skrypt i zobacz, co się stanie (patrz listing 11.7).

Listing 11.7. *Próba uruchomienia skryptu*

```
[root@centos64 /]# ./skr1.sh
1
2
3
5
6
```

```
./skr1.sh: fork: Cannot allocate memory
./skr1.sh: fork: Cannot allocate memory
./skr1.sh: fork: Cannot allocate memory
```

Jak widzisz, nie udało nam się wywołać stu instancji programu `top`. Podejrzyj teraz zawartość pliku */proc/user_beancounters*. Zwróć uwagę na wartość kolumny `failcnt` przy pozycji `kmemsize`. Wróć jeszcze raz do edycji pliku konfiguracyjnego kontenera 101 i przywróć poprzednią wartość parametru `KMEMSIZE`, a jednocześnie zmniejsz wartość bariery parametru `NUMPROC` (liczba procesów). Uruchom ponownie maszynę o identyfikatorze 101 i spróbuj znów wykonać ten sam skrypt. Jak się domyślasz, skrypt ponownie przestanie działać (patrz listing 11.8). Podejrzyj teraz zawartość pliku */proc/user_beancounters* i zwróć tym razem uwagę na zawartość kolumny `failcnt` przy pozycji `numproc`.

Listing 11.8. *Ponowna próba uruchomienia skryptu przy obniżonej wartości bariery NUMPROC*

```
        top: failed tty get
        top: failed tty get
        top: failed tty get
        top: failed tty get

./skr1.sh: fork: Cannot allocate memory
./skr1.sh: fork: Cannot allocate memory
./skr1.sh: fork: Cannot allocate memory
./skr1.sh: fork: Cannot allocate memory
```

Skoro już mowa o „procesach", warto wspomnieć, iż z poziomu systemu gospodarza widoczne są wszystkie procesy kontenerów. Oznacza to, że jeśli w systemie gospodarza uruchomisz program `top` lub użyjesz polecenia `ps aux`, w wyniku zobaczysz także procesy kontenerów. Nic w tym dziwnego, wszak systemy te wykorzystują to samo jądro. Mając uprawnienia użytkownika `root`, możesz normalnie wymuszać zakończenie tych procesów. Przykładowy wycinek polecenia `ps` przedstawiłem na listingu 11.9. Zwróć uwagę na identyfikator kontenera widniejący na liście.

Listing 11.9. *Procesy kontenera widziane z poziomu systemu gospodarza*

```
root@debian:~# ps auxww|grep make
root     29250  0.0  0.0 101168  1116 /var/lib/vz/root/101/dev/pts/0 S+ 15:10
↪0:00 make -e PLATFORM=linux-x86_64 PROCESSOR= CC=gcc
```

Ręczne przeglądanie pliku */proc/user_beancounters* nie należy do wygodnych, dlatego napisano kilka skryptów ułatwiających to zadanie. Jednym z nich jest *userbeans.sh*, który możesz pobrać ze strony: *http://amos.freeshell.org/userbeans.sh*

Pobierz skrypt, nadaj mu uprawnienia wykonywalności i uruchom w wirtualnym środowisku. W tym celu wpisz polecenie `./userbeans.sh +a` (patrz listing 11.10).

Listing 11.10. *Wynik działania skryptu userbeans.sh +a*

```
[root@centos64 skrypty]# ./userbeans.sh +a
------------------------------------------------
Processing UBC version 2.5 for VEID: 101
```

```
Thu Oct  6 17:12:01 MSD 2011       centos64
 17:12:01 up 1 min,  0 users,  load average: 0.00, 0.00, 0.00
------------------------------------------------
 272 MB Allocation Limit [privvmpages limit]
****** only high value processes have a chance in this range
****** having this safety range is important to permit critical processes
 256 MB Allocation Barrier [privvmpages barrier]
****** allocation requests in this range have a chance
 132 MB Allocation Guarantee [vmguarpages barrier]
****** allocation will succeed in this range
 102 MB Memory Guarantee [oomguarpages barrier]
  39 MB (  55 MB Max) page memory allocated [privvmpages held]
  10 MB (  10 MB Max) memory + swap used [oomguarpages held]
  10 MB (  10 MB Max) page memory used [physpages held]
  14 MB ( 14443 KB) kernel memory limit [kmemsize limit]
****** a safety range here, between limit and barrier, is important
  13 MB ( 14035 KB) kernel memory barrier [kmemsize barrier]
   2 MB (  2581 KB) kernel memory used [kmemsize held]
   0 MB (   103 KB) buffer memory used [*buf held]
------------------------------------------------
 Used : Max_Used : Limit    for Other Resources
   397     428    9312   numfile
     3       3     206   numflock
    14      14     128   numiptent
    26      28     360   numothersock
    18      19      80   numproc
     1       1      16   numpty
     0       2     256   numsiginfo
     3       4     360   numtcpsock
------------------------------------------------
```

Dodając do skryptu odpowiednie przełączniki, możesz ograniczyć wyświetlany wynik do pożądanych parametrów (pamięć jądra, liczba procesów itd.). Innym użytecznym poleceniem służącym do podglądu bieżącego użycia zasobów jest `vzcalc`.

Limitowanie zasobów CPU

Oprócz ograniczenia pamięci przyda Ci się zapewne ustawienie limitów czasu procesora (procesorów). Z limitem CPU związane są dwa polecenia: `cpulimit` i `cpuunits`. Pierwsze z nich określa procentowy przydział czasu procesora dla danego kontenera. Przykładowo jeśli chcesz, aby dany kontener nie przekroczył 25% czasu procesora, musisz wpisać poniższe polecenie:

```
vzctl set 101 --cpulimit 25 --save
```

Zauważ, że limit dotyczy pojedynczego rdzenia. Jeśli dysponujesz maszyną wieloprocesorową, należy wziąć to pod uwagę. Przykładowo w przypadku 4-rdzeniowej maszyny ustawienie limitu na 100% oznaczać będzie, że kontener będzie miał do dyspozycji jeden rdzeń, na 200% — dwa rdzenie itd. Zwróć uwagę, że w przypadku limitowania procentowego nawet jeśli komputer „nudzi się”, VPS nie może „przeskoczyć” swojego limitu przydziału.

Drugą metodą jest ograniczenie przydziału jednostek CPU. Łatwiej wyjaśni to poniższy przykład:

```
root@debian:~#vzctl set 101 --cpuunits 1000 --save
root@debian:~#vzctl set 102 --cpuunits 2000 --save
root@debian:~#vzctl set 103 --cpuunits 3000 -save
```

Jeśli teraz w każdym z kontenerów uruchomisz operacje mocno obciążające procesor (jak np. kompilacja kernela), to przydział czasu procesora będzie rozkładał się następująco: maszyna 103 będzie miała do dyspozycji trzy razy więcej czasu niż maszyna 101, a maszyna 102 dwa razy więcej niż 101. Innymi słowy, maszyna 101 otrzyma 1/6 czasu procesora, maszyna 102 — 1/3, a maszyna 103 połowę czasu procesora.

Limitowanie dysku (disk quota)

Aby ustawić kontenerowi 101 ograniczenie dysku, np. 30 GB bariera (tzw. soft quota) i 33 GB limit (tzw. hard quota), wpisz polecenie:

```
vzctl set 101 --diskspace 30G:33G -save
```

Następnie zaloguj się do kontenera i sprawdź za pomocą polecenia `df -h` dostępną pojemność. Gdyby zaszła potrzeba zwiększenia liczby indów, możesz to zrobić, używając polecenia:

```
vzctl set 101 --diskinodes  <bariera>:<limit>
```

Limitowanie parametrów sieciowych

W pliku konfiguracyjnym danego kontenera możesz zdefiniować także ograniczenia związane z obsługą TCP/IP. Wartość limitów przekłada się na zużycie zasobów komputera (im więcej otwartych połączeń, tym więcej zaalokowanej pamięci i więcej pracy dla procesora).

- `NUMTCPSOCK` — maksymalna liczba utworzonych gniazd protokołu TCP (tylko TCP). W tym przypadku bariera powinna być równa limitowi.
- `NUMOTHERSOCK` — jak wyżej, tylko dla wszystkich pozostałych gniazd — UDP, gniazd Unix. Analogicznie jak powyżej, wartości bariery i limitu winny być równe.

Więcej na temat ograniczania zasobów kontenerów możesz przeczytać na stronie projektu OpenVZ:

http://wiki.openvz.org/UBC_primary_parameters

http://wiki.openvz.org/UBC_systemwide_configuration

Filtrowanie pakietów w kontenerach (iptables)

Systemy wirtualne VPS mogą używać swojego firewalla (`iptables`) wraz z wszystkimi dostępnymi modułami. Domyślnie dostępne są moduły `iptables` wymienione w globalnym pliku konfiguracyjnym. Jeśli chciałbyś dodać jakiś niestandardowy moduł,

musisz dopisać go do listy modułów w pliku */etc/vz/vz.conf* (patrz listing 11.11). Oczywiście najpierw moduł ten musi obsługiwać samo jądro.

Listing 11.11. *Domyślne moduły iptables*

```
IPTABLES="ipt_REJECT ipt_tos ipt_limit ipt_multiport iptable_filter iptable_mangle
➥ipt_TCPMSS ipt_tcpmss ipt_ttl ipt_length"
```

Składnia programu `iptables` jest taka sama jak w standardowym systemie, więc jej opis pominę.

Archiwizowanie kontenerów

Do tworzenia backupów służy program `vzdump`. Prawdopodobnie będziesz go musiał dodatkowo zainstalować z paczek. W przypadku Debiana wpisz polecenie:

```
apt-get install vzdump
```

W pierwszej kolejności utworzymy pełne archiwum *.tar* kontenera. W tym celu wpisz po prostu polecenie `vzdump <ID_kontenera>` (patrz listing 11.12).

Listing 11.12. *Tworzenie archiwum kontenera*

```
root@debian:~# vzdump 101
INFO: starting new backup job: vzdump 101
INFO: Starting Backup of VM 101 (openvz)
INFO: CTID 101 exist mounted running
INFO: status = CTID 101 exist mounted running
INFO: mode failure - unable to detect lvm volume group
INFO: trying 'suspend' mode instead
INFO: backup mode: suspend
INFO: bandwidth limit: 10240 KB/s
INFO: starting first sync /var/lib/vz/private/101/ to /var/lib/vz/dump/vzdump-
➥openvz-101-2011_10_08-06_35_52.tmp
```

W wyniku otrzymasz plik o nazwie *vzdump-openvz-{ID}-{bieżąca-data}.tar*, np. *vzdump-openvz-101-2011_10_08-06_35_52.tar*. Aby przywrócić tak utworzone archiwum *.tar*, w pierwszej kolejności należy usunąć kontener, a następnie przywrócić archiwum i uruchomić kontener (patrz listing 11.13).

Listing 11.13. *Przywracanie kontenera z archiwum*

```
root@debian:~# vzctl stop 102
Stopping container ...
Container was stopped
Container is unmounted
root@debian:~# vzctl destroy 102
Destroying container private area: /var/lib/vz/private/102
Container private area was destroyed
```

```
root@debian:~# vzrestore /var/lib/vz/dump/vzdump-openvz-102-2011_10_08-07_10_28.tar 102
INFO: restore openvz backup '/var/lib/vz/dump/vzdump-openvz-102-2011_10_08-
↪07_10_28.tar' using ID 102
INFO: extracting archive '/var/lib/vz/dump/vzdump-openvz-102-2011_10_08-07_10_28.tar'
INFO: Total bytes read: 758855680 (724MiB, 294MiB/s)
INFO: extracting configuration to '/etc/vz/conf/102.conf'
INFO: restore openvz backup '/var/lib/vz/dump/vzdump-openvz-102-2011_10_08-
↪07_10_28.tar' successful
root@debian:~# vzctl start 102
Starting container ...
Initializing quota ...
Container is mounted
Adding IP address(es): 10.6.8.115
Setting CPU units: 5000
Set hostname: centos2
File resolv.conf was modified
Container start in progress...
```

Podsumowanie

Podstawową zaletą OpenVZ jest wysoka wydajność wirtualnych kontenerów. Brak narzutu warstwy wirtualizacji sprzętu czy konieczności uruchamiania dodatkowego jądra powoduje, że osiągamy wydajność praktycznie tę samą co macierzysty system, przy jednoczesnym zachowaniu separacji środowisk. Polecam Ci przeprowadzenie własnych testów, np. porównanie czasu kompilacji programów w kontenerach i w macierzystym systemie. Dodatkową zaletą są duże możliwości nakładania ograniczeń (limitów) na poszczególne kontenery, a także łatwość robienia backupów (np. poprzez zwykły rsync). Dzięki wymienionym właściwościom OpenVZ jest często używany przez dostawców usług hostingowych lub firmy, które potrzebują uruchomić wiele odseparowanych od siebie systemów Linux. Jeżeli masz w firmie działający serwer linuksowy i chciałbyś uruchomić kilka dodatkowych „wirtualek" linuksowych — także w tym przypadku wybór OpenVZ jest godny polecenia.

Skorowidz

I

J

K

L

M

N

O

P

R

S

Ś

T

U

V

W

X

Z

Wirtualizacja w praktyce

Temat wirtualizacji z roku na rok staje się w świecie IT coraz popularniejszy. Nie ma w tym nic dziwnego — wirtualizacja serwerów i zasobów pozwala używać jednocześnie kilku różnych systemów operacyjnych na tym samym komputerze, co znacznie zwiększa możliwości wykorzystania posiadanego sprzętu, obniża koszty związane z jego utrzymaniem i poprawia elastyczność konfiguracji zasobów. Na tym nie koniec: wirtualizacja zdecydowanie ułatwia administratorom ich codzienną pracę — zmniejsza podatność systemów na awarie i przeciążenia, upraszcza tworzenie kopii zapasowych i przywracanie konfiguracji oraz umożliwia wykonywanie bezpiecznych testów nowych rozwiązań.

Dlaczego więc wielu administratorów systemów informatycznych nadal jej unika? Zwykle boją się oni kłopotów związanych z instalacją i konfiguracją maszyn wirtualnych; na przeszkodzie stoją też często ich przyzwyczajenia i niechęć do nowych rozwiązań. O tym, że wirtualizacja wcale nie musi być problematyczna i może znacznie ułatwić pracę, przekona Cię lektura książki *Wirtualizacja w praktyce*. Poznasz korzyści płynące z zastosowania najbardziej popularnych rozwiązań, sposoby instalacji odpowiedniego oprogramowania oraz tworzenia i konfiguracji maszyn wirtualnych, metody instalowania systemów operacyjnych gości, a także najważniejsze działania związane z używaniem i konserwowaniem wirtualnych zasobów. Przekonasz się, że wirtualizacja może być naprawdę prosta!

Wirtualizacja w praktyce to kolejna pozycja autora bestsellerowej, świetnie ocenianej przez fachowców i doskonale przyjętej w środowisku informatycznym książki *Sieci VPN. Zdalna praca i bezpieczeństwo danych.*

W książce znajdziesz:

- wykaz korzyści płynących z wirtualizacji systemów,
- opis rodzajów wirtualizacji i ich zastosowanie,
- przegląd najważniejszych rozwiązań w tym zakresie,
- informacje o instalacji i konfiguracji środowisk VirtualBox, VM Ware Server, VM Ware ESX, Microsoft Virtual PC, Microsoft Hyper-V, Citrix Xenserver i OpenVZ,
- sposoby tworzenia i konfiguracji maszyn wirtualnych oraz instalacji systemów gości,
- metody zarządzania systemami i zasobami, a także tworzenia i korzystania z kopii bezpieczeństwa.

Poznaj sposoby wirtualizacji w praktyce!

ISBN: 978-83-246-3724-9
9 788324 637249
Cena: 54,90 zł