

Baixo custo

O projeto VServer oferece uma alternativa de virtualização segura e altamente eficiente.

por **Wilhelm Meier**
e **Torsten Kockler**



Hannah Boettcher - www.sxc.hu

Sistemas de para-virtualização como o *Xen* habitam um canto popular da paisagem da virtualização, mas outras tecnologias também estão crescendo. O projeto *Linux-VServer* representa uma abordagem diferente para a virtualização. O VServer [1] e projetos semelhantes utilizam uma técnica conhecida como *isolamento no nível do kernel*. Essa técnica oferece ao sistema virtual um conjunto isolado de recursos do sistema físico. Outros esquemas de virtualização geralmente exigem um kernel separado e espaços de memória e disco separados para cada sistema virtual. Com o VServer, por outro lado, a virtualização ocorre na interface do processo virtual com o kernel. Todos os sistemas virtuais compartilham o mesmo kernel, e processos virtuais realmente rodam como processos normais do hospedeiro.

Do ponto de vista da segurança, o VServer é semelhante a um mecanismo de engaiolamento. Os processos dentro do ambiente virtual ficam isolados do resto do sistema, de forma que, se um intruso explorar uma vulnerabilidade para ganhar acesso, ele não conseguirá sair da “gaiola” criada pelo isolamento no nível do sistema.

A idéia por trás do VServer é oferecer um sistema que faça uso ótimo dos recursos do hospedeiro e, portanto, exija o mínimo possível de processamento para gerenciar o ambiente virtual. O VServer e outros sistemas de isolamento do kernel oferecem alto desempenho e são bastante escaláveis para grandes números de unidades virtuais.

Devido a sua grande eficiência, escalabilidade e segurança do tipo gaiola embutida,

o VServer geralmente é usado em serviços de hospedagem de websites, embora seus benefícios o levem a ser aplicado muitas vezes nas redes corporativas. Este artigo descreve como configurar um sistema VServer. Também mostraremos uma solução alternativa de virtualização semelhante ao VServer, chamada OpenVZ [2].

Servos virtuais

O VServer funciona como parte do kernel Linux. Infelizmente, os benefícios do VServer atualmente ainda não estão disponíveis no kernel “oficial”. Apesar de o kernel padrão ter todos os ingredientes necessários para suportar o particionamento, também é necessário aplicar vários *patches* do projeto Linux VServer.

A maioria das distribuições oferecem pacotes com os patches necessários, eliminando a necessidade dos usuários de aplicá-los ao kernel pessoalmente. Se você preferir evitar a recompilação de um novo kernel, há kernels pré-compilados nas páginas dos projetos Linux VServer [1] e OpenVZ [2].

Gentoo como hospedeiro

Os exemplos deste artigo são baseados em um sistema *Gentoo* como hospedeiro. Os conceitos são semelhantes, apesar de alguns detalhes talvez serem diferentes em outras distribuições.

O pacote `vserver-sources` do Gentoo traz os fontes do kernel com os patches

apropriados, enquanto o `util-vserver` contém o pacote de ferramentas de usuário para o Gentoo. Note que há outro pacote não relacionado, mas com um nome parecido: `vserver-utils` [3]. Esse pacote ainda está sendo intensamente desenvolvido e alterado, e só funcionará com versões de desenvolvimento do VServer.

Para instalar os pacotes no Gentoo, simplesmente faça:

```
emerge vserver-sources util-vserver  
rc-update add vserver default
```

O último comando inicia os servidores virtuais na inicialização. Depois, você pode compilar e instalar o kernel VServer para o hospedeiro, usando o `genkernel`, por exemplo. Em seguida, modifique o carregador de inicialização e reinicie a máquina. Mas antes disso, certifique-se de introduzir o novo kernel ao script auxiliar do VServer, rodando `echo 'kernel.vshelper=/usr/lib/util-vserver/vshelper' >> /etc/sysctl.conf`. Esse script é usado para fazer `halt` e `reboot` no servidor virtual: afinal, você quer ligar e desligar apenas o sistema virtual, enquanto o hospedeiro deve continuar rodando.

Gentoo como hóspede

É fácil configurar um ambiente VServer. Comece especificando o nome e a ID de contexto – obviamente, esses valores têm que ser únicos. Depois, especifique uma interface de rede para o servidor virtual,

Quadro 1: Baselayout VServer no Gentoo

Para utilizar o sistema Gentoo hóspede instalado no mundo virtual do VServer, será necessário substituir o pacote `baselayout` pelo `baselayout-vserver`. O principal efeito disso é a modificação dos scripts de inicialização, causando algumas mudanças dramáticas a `/etc/init.d/` (veja o [exemplo 1](#)).

junto com seu *alias* e endereço IP estático. O comando a seguir cria a configuração para uma instância chamada *VSo1* em `/etc/vservers/vs01`. Ao mesmo tempo, cria uma árvore de diretórios mínima em `/vservers/vs01`. A árvore-esqueleto (*skeleton*) depois será sobrescrita.

```
vserver vs01 build -m skeleton --hostname
└─ vs01 --initstyle plain --context 1001
└─ --interface vs01=eth1:192.168.39.11/24
```

Agora podemos continuar a instalação do sistema Gentoo hóspede em `/vservers/vs01` da forma normal, e descompactar um *stage3*, junto com um *snapshot* da árvore do *portage*. Se você estiver usando um hospedeiro Gentoo, pode ignorar o pedaço do *snapshot* do *portage* e usar uma montagem somente-leitura para permitir que o sistema hóspede acesse o sistema de arquivos do *portage* durante a fase de instalação.

```
mount /usr/portage
└─ /vservers/vs01/usr/portage -o bind,ro
mount /usr/portage/distfiles
└─ /vservers/vs01/usr/portage/distfiles
└─ -o bind,rw
```

Extensões BME

É bastante comum querer permitir que um grupo de servidores virtuais (ou todos eles) tenham acesso compartilhado a uma seção específica do sistema de arquivos. É para isso que servem as montagens em *bind* no Linux. O que isso faz é montar uma subárvore do sistema de arquivos em uma posição diferente. Se você quiser restringir isso ao acesso somente-leitura, você esbarará em um dos recursos menos memoráveis do Linux: a opção *ro*, em combinação com a *bind*, é silenciosamente ignorada pelo comando *mount*. Você esperaria aqui uma mensagem de erro dizendo que as opções de montagem para o diretório de origem serão aplicadas.

No ambiente *chroot* do servidor virtual, mude agora o *profile* para *vserver*:

```
(vs01 chroot) gs / # rm /etc/make.profile
(vs01 chroot) gs / # ln -s
└─ /usr/portage/profiles/default-linux/
└─ x86/2005.1/vserver /etc/make.profile
```

Para completar essa etapa, instale os pacotes *Syslog-NG* e *OpenSSH*, e coloque-os no *runlevel default*. É claro que você não precisa compilar um kernel ou instalar um carregador de inicialização para o servidor virtual. Em vez disso, apenas substitua os scripts de inicialização normais pelos do pacote *baselayout-vserver* (veja o [quadro 1](#)).

É importante mudar esses pacotes, pois servidores virtuais não podem ter acesso direto aos dispositivos, o que impossibilita algumas ações, como o carregamento de módulos, pelo servidor virtual. O mesmo princípio deveria ser aplicado ao OpenVZ, entretanto, algumas alterações são necessárias nesse caso [\[4\]](#).

Nesse ponto, você também precisará modificar a configuração do Syslog-NG (veja o [exemplo 2](#)), pois o acesso a `/proc/kmsg` é impossível num ambiente VServer. Os patches do VServer impedem a abertura desse pseudo-arquivo, assim como bloqueiam o `/proc/uptime`.

Antes de deixar o ambiente *chroot*, não se esqueça de modificar o conteúdo de seus arquivos `/etc/conf.d/hostname` e `/etc/conf.d/domainname`.

Vamos agora iniciar o novo servidor, digitando `vserver vs01 start`. A fronteira entre o sistema hospedeiro, que tem contexto 0 (zero), e o VServer é semipermeável. Em outras palavras, você pode entrar nos servidores virtuais sem se autenticar, apesar de isso não se aplicar na direção contrária.

A verificação da lista de processos nos mostra que há algumas diferenças entre o mundo virtual e um ambiente nativo. Então, o que acontece, exatamente, quando iniciamos o VServer?

Namespaces

Primeiramente, a ferramenta *vnamespace* instala um novo *namespace* Linux para processos ao chamar `clone()` com a opção `CLONE_NEWNS`. *Namespace* é uma visão, específica para o processo, do sistema de arquivos. Se você rodar *mount* no *namespace* do processo A, isso não será visível para o processo B, que está em outro *namespace*. Isso explica por que montar servidores virtuais não afeta o *namespace* do sistema hospedeiro. Além disso, o

sistema de arquivos raiz do VServer no novo *namespace* é montado (em *bind*) recursivamente como `/`, tomando impossível escapar da gaiola *chroot*. Agressores que conseguirem escapar chegarão no sistema de arquivos do VServer.

Depois disso, são configurados *aliases* para as interfaces de rede. A ferramenta *chbind* associa outros processos ao endereço IP do VServer.

```
ListenAddress 192.168.1.0
ListenAddress 192.168.39.10
ListenAddress 192.168.48.10
```

A configuração da interface do VServer está localizada em `/etc/vservers/vs01/interfaces/0/`. Se o servidor virtual tiver múltiplas interfaces, é possível acrescentar mais diretórios em `/etc/vservers/vs01/interfaces/[1-9]` e copiar os arquivos.

O próximo passo é montar os diretórios necessários no *namespace*. O diretório raiz vem primeiro; ele é especificado como um link em `/etc/vservers/vs01/vdir`, seguido pelas montagens em `/etc/vservers/vs01/fstab`. Serão necessárias entradas para, no mínimo, `/proc` e `/dev/pts`. Se você também quiser montar a árvore do *portage* do sistema hospedeiro durante o funcionamento do sistema, pode usar para isso o *fstab* do [exemplo 3](#).

Exemplo 1: Baselayout do VServer

```
(vs01 chroot) gs / # emerge baselayout-vserver
(vs01 chroot) gs init.d # ls -l
total 84
-rwxr-xr-x 1 root root 2871 Jan 16 14:48 bootmisc
lrwxrwxrwx 1 root root 5 Jan 16 14:48 checkfs -> dummy
lrwxrwxrwx 1 root root 5 Jan 16 14:48 checkroot -> dummy
lrwxrwxrwx 1 root root 5 Jan 16 14:48 clock -> dummy
lrwxrwxrwx 1 root root 5 Jan 16 14:48 consolefont -> dummy
...
...
lrwxrwxrwx 1 root root 5 Jan 16 14:48 urandom -> dummy
```

Exemplo 2: syslog-ng.conf

```
01 options {chain_hostnames(on);
02 sync(0);
03 stats(43200);};
04 source src { unix-stream("/dev/log"); internal();};
05 destination server { udp("192.168.39.10",port(514)); };
06 log { source(src); destination(server);};
```

Exemplo 3: fstab para o VServer

```
01 none /proc proc defaults 0 0
02 none /tmp tmpfs size=16m,mode=1777 0 0
03 none /dev/pts devpts gid=5,mode=620 0 0
04 # Arvore portage compartilhada
05 /usr/portage /usr/portage none bind,ro 0 0
06 /usr/portage/distfiles /usr/portage/distfiles none bind,rw 0 0
```

Tenha cuidado se o VServer não for plenamente confiável. Embora a aplicação de *checksums* impeça a sobrescrita de arquivos do hospedeiro, o hóspede conseguirá injetar dados tanto no hospedeiro quanto nos outros hóspedes, nessa configuração.

Finalmente, digite *vcontext* para definir o contexto de processo para o primeiro processo do VServer, e *chbind* para restringir a ligação aos endereços IP do servidor virtual. Só especifique parâmetros de agendamento e limites de recursos se o administrador decidir usá-los. Num servidor virtual Gentoo, o primeiro processo a iniciar é o */sbin/init*; num SV Debian, seria */etc/init.d/rc 3*.

Num sistema Gentoo, executar *init* normalmente apenas rodaria os scripts do runlevel *boot*, antes de continuar carregando o *default*. Naturalmente, não se pode usar os scripts originais do Gentoo para isso, pois eles pressupõem um sistema nativo e acesso privilegiado ao dispositivo de armazenamento (como */dev/hda1*, por exemplo) para verificar o sistema de arquivos, ou para configurar o relógio do sistema. É por isso que o pacote *baselayout-vserver* modifica o */etc/inittab* e substitui os scripts de inicialização para rodarem apenas o seguinte:

```
bootmisc
domainname
hostname
local
net.lo
rminologin
ssh
syslog-ng
```

Faz sentido parar o VServer nesse ponto, rodando *vserver vs01 stop*, e fazer um backup de todo o sistema de arquivos como um modelo para servidores virtuais que porventura venham a surgir:

```
cd /vservers
tar jcvf vserver-gentoo-template.tbz2
./vs01
```

Anomalias

Um servidor virtual deve aparecer exatamente como um ambiente Linux nativo para os aplicativos e usuários. Um

pouco de mapeamento é necessário para atingir esse objetivo. Um exemplo disso é o *PID* do processo *init*. Apesar de isso não ser estritamente necessário, diversos programas implicitamente pressupõem que o *init* tem *PID 1*. Não existe uma forma disso ser verdadeiro num VServer. Portanto, a entrada do *init* tem que receber explicitamente o *PID 1* quando o sistema de arquivos */proc* for virtualizado.

Se o sistema hóspede for um SV Debian, apenas */etc/init.d/rc 3* é executado na inicialização, o que significa que um processo *init* nem existe. Nesse caso, o */proc/1* tem que ser simulado. Isso também se aplica a outras entradas do */proc*, como */proc/uptime*, por exemplo.

Acrescentar outros ambientes é bem simples agora. Use as ferramentas *vserver* para criar uma nova configuração de SV, e copie o sistema de arquivos do SV. Depois, mude as configurações de rede. A ferramenta *vserver-new* facilita isso ainda mais:

```
vserver-new vs02 --context 1002
--hostname vs02 --interface
vs02=eth1:192.168.39.14/24 clone vs01
```

Outros hóspedes

O interessante da virtualização baseada no particionamento no nível do kernel é que o kernel é o mesmo para todas as partições ou servidores virtuais. Essa não é uma restrição muito importante, já que não impede a instalação de outras distribuições como sistemas hóspedes. Em nosso exemplo, instalaremos um hóspede *Debian* num hospedeiro Gentoo.

É incrivelmente fácil fazer isso usando o *debootstrap*. Ao invés de instalar um esqueleto VServer conforme descrito acima, pode-se permitir que o *vserver* coopere com o *debootstrap*. Após executar *emerge debootstrap*, faça o seguinte para completar a instalação do VServer:

```
vserver vs03 build --context 1003
--hostname vs03 --interface
vs03=eth1:192.168.39.23/24 -m
debootstrap -- -d sarge
```

Administração

Quanto mais VServers você instalar, maior será a capacidade de armazenagem necessária. Se todos os seus servidores forem criados usando o mesmo modelo, a maioria dos arquivos será idêntica em vários servidores. Para simplificar e acelerar o processo, pode-se deixar que seus servidores virtuais compartilhem partes do sistema de arquivos.

Se os administradores dos hóspedes não forem fazer grandes modificações nos servidores, uma opção é montar o sistema de arquivos raiz em modo somente-leitura, com exceção do */etc/*, */var* e os sistemas de arquivos de dados. Infelizmente, em várias situações, isso é muito restritivo e, por isso, o Linux VServer utiliza uma técnica diferente.

Unificação e quebra de links em copy-on-write

Se os diretórios dos servidores virtuais residirem em um mesmo sistema de arquivos, pode-se usar o *vhashify* para agrupar arquivos idênticos num diretório central e substituir os arquivos por hardlinks nos servidores virtuais, deixando apenas uma instância de cada objeto de arquivo, economizando assim espaço de armazenamento. Obviamente, o acesso de gravação só pode ser dado ao servidor virtual em execução. A solução que permite isso é conhecida como *CoWLB*, ou *copy-on-write link breaking*. O hardlink que aponta para o diretório central é apagado, e o arquivo é substituído pela cópia modificada. O *CoWLB* é suportado pelas versões *developer 2.1.0* e posteriores dos patches do VServer.

Essa solução pode ser aplicada a diretórios arbitrários, como mostra o exemplo a seguir. Um arquivo chamado *x* reside nos diretórios */vservers/a0[12]/x*:

```
gs vservers # ls -li a0[12]/x
685511 -rw-r--r-- 1 root root 942
Jan 17 07:12 a01/x
685514 -rw-r--r-- 1 root root 942
Jan 17 07:12 a02/x
```

Com o comando a seguir, podemos substituir o arquivo por uma cópia com *inode 122620*. Arquivos muito pequenos são ignorados.

```
gs vservers # /usr/lib/util-vserver
/vhashify --manually --destination
/vservers/.hash /vservers/a01 exclude
/vservers/vexclude
```

Quadro 2: A armadilha de associação de endereços

Você precisa se certificar de que os processos do servidor no contexto do hospedeiro não se liguem a todas as interfaces e *aliases* disponíveis. Caso contrário, os *daemons* que estiverem rodando nos servidores virtuais serão incapazes de se ligar a eles. O daemon *ssh* geralmente causa problemas. Ele é configurado para se ligar a todos os endereços por padrão. Para eliminar essa armadilha, assegure-se de especificar os endereços certos em */etc/ssh/sshd_config*, e retire todas as entradas referentes aos aliases dos servidores virtuais.

Para o arquivo `a01/x`, os rótulos `immutable` e `unlink` são especificados pelo `vhashify`. Isso impede a modificação do arquivo mas permite que os hardlinks sejam apagados, e é usado para implementar o *copy-on-write*.

```
gs vservers # showattr a0[12]/x
---UI- a01/x
---ui- a02/x
```

Até agora isso não teve muito efeito. Então, vamos aplicar o mesmo princípio ao diretório `/vservers/a02`:

```
gs vservers #
➤ /usr/lib/util-vserver/vhashify
➤ --manually --destination
➤ /etc/vservers/.defaults/apps/vunify/hash
➤ /vservers/a02 exclude /vservers/vexclude
gs vservers # ls -li a0[12]/x
122620 -rw-r--r-- 3 root root 942
➤ Jan 17 07:12 a01/x
122620 -rw-r--r-- 3 root root 942
➤ Jan 17 07:12 a02/x
```

As duas novas entradas existem como hardlinks no inode 122620. Se alterarmos o arquivo agora, ele será apagado e substituído por uma cópia:

```
gs vservers # echo "test" >> a02/x
gs vservers # ls -li a0[12]/x
122620 -rw-r--r-- 2 root root 942
➤ Jan 17 07:12 a01/x
685511 -rw-r--r-- 1 root root 947
➤ Jan 17 07:14 a02/x
```

O `vhashify` unifica os dois diretórios no VServer `vs01` e no clone `vs02`. O arquivo `/vservers/vexclude` contém uma lista de exceções com diretórios que não queremos unificar. Isso faz sentido para dados inerentemente variáveis, como o conteúdo do diretório `/var`, ou os arquivos de dispositivos sob `/dev/`. O `vhashify` é o sucessor do `vunify` e pode usar a mesma lista de exceções padrão em `/usr/lib/util-vserver/defaults/vunify-exclude`. Se você criar um link agora, `ln -s /vservers/.hash /etc/vservers/.defaults/apps/vunify/hash/0`, você pode unificar digitando simplesmente `/usr/lib/util-vserver/vhashifyvs02`.

Você verá que o diretório `.hash` cresceu consideravelmente, apesar de parecer que nada aconteceu com `/vservers/vs0` [1].

```
gs vservers # du -sh /vservers/.hash
➤ /vservers/vs0[12]
391M /vservers/.hash
331M /vservers/vs01
331M /vservers/vs02
```

A ferramenta `vdu` consegue distinguir corretamente entre hardlinks. A econo-

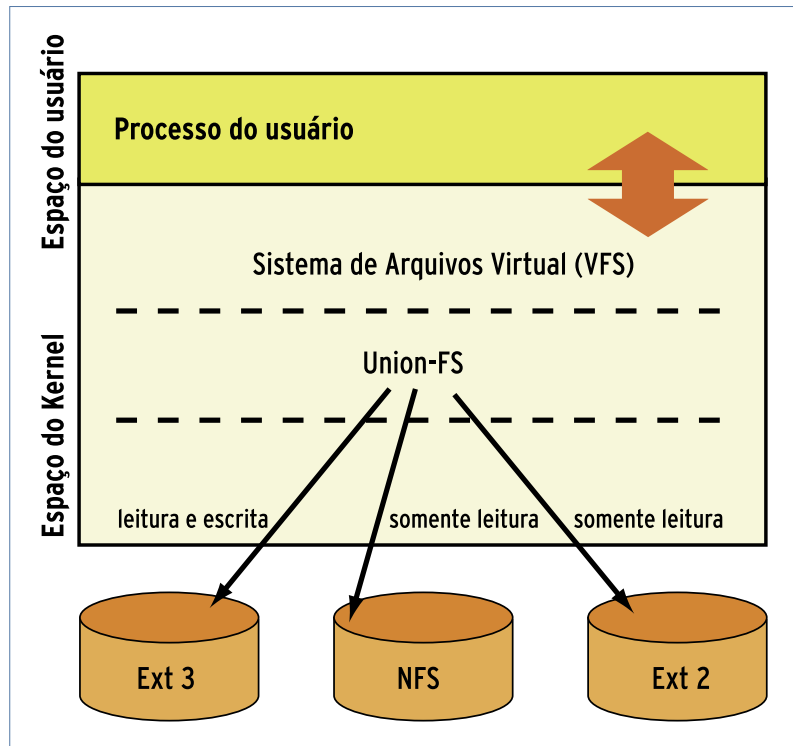


Figura 1 Unificação de ramos de sistemas de arquivos com o UnionFS.

mia de espaço para um grande número de VServers é enorme.

```
gs vservers # vdu /vservers/vs0[12]
/vservers/vs01 5K
/vservers/vs02 5K
```

Dependendo de qual aplicação você tiver em mente para seus VServers, você notará que os diretórios dos SVs tendem a se diferenciar progressivamente.

UnionFS

Outra solução elegante para unificar os diretórios do VServer envolve a utilização do UnionFS [5], e suporta a unificação além dos limites do sistema de arquivos. O UnionFS (ainda) não está incluído no kernel “oficial”, e está mascarado no Gentoo. Isso significa que se deve instalar o módulo do kernel e as ferramentas do espaço do usuário com o comando:

```
echo 'sys-fs/unionfs' >>
➤ /etc/portage/package.keywords
➤ && emerge unionfs
```

O desenvolvimento desse sistema de arquivos está avançando a passos largos, e a versão atual é suficientemente estável para a maioria dos propósitos.

O UnionFS é um sistema dito *ventilado*. As operações no sistema de ar-

quivos são distribuídas ao longo de um grupo de sistemas de arquivos (figura 1). Quanto mais para a esquerda estiver um ramo, mais alta é sua prioridade. Uma operação de busca de um arquivo `a` começará na extremidade esquerda do ramo, e terminará no ramo onde se encontra o arquivo.

Se o ramo com a maior prioridade for montado em modo de leitura e escrita no grupo, seu espaço pode ser usado para operações de leitura. Para apagar um arquivo `b`, não é suficiente deletar o arquivo no ramo com a maior prioridade, pois ele pode ainda existir nos ramos mais para a direita. Nesse caso, um arquivo em branco, `.wh.b`, é gravado para mascarar os arquivos à direita. Isso é necessário até mesmo se o arquivo vier de um ramo somente-leitura. Se você criar um arquivo `c`, ele só poderá ser criado no ramo de leitura e escrita. Para modificar um arquivo `d` de um ramo somente-leitura, primeiro é necessário copiar o arquivo para o ramo de leitura e escrita com a segunda maior prioridade.

Podemos usar o `unionctl` para inserir dinamicamente o ramo `/vservers/vsno-portage` num sistema de arquivos `unionfs` em frente ao `/vservers/vs0master`, e provavelmente vamos querer acrescentar uma entrada em `/etc/fstab` para tornar isso permanente: ▶

```
none /vservers/vs04 unionfs
└─ dirs=/vservers/vs04diff=rw:/vservers/
└─ vsnportage=ro:/vservers/vsmaster=ro 0 0
```

Depois de iniciar o SV `vs04`, temos um sistema sem o sistema de gerenciamento de pacotes `portage`. Obviamente, esse princípio pode se aplicar a outros pacotes que tenhamos que adicionar ou remover independentemente do mestre. Contudo, a manipulação de pacotes deve ficar restrita a um ramo do UnionFS por motivo de consistência.

Se você pretender usar BME e Co-WLB em combinação com o UnionFS num sistema hospedeiro, atualmente isso significa aplicar um patch do UnionFS [6]. Esse passo é necessário por causa das alterações na assinatura do auxiliar do VFS causadas pelos patches do VServer. Você perde a compatibilidade com os módulos de kernel que chamam o auxiliar do VFS da maneira tradicional.

OpenVZ

O OpenVZ [2], a variante livre da solução comercial de virtualização Virtuozzo [7], oferece uma funcionalidade semelhante à do Linux VServer. O patch para o kernel está disponível no website do OpenVZ, assim como kernels pré-compilados. No Gentoo, há o pacote `openvz-sources` para ajudar na integração dos patches do OpenVZ e outros patches críticos. As ferramentas necessárias

Quadro 3: Contextos

A introdução de contextos é uma das modificações mais significativas que os patches do VServer fazem no kernel Linux. Um contexto define o número de processos que cooperam e competem uns com os outros. Os processos em diferentes contextos não podem mais cooperar no sistema local (veja a figura 2). Para deixar isso acontecer, um ID de contexto é adicionado ao PID para suportar a designação de processos únicos. O contexto original recebe uma ID de 0 (zero). Esse contexto é automaticamente gerado quando o kernel é inicializado. O contexto da raiz não tem qualquer papel especial na separação dos processos, e os processos em outros contextos não podem ser influenciados de dentro dele. Para dar aos administradores uma idéia geral de todos os processos em todos os contextos de um sistema, o contexto de monitoramento 1 foi introduzido. Todos os processos do sistema ficam visíveis no contexto 1, que você só pode entrar se vier do contexto da raiz. O `vcn-text` permite que você crie uma nova ferramenta, ou entre em algum outro contexto.

O comando `vcn-text --migrate --xid 1 ps aux` entra no contexto de monitoramento 1 e inicia o `ps aux` nesse contexto, para gerar como saída uma lista de processos de todos os contextos. As ferramentas `vps` e `vtop` estão disponíveis para trabalhos de administração simples e listam ainda a ID e o nome do contexto. Por motivos de segurança, o contexto raiz 0 deveria ser acessível somente para gerenciar os outros contextos de um sistema VServer Linux. Os serviços rodam em seus próprios contextos, e o acesso ao contexto raiz deve ser restrito.

do espaço do usuário vêm nos pacotes `vcctl` e `vzquota`. No momento da escrita deste artigo, os patches do OpenVZ foram feitos para o kernel 2.6.8. Patches estáveis não estão disponíveis para versões mais recentes do kernel. Existe uma versão beta disponível para o kernel 2.6.15. E você sempre pode usar o `genkernel` para compilar um kernel. Mas a abordagem mais fácil é usar uma configuração estável do website do OpenVZ. Descompacte a versão em `/usr/src/linux` com o arquivo `.config`, e depois prossiga com a compilação, como de costume. Para iniciar automaticamente o servidor OpenVZ, você deverá executar `rc-update add vz default`.

Note que você precisará criar o `/dev/vzctl` antes de iniciar o hospede pela primeira vez: `/bin/mknod /dev/vzctl c 126 0`. Se a opção `RC_DEVICE_TARBALL="yes"` estiver em uso no `/etc/conf.d/rc`, o arquivo de dispositivo será salvo em `/lib/udev-state/devices.tar.bz2` quando você desligar o hospedeiro, e será recuperado na próxima vez que você inicializar o hospedeiro.

Outra coisa importante é realizar as alterações necessárias em `/etc/sysctl.conf`:

```
net.ipv4.ip_forward = 1
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
kernel.sysrq = 1
```

Após gerar o kernel, modificar o carregador de inicialização e instalar as ferramentas de espaço de usuário, você pode reiniciar o sistema.

Hóspedes no OpenVZ

Podem parecer uma boa idéia instalar um sistema Gentoo hospede agora mesmo, e não demorará muito até você poder usar para isso o modelo `base1ayout-vserver` mencionado acima [4]. Compacte o VServer mestre e guarde-o em `/vz/template/cache/vsmaster.tar.gz`. Você pode prosseguir criando um hospede do OpenVZ assim:

```
vcctl create 2001 --ostemplate vsmaster
└─ --ipadd 192.168.39.21 --hostname ovz01
```

Novamente, de forma semelhante ao VServer, precisamos associar um ID de contexto. Acrescente `.tar.gz`

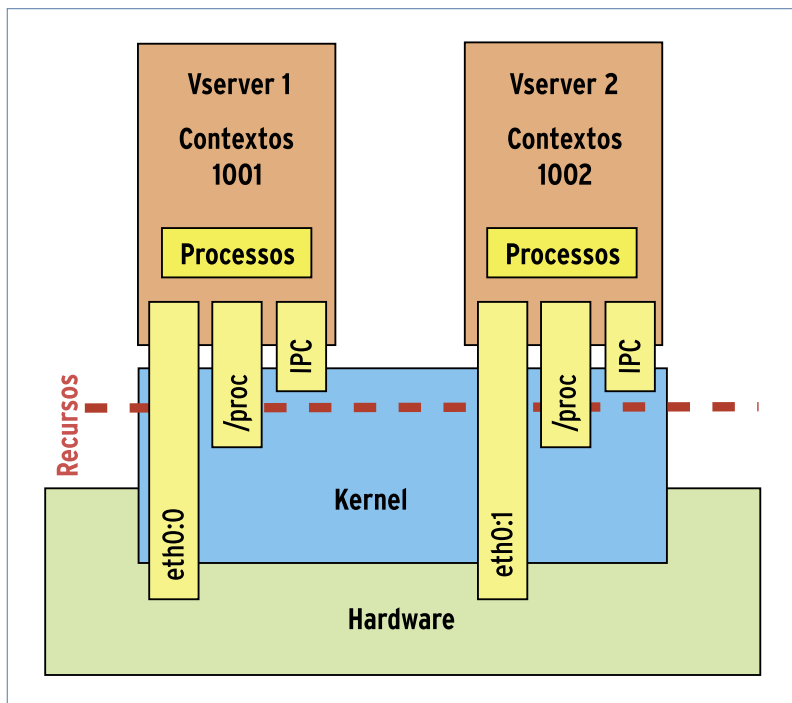


Figura 2 O Vserver cria partições isoladas de sistemas operacionais.

ao valor do parâmetro para `ostemplate` e renomeie o arquivo com o sistema hospede para `/vz/template/cache`. Isso não funcionará no Gentoo, devido a algumas falhas que deveriam ser fáceis de eliminar, mas atualmente impedem o uso em produção.

Por outro lado, é bem fácil instalar um sistema Debian como hospede. Use `debootstrap` para preparar um sistema, e comprima o sistema para instalação no OpenVZ:

```
cd /vz/template/cache
mkdir debian_sarge
debootstrap sarge ../debian_sarge
cd debian_sarge; tar zcvf
../debian_sarge.tar.gz .
```

Rodar a ferramenta de instalação `vmzctl` descompacta o arquivo, joga-o em `/vz/private/<id>` e roda o script `/etc/vz/dists/scripts/postcreate.sh` para fazer algumas alterações.

```
vmzctl create 2005 --ostemplate
> debian-sarge --ipadd 192.168.39.25
> --hostname ovz05
```

Depois, você pode iniciar o VServer. Note que o OpenVZ utiliza o `/sbin/init` para iniciar os sistemas hóspedes. Os processos `getty` iniciados pelo `init` são inúteis, devido à falta de acesso ao hardware, e o mesmo se aplica ao `klogd`. A virtualização com o OpenVZ elimina o `/proc/kmsg`. Certifique-se de desativar esses serviços para resolver esses problemas. Como os scripts de inicialização não rodam nenhum comando `mount`, será necessário substituir o `/etc/mtab` por um link simbólico apontando para `/proc/mounts`.

Use `vmzctl enter 2005` para mudar para o contexto do VPS. Certifique-se de verificar ou especificar explicitamente as variáveis de ambiente do `Bash`. Após rodar `apt-get update`, pode-se instalar qualquer software de que se precise. Faz sentido salvar o sistema hospede modificado como um modelo para o Debian quando você terminar; depois, você pode usar o modelo para gerar mais sistemas hóspedes.

Se você precisar realizar alguma ação especial ao iniciar e parar um servidor virtual, pode usar scripts para isso. Ao contrário do que o manual do OpenVZ informa, o sistema Gentoo hospede espera que esses scripts estejam em `/etc/vz/<vsid>.{mount,umount,start,stop}`. Os scripts `mount/umount` rodam no contexto do hospedeiro, enquanto os scripts `start/stop` rodam no contexto do SV.

Modelos para várias distribuições podem ser baixados em [2]. Além do Debian, no momento há modelos para *CentOS*, *Fedora*, *Gentoo*, *Mandriva* e *OpenSuse*. Os modelos oferecem uma solução fácil para a instalação de todos os sistemas hóspedes.

Infelizmente, os métodos a que nos referimos antes para suportar o gerenciamento de grandes números de servidores virtuais não são atualmente suportados pelo OpenVZ. Todavia, novamente, a *SWsoft* oferece uma ferramenta comercial para isso: o *Virtuozzo*. O *UnionFS* não é mais compatível com o já idoso kernel 2.6.8, e não é possível aplicar os patches *BME*. Tudo isso vai mudar quando for implementado o suporte a kernels mais recentes. Para contornar o problema, você pode mover os sistemas de arquivos do SV para um servidor NFS de alta performance, que suportará todas as opções descritas anteriormente. Isso acrescenta o benefício da migração indolor de um servidor virtual de uma máquina física para outra.

Futuro

Nem o VServer nem o OpenVZ atingiram todo seu potencial. Além do gerenciamento efetivo do VServer, também é importante ter um controle granular dos recursos para gerenciar o espaço em disco e os ciclos de CPU. Mais uma vez, ambos os projetos possuem mecanismos como quotas e escalonador justo para controle dos recursos. ■

Mais Informações

- [1] Linux-VServer: <http://linux-vserver.org/>
- [2] OpenVZ: <http://openvz.org/>
- [3] Projeto Gentoo VServer-Utils: <http://dev.croup.de/proj/vserver-utils>
- [4] Gentoo VPS: <http://dev.croup.de/proj/gentoo-vps>
- [5] Homepage do UnionFS: <http://www.fsl.cs.sunysb.edu/project-unionfs.html>
- [6] Patch de desenvolvimento do VServer para UnionFS: <http://mozart.informatik.fh-kl.de/download/Software/VServer/vserver.html>
- [7] SWsoft Virtuozzo: <http://www.virtuozzo.com/>

Instant Messaging

**Não proíba!
Gerencie e Controle!**

Com o messengerPOLICY®, você conta com todas as facilidades de que precisa para garantir o uso produtivo e seguro do MSN® em sua empresa.

- Interface web de administração
- Orientado a objetos (usuários, grupos, horários e recursos)
- Uso personalizado do MSN® conforme suas necessidades
- Monitoramento on-line das mensagens trocadas
- Histórico de todas as conversas
- Gráficos estatísticos para gerenciamento

No messengerPOLICY®, o administrador pode criar os mais diversos tipos de regras, como por exemplo:

"JOÃO" (OBJETO - USUÁRIO) PODE CONVERSAR COM "COLEGAS" (OBJETO - GRUPO DE USUÁRIOS) RESPEITANDO "LAZER" (OBJETO - TABELA DE HORÁRIO).



**messenger
POLICY**

**Não seja alvo de pragas virtuais,
proteja-se com o messengerPOLICY®,
saiba mais acessando www.brc.com.br.**

BRCONNECTION
CRESCER SEGURO

www.brc.com.br | 55 (11) 2165-8888