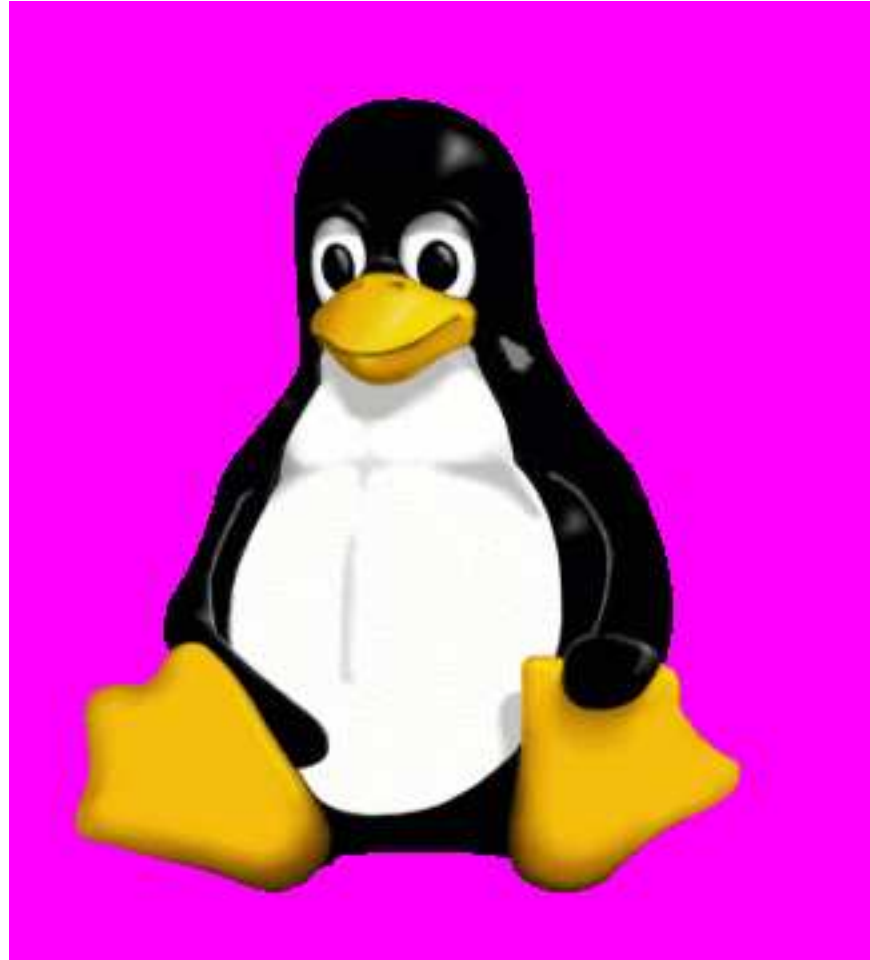


# Running Vservers on Debian

by Bart

bart@jukie.net



April OCLUG tutorial  
2005-04-28

# Overview

---

- Virtualization techniques
- What is a Vserver
- Simple Demo
- Installing
- Tips and Tricks
- Q & A

# Virtualization Techniques

---

- CPU emulation
- CPU virtualization
- User-mode kernel
- Process separation
- Microkernels & hypervisors

# ... CPU Emulation

- What is it?
  - "processor run-time simulation down to each register"
  - transparent to client software
  - very low performance
  - very portable as there is no host dependency
  - can simulate non-existent hardware
  
- Examples:
  - QEMU
  - bochs
  - PearPC
  - VirtualPC (Mac version)

# ... CPU Virtualization

- What is it?
  - "client OS is fooled into thinking it's in charge"
  - virtualization support from processor
  - transparent to client software
  - privileged operations are captured and emulated
  - most non-IO code is executed w/o penalty
  
- Examples:
  - VMWare
  - VirtualPC (Windows version)

# ... User Mode Kernels

- What is it?
  - "port of kernel to POSIX API"
  - executes a client kernel in a user space process
  - client OS drivers emulate access to hardware through host OS APIs
  - transparent to client user space software
  
- Examples:
  - User Mode Linux
  - CoLinux

# ... Process separation

- What is it?
  - "hide processes from each other"
  - multiple execution contexts for user space processes
  - contexts are restricted in CPU, network and disk usage
  - almost zero overhead on process execution
  
- Examples:
  - vserver (linux 2.4 & 2.6)
  - jails (BSD)
  - Virtuozzo (comertial product for Linux & Windows)

# ... Microkernels & Hypervisors

- What is it?
  - referred to as Paravirtualization
  - hypervisor -- layer between hardware & client OS
  - presents "virtual machines" with a software interface
  - software interface is similar to underlying hardware
  - non transparent -- requires client kernel changes
  
- Examples:
  - Traditional Microkernels
    - ▷ Mach
    - ▷ L4
  
  - Microkernels for the sake of virtualization
    - ▷ Xen
    - ▷ IBM zSeries' z/VM & LPAR

# What is a Vserver

---

- Why?
- Overview
- Isolation
- Tools
- Unification
- Limitations

# ... Why would you want a vserver?

---

- (1) security,
- (2) portable setup,
- (3) test an upgrade path,
- (4) develop in a sandbox.

# ... Overview

- Components
  - kernel patch
  - vservers tools
  
- Each client OS...
  - can have its own unique root filesystem (via chroot)
  - has its own IP address (and privileged ports)
  - has its own user/group database (even unique root account)
  - cannot corrupt other client OS's
  - cannot see or access other context's processes
  - cannot break out of a chroot
  - cannot create new device nodes

# ... Process isolation

- **File system**
  - chroot protection for each vserver
  - each context can have it's own quotas on a filesystem
- **Networking**
  - contexts are assigned IP number and hostname
  - only this IP number can be used from the context
- **Processes**
  - contexts only see processes in their context
  - contexts have private SysV IPC resources
- **Super user capabilities**
  - 'root' of vserver has limited capabilities

# ... Tools

- **chcontext**
  - create a new security context
  - root of base context can enter any context
  
- **chbind**
  - lock processes to use specific IP on bind() call
  
- **reducecap**
  - allows for reduction of capabilities

# ... Unification

## □ Problem

- you have 10 Debian/sarge vservers
- you have 10 clients with similar software requirements
- majority of the apt-get installed files are identical

## □ Solution

- hardlinks used remove the duplication
- shared files use 'immutable' file attribute

# ... Vserver limitations

- Kernel is shared
  - Kernel-aimed attacks can still compromise systems
  - Not good for kernel/driver development/testing
  - You may not be able to run certain OS versions together
  
- Network access is limited
  - you have to configure the firewall from the host OS
  - you cannot ping, traceroute, or tcpdump
    - ... but, you can loosen up security restrictions
  
- X is a pain
  - a lot of configuration, lower security
    - ... but, not really a requirement for most servers

# Demo

---



# Installing

- Host OS install
- Kernel patch
  - Installing using kpkg (2.4)
  - Installing from scratch (2.6)
- Configuration
  - Host OS
  - The first vserver
  - Subsequent vservers
  - Unification

# ... Host OS install

## □ Recommendations

- minimal
- no services, daemons, etc
  - ▷ except for ssh, and
  - ▷ those that fail to run in vserver (ex: bind)

## □ Packages

- ssh daemon and client
- network testing tools
- iptables firewall
- vserver tools package: `util-vserver`
- debian tools package: `vserver-debiantools`

# ... linux-2.4 w/vserver using kpkg

## □ Requirements

```
apt-get install kernel-package fakeroot \  
build-essential kernel-patch-vserver
```

## □ Building

```
CONF=" --rootcmd=fakeroot  
      --append-to-version=-vs1.2.11  
      --bzimage  
      --subarch i686  
      --added-patches ctx"  
make-kpkg ${CONF} kernel-image  
make-kpkg ${CONF} modules-image  
make-kpkg ${CONF} kernel-headers
```

# ... linux-2.6 w/vserver from a patch

## □ Requirements

- [http://www.13thfloor.at/vserver/d\\_rel26/v1.9.5/](http://www.13thfloor.at/vserver/d_rel26/v1.9.5/)
  - ▷ get the .diff.bz2 file
- <http://kernel.org/pub/linux/kernel/v2.6/>
  - ▷ get a kernel that matches the patch
- `apt-get install build-essential gcc-3.4`

# ... linux-2.6 w/vserver from a patch

## □ Building

```
cd /usr/src/  
tar xjf ${KERNEL}  
cd linux-2.6.11.5  
bunzip2 -c < ${PATCH} | patch -p1 --dry-run  
bunzip2 -c < ${PATCH} | patch -p1  
  
export CC=gcc-3.4  
make menuconfig  
make bzImage modules  
make modules_install install
```

# ... linux-2.6 w/vserver from a patch

## □ Configuration

Linux VServer --->

- Enable Legacy Kernel API
- Enable Proc Security
- Enable Hard CPU Limits
- Persistent Inode Context Tagging
  - Disabled
  - UID16/GID32
  - UID32/GID16
  - UID24/GID24
  - UID32/GID32
  - Runtime
- Compile Debugging Code

# ... Host OS configuration

- /etc/vservers.conf
  - ▷ BACKGROUND=no
- /etc/vservers/util-vserver-vars
  - ▷ VROOTDIR=/var/lib/vservers
- /etc/newvserver-vars
  - ▷ ONBOOT=yes
  - ▷ ...
- protecting `${VROOTDIR}`
  - `chmod 000 ${VROOTDIR}`
  - `chattr +t ${VROOTDIR}`

# ... Setting up a new vservers

- on Sarge (vservers-debiantools)

```
newvservers --mirror http://ftp.debian.org \  
  --hostname foo --domain domain \  
  --ip 192.168.0.100
```

# ... Setting up a new vserver

- on Woody (debootstrap & util-vserver)

```
mkdir ${VROOTDIR}/bar
debootstrap woody ${VROOTDIR}/bar \
    http://ftp.debian.org
cp /usr/lib/util-vserver/sample.conf \
    /etc/vservers/bar.conf
vim /etc/vservers/bar.conf

chroot ${VROOTDIR}/bar
    tzsetup -y
    dpkg-reconfigure passwd
    rm -f /etc/exim/exim.conf
    eximconfig
```

# ... Setting up a new vserver

- Fedora (FC1) on Debian (util-vserver)

```
# put RH9 cd into drive
```

```
mount /dev/cdrom /mnt/cdrom
```

```
/usr/lib/util-vserver/install-fc1 minimum
```

# ... The configuration file

- `/etc/vservers/myvserver.conf`

```
ONBOOT=yes
```

```
IPROOT=192.168.0.100
```

```
IPROOTDEV=eth0
```

```
S_HOSTNAME=bar.domain
```

```
S_CONTEXT=100
```

```
...
```

# ... Cloning vservers

- on Sarge (vserver-debiantools)

```
dupvserver --vsroot ${VROOTDIR} \  
    --from foo --to fiz \  
    --ip 192.168.0.101
```

- on Woody (util-vserver)

```
vserver-copy --vsroot ${VROOTDIR} \  
    foo fiz  
vim /etc/vservers/biz.conf
```

# ... Unification

## □ vunify (util-vserver)

```
# /usr/lib/util-vserver/vunify /vs/foo /vs/bar -- ALL
vservers /vs/bar
Unify pkg adduser-3.63 from /vs/foo to /vs/bar
Unify pkg apt-0.5.28.1 from /vs/foo to /vs/bar
Unify pkg bsdmainutils-6.0.17 from /vs/foo to /vs/bar
Unify pkg debconf-1.4.30.13 from /vs/foo to /vs/bar
Unify pkg debianutils-2.8.4 from /vs/foo to /vs/bar
Unify pkg dpkg-1.10.27 from /vs/foo to /vs/bar
Unify pkg dselect-1.10.27 from /vs/foo to /vs/bar
Unify pkg hostname-2.13 from /vs/foo to /vs/bar
Unify pkg liblockfile1-1.06 from /vs/foo to /vs/bar
Unify pkg netbase-4.21 from /vs/foo to /vs/bar
Unify pkg tasksel-2.24 from /vs/foo to /vs/bar
...
```

# Tips and Tricks

- consider 'testing' for host OS
  - Sarge has better tools for vservers, raid, etc.
  - run all public services from vservers
  
- ssh into vservers, limit use of 'enter'
  - apps fail when they don't have control of the terminal
  
- rebuild bind9 w/o linux capabilities
  - otherwise you cannot run it in a vserver

# ... more Tips and Tricks

- bind host daemons to main IP
  - daemon will fail to bind if ctx 0 has bound to that port
- use 'bind' mounts to share data between vservers

```
mount /dev/big RAID_drive /export
mount -o bind /export/home /vs/one/home
mount -o bind /export/home /vs/two/home
```

- or in /etc/fstab

```
$ cat /etc/fstab
```

```
□ . . .
```

```
                /dev/big RAID_drive /export auto
defaults 0 2

                /export/home /vs/one/home auto
bind 0 2

                /export/home /vs/two/home auto
bind 0 2
```

# Vserver References

---

- Project page
  - <http://linux-vserver.org/>
  
- Linux Vserver Paper
  - <http://linux-vserver.org/Linux-VServer-Paper>
  
- S/W that fails to run in vservers
  - <http://linux-vserver.org/ProblematicPrograms>

# Other References

---

- Additional build scripts
  - <http://www.marlow.dk/site.php/tech/vserver>
  
- L4 microkernel
  - <http://l4ka.org/>
  
- Xen
  - <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>

# Questions

---

