Extreme Filesystem Sharing

Read-Only Op Sys and Other Untouchables

Rick Troth <rickt@velocitysoftware.com>

July 29, 2011

VM and Linux Workshop, Columbus, Ohio

Disclaimer

The content of this presentation is informational only and is not intended to be an endorsement by Velocity Software. (I am speaking only for myself.) The reader or attendee is responsible for his/her own use of the concepts and examples presented herein.

In other words: Your mileage may vary. “It Depends.” Results not typical. Actual mileage will probably be less. Use only as directed. Do not fold, spindle, or mutilate. Not to be taken on an empty stomach. Refrigerate after opening.

When in doubt, ask! Don't believe me? Ask the list! Still in doubt? Try it!

Extreme Filesystem Sharing

- Some history of shared content
- Some ways of sharing content
- Some reasons for sharing content
- Some solutions to sharing content

Shared Data in Computing History

- Tapes
- Disks
- Network
- social/consumer
- excessive duplication

Only wimps use tape backup: real *men just upload their important stuff on ftp, and let the rest of the world mirror it*

Linus

Filesystem Sharing Rationale

- Distribution
- Collaboration
- Recovery
- Control
- Deduplication
- Scalability

Filesystem Sharing

- Solaris sharing of `/usr`
- academic work (AIX/370 and UTS)
- Linux/390 and shared `/usr`
- Linux/390 at NW and shared root
- RW root with shared op sys
  (bind mount selected directories)

Filesystem Sharing

- Shared `/usr` and others
- RO root with RW `/etc`
- RO op sys with RW root

- System maint and package management
- Relocatable Packages
- DASD on Demand – Disk Automounter

Solution: Share More Stuff

- Install Once, Run Many
  (isn't that why they pitched Java?)
- Sharing **`/usr`**, **`/opt`**, and others,
  so why not also share the root?
- Sharing **`/bin`**, **`/lib`**, and standard op sys

Untouchable root? Sounds Weird

- Solaris/SunOS supports NFS root including read-only **`/usr`** content
- "Live CD" Linux uses bulk R/O content
  - Knoppix, Ubuntu, Kubuntu, recovery tools
- USS supports ROR already (Unix on z/OS)

Not weird, Not even new

Many uses, but not widely understood

Stability and Manageability

- R/O media is incorruptible
- R/O content is centrally maintained
- R/O packages are available on-demand
- Better D/R – less per-server replication

R/O zLinux no different from R/O PC Linux

How to Build Read-Only OS

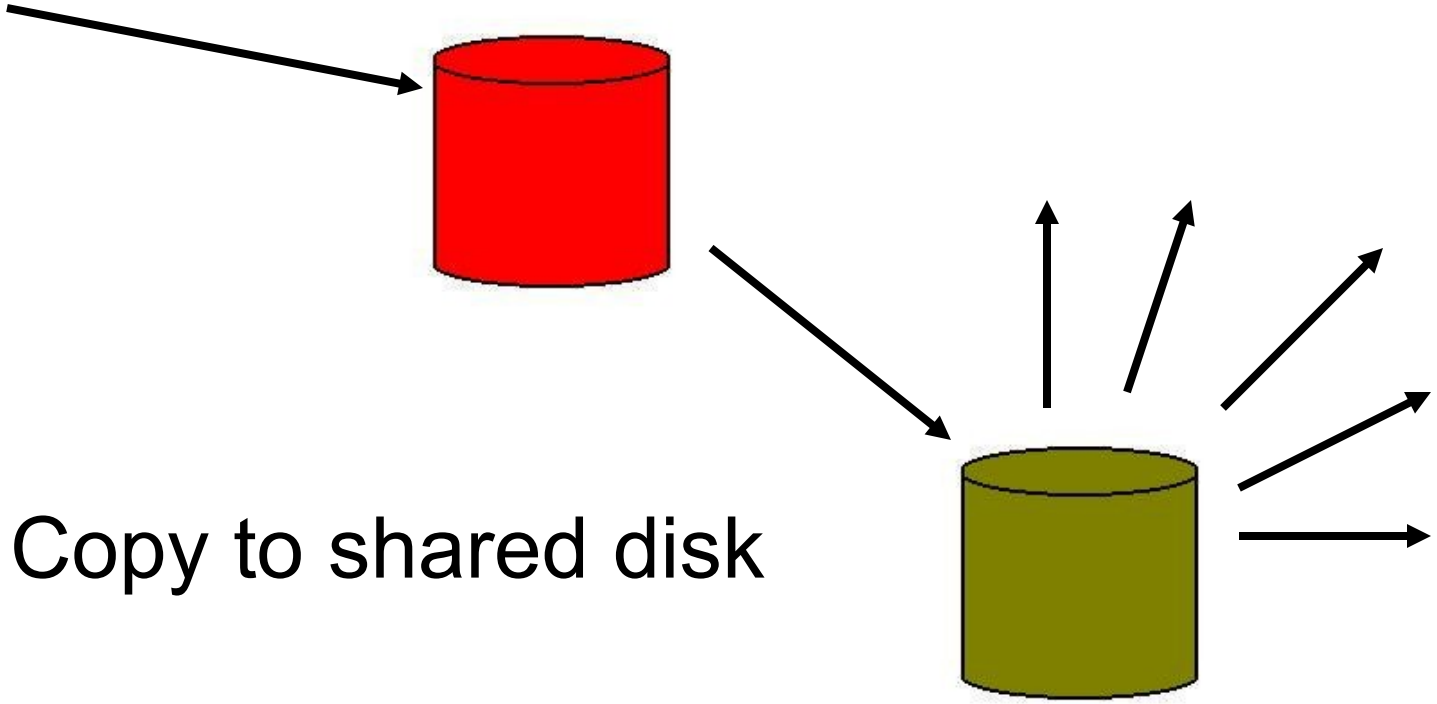
- Start with standard installation
- Copy `/etc` and `/var` to “run root”
- Create other root mount points
- Insert `/sbin/init+vol` script to boot parm

/sbin/init+vol Startup Script

```
#!/bin/sh
mount -r $_RUNFS /mnt
for D in lib bin sbin usr ; do
  mount -o bind /$D /mnt/$D
done
pivot_root /mnt /mnt/$SYSTEM
cd /
exec /sbin/init $*
```

How to Build Read-Only OS

Start with standard installation



Copy to shared disk

Reconciling RPM Database

- Initial RPM DB matches master
- “Client” systems may vary
- Master may get updates

… now what?

Reconciling RPM Database

- Extract master package list

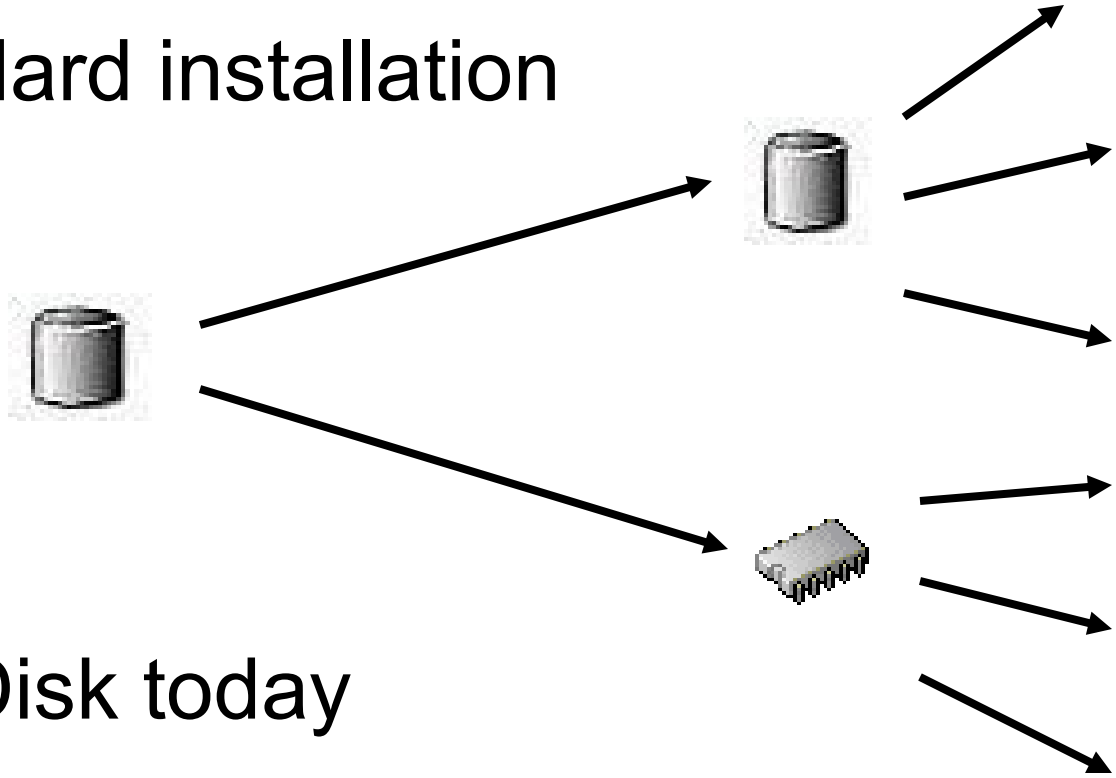
```
# rpm -q -a > master.rpml
```

- Update client RPM database

```
# for P in `cat master.rpml` ; do
    rpm –U --justdb $P.rpm ; done
```

How to Build Read-Only OS

Standard installation



Disk today

Virtual ROM tomorrow

How to … reference

1b0 == boot and op sys root

1b1 == “run root” with **/bin**, **/lib**, ... bound

1b5 == /local

1be == /usr

1bf == /opt

2b0-2bf == LVM phys vols and/or maint

320-33f == “User Space” LVM phys vols

100,200 == FCP “HBAs” for SAN

How to … reference

1b0 == boot, bootable and **/boot**

1b1 == root

Contains **etc**, **dev** and others

"personality" of the system

1b5 == /local

1be == /usr

1bf == /opt

R/O OS with Xen

```
nehemiah:~ # df
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/xvdb        5160576   1427492   3523372  29% /
udev              131168       112    131056   1% /dev
tmpfs             131168         8    131160   1% /tmp
/dev/xvdj       20642428  10102248   9491604  52% /export/home
/dev/xvdk       20642428    176320  19417532   1% /export/opt
/dev/xvdl       30963708  20238400   9152444  69% /export/media
```

R/O OS with Xen

```
nehemiah:~ # df
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/xvda        4127076   1951568   1965864  50% /Linux-i386
/Linux-i386/lib  4127076   1951568   1965864  50% /lib
/Linux-i386/bin  4127076   1951568   1965864  50% /bin
/Linux-i386/sbin 4127076   1951568   1965864  50% /sbin
/Linux-i386/usr  4127076   1951568   1965864  50% /usr
/dev/xvdb        5160576   1427500   3523364  29% /
udev              131168       112    131056   1% /dev
tmpfs             131168         8    131160   1% /tmp
/dev/xvdj       20642428  10102248   9491604  52% /export/home
/dev/xvdk       20642428    176320  19417532   1% /export/opt
/dev/xvdl       30963708  20238400   9152444  69% /export/media
```

R/O OS with Xen

```
nehemiah:~ # df
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/xvda        4127076   1951568   1965864  50% /Linux-i386
/dev/xvdb        5160576   1427496   3523368  29% /
udev              131168       112    131056   1% /dev
tmpfs             131168         8    131160   1% /tmp
/dev/xvdj       20642428  10102248   9491604  52% /export/home
/dev/xvdk       20642428    176320  19417532   1% /export/opt
/dev/xvdl       30963708  20238400   9152444  69% /export/media
```

R/O OS with Xen

```
obadiah:~ # df
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/xvda        4127076   1951568   1965864  50% /Linux-i386
/dev/xvdb        4128448   1927680   1991056  50% /
udev               32864       104     32760   1% /dev
tmpfs              32864        16     32848   1% /tmp
```

R/O OS with Xen

```
disk=[ 'file:/var/vmachine/nehemiah/disk0.xvd,xvda,r',
       'phy:/dev/sysvg1/nehemiah,xvdb,w',


-rw------- 5 root root 4294967296 2011-03-25 09:07
                          /var/vmachine/nehemiah/disk0.xvd
```

Relocatable Packages

On-Demand Software, Ready to Run

Relocatable Packages

- Immediate deployment
- Simplified back-out
- Non-intrusive
- Multiple release concurrency
- Variable platform detail (per build)
- Reduced “scatter”
- Think ‘**`vmlink`**’

Relocatable Packages – versus today

currently (ie: read-write, not shared) …

- Packages [re]deployed on each system
- Deployment causes multiple disruptions
- Demands private (R/W) file storage
- Upgrade and/or removal is “messy”
- Installed files are vulnerable
- More things needing to be backed up

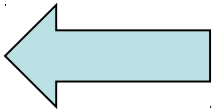
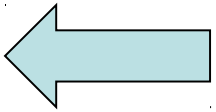
Relocatable Packages

we can (with shared read-only) …

- Deploy instantly
- Protected copies (R/O to each client)
- Less content to be backed up
- Non-intrusive (to the guest op sys)
- Non-disruptive (to the users and work)
- Mixed releases as needed

Relocatable Packages

sharing options …

- NFS
- SMB (SAMBA)
- VM minidisk  today
- SAN  future

R/O packages do not require R/O root

Relocatable Packages – How

- Separate software residence
  from software reference
- Inst must distinguish program from data
- Installation must tolerate R/O systems

Relocatable Packages – Concept

$APPROOT/bin

$APPROOT/lib

$APPROOT/*otherstuff*

APPROOT=/usr/opt/x3270-3.3

- Use *package-version* syntax or similar

Relocatable Packages – Build

What is the “standard recipe”?

- extract
- `./configure --prefix=$APPROOT`
- **`make`**
- **`make install`**

Relocatable Package Example

Build with the standard recipe:

- extract
- `./configure --prefix=/usr/opt/x3270-3.3`
- `make`
- `make install`

`/usr/opt` is ready and writable

Relocatable Package

```
$ ls -atl /home/trothr/x3270-3.3

drwxr-xr-x 6 trothr ... CYGWIN
drwxr-xr-x 6 trothr ... Linux-s390x
drwxr-xr-x 6 trothr ... Solaris-sparc
drwxr-xr-x 7 trothr ... x3270-3.3
lrwxrwxrwx 1 trothr ... src -> x3270-3.3
-rwxr--r-- 1 trothr ... makefile
-rwxr-xr-x 1 trothr ... setup
```

Relocatable Package Example

```
$ /home/trothr/x3270-3.3/setup

+ ln -s
  /home/trothr/x3270-3.3/Solaris-sparc
  /usr/opt/x3270-3.3
+ ln -s x3270-3.3 /usr/opt/x3270
+ ln -s /usr/opt/x3270/bin/x3270 /usr/bin/.
+ ln -s /usr/opt/x3270/bin/x3270if /usr/bin/.
+ ln -s /usr/opt/x3270/bin/pr3287 /usr/bin/.
```

Relocatable Pkgs – Multiple Versions

```
lrwxrwxrwx … gcc -> gcc-3.2.3
```
(production)

```
lrwxrwxrwx … gcc-3.2.3 ->
  /import/opt/gcc-3.2.3/Linux-s390x
lrwxrwxrwx … gcc-3.4 ->
  /auto/apps/gcc-3.4/Linux-2.6-s390x
```

- Simple **`PATH`** change to access the variant:

```
PATH=/usr/opt/gcc-3.4/bin:$PATH
```

Disk-Based Automounter

On-the-fly Mainframe Media

Disk Automounter: Purpose

Automate best practice media access

- z/VM supports dynamic devices
- Linux supports dynamic devices but with different semantics
- Automounter bridges the gap and eliminates operator error

Disk Automounter: Misconceptions

NOTE: DOES NOT REQUIRE NFS

- Most automounter is for networked FS
- Other FS also good for on-demand use (CD-ROM, flash media, USB disk, etc)
- No network requirement in automounter

Dynamic Disk on Linux on z/VM

How it works, manually:

- Attach the disk (‘**`hcp link`**’)
- Find where Linux slotted it
- Vary it on-line (‘**`chccwdev`**’)
- Mount it

Convoluted and error prone

Automating Disk Attachment

```
#
# /etc/auto.master
#
/home    /etc/auto.home
/misc    /etc/auto.misc
/dasd    /etc/auto.dasd
```

Automating Disk Attachment

```
# parse off the partition number, if any:
PART=`echo "$1" | awk -F. '{print $2}'`

# normalize the device number:
DASD=`echo "0000$1" \
   | awk -F. '{print $1}' \
   | tr A-Z a-z \
   | awk '{print "0.0."
               substr($1,length($1)-3,4)}'`
```

Automating Disk Attachment

```
# find the pseudo file to control this dev:
CTRL=`ls -d
   /sys/devices/css0/*/$DASD/online
   2>/dev/null | head -1`

# is the disk on-line (is it ATTACHed)?
if [ ! -f "$CTRL" ] ; then
   hcp "link * $DASD $DASD rr"
   # and re-set CTRL shell var as above
fi
```

Automating Disk Attachment

```
# vary it on-line to Linux:
echo 1 > $CTRL

# and find the block dev assigned:
BDEV=`ls -d
   /sys/devices/css0/*/$DASD/block
   2>/dev/null | head -1`
# also clean-up that file path
```

Automating Disk Attachment

```
# voi-la! create a directory and mount it
mkdir -p -m 555 $1
# mount command varies per the following
```

- Unqualified, try partition 0 or partition 1
- Qualified partition 1, 2, or 3, try as-is
- Qualified partition 0 is “the whole disk”

Disk Automounter Examples

```
zservx01:~ # df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/dasde1            7098008    817616   5919824  13% /
tmpfs                   124700         0    124700   0% /dev/shm
/dev/dasda1              52200      8940     40568  19% /boot
```

Initial state of the system

Disk Automounter Examples

```
zservx01:~ # cd /dasd/25f/sles9
zservx01:/dasd/25f/sles9 # df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/dasde1            7098008    817616   5919824  13% /
tmpfs                   124700         0    124700   0% /dev/shm
/dev/dasda1              52200      8940     40568  19% /boot
/dev/dasdg1           23216172  18301524   3735332  84% /dasd/25f
```

Automounter did the following:

- Found the “25F” disk, varied it on-line
- Found slot “dasdg” and partition 1
- Mounted FS in the expected location

Disk Automounter Examples

```
vst $ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/dasdb2             222464     98332    112648  47% /
/dev/dasda1              20908      8948     10880  46% /boot
/dev/dasda2            2126020    531716   1486304  27% /usr
/dev/dasda3             214096     27624    175420  14% /opt
tmpfs                   124700        20    124680   1% /tmp
/local/home             104608     34944     64264  36% /home
/local/var              104608     34944     64264  36% /var
```

Initial state (round two)

Disk Automounter Examples

```
vst $ cd /dasd/1bd.1 ; cd /dasd/1bd.2 ; cd /dasd/1bd.3
vst $ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/dasdb2             222464     98336    112644  47% /
/dev/dasda1              20908      8948     10880  46% /boot
/dev/dasda2            2126020    531720   1486300  27% /usr
/dev/dasda3             214096     27624    175420  14% /opt
tmpfs                   124700         0    124700   0% /tmp
/local/home             104608     34976     64232  36% /home
/local/var              104608     34976     64232  36% /var
/dev/dasdn1             849696     24752    781780   4% /dasd/1bd.1
/dev/dasdn2             566936      7140    530996   2% /dasd/1bd.2
/dev/dasdn3             948184     92696    807320  11% /dasd/1bd.3
```

The “doc disk”: **`man`**, **`info`**, **`doc`**

Automating DCSS Attachment

```
#
# /etc/auto.master
#
/home    /etc/auto.home
/misc    /etc/auto.misc
/dasd    /etc/auto.dasd
/dcss    /etc/auto.dcss
```

Summary

- Use Shared Filesystem images
- No need for partitioning
- Start with EXT2, maybe ISO-9660
- Put add-on data and software there
- Consider putting op sys there
- use XIP

Summary

- The real advantage is *not* storage savings but is management of myriad systems

Thank You!!

Rick Troth

Senior Developer

Velocity Software, Inc

<rickt@velocitysoftware.com>

http://www.velocitysoftware.com/