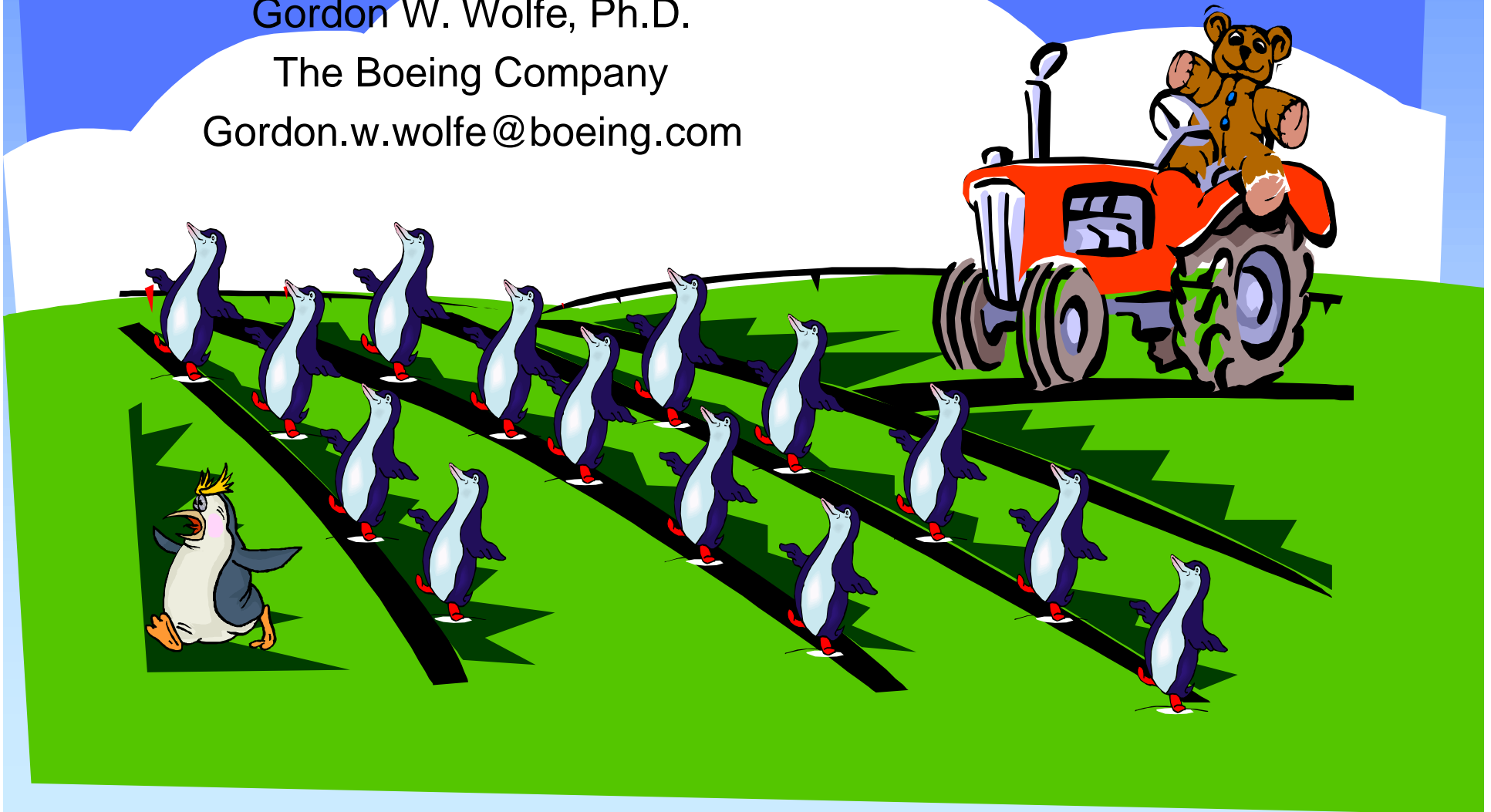# Managing a Penguin Farm on the VM Prairie

Share 99, San Francisco 2002

Session 9341

Gordon W. Wolfe, Ph.D.

The Boeing Company

Gordon.w.wolfe@boeing.com

# Disclaimer

- Use the contents of this presentation at your own risk. Neither the author nor the Boeing Company accept any liability, implicitly or explicitly, direct or consequentially, for use, abuse, misuse, lack of use, misunderstandings, mistakes, omissions, mis-information for anything in or not in, related to or not related to, pertaining or not pertaining to this document, or anything else that a lawyer can think of or not think of.

*BOEING*®

# Introduction

- VM originally created to run many operating systems on one real machine

- Perfect for running Linux guests.  VM acts as "hypervisor' or "Netbios" for Linux.

- Won't go into reasons for using VM or Linux here.

- Assume you are familiar with both VM and Linux.

- Presentation will be specific to z/VM 4.2.0 and SuSe Linux SLES7 as practiced at Boeing, with extensions by implication to other operating systems releases.

- Object here will be to discuss how to run MANY linuxes as VM guests and have a productive system

- I will sometimes note availability of some commercial software products.  I am not aware of them all.  Sorry if I missed anyone.

*BOEING*

# The Problem

- VM can support hundreds (or thousands) of virtual servers.  We estimate about 300 productive Linux servers per G5 IFL engine.

- When you get that many guest operating systems, how do you

    – Keep everything consistent?

    – Let them talk to each other and clients?

    – Handle updates?

    – Use VM's resources most efficiently?

- You need a Plan!

*BOEING*®

# Boeing's Implementation

- One 9672-R36 with one IFL Engine

- Six z/VM 4.2 LPARs - One for Linux

- Cisco routers and OSA Express

- SuSE SLES7 on 16 guests

*BOEING*®

# Consistency!

- Multiple Linux guests become impossible to manage if they are all different.

- Stick to one distribution, one release of Linux

- Try as much as possible to make every Linux guest work like every other Linux Guest.

- Keep similar files in the same places

# Have Written Policies

- **Have Formal Service Level Agreements**
  - Times of Operation/time of maintenance
  - Guaranteed levels of performance
  - Software levels
  - General agreement, not one for each server.
  - Help desk

*BOEING*®

# Have Written processes to follow

- For cloning a new server

- For fixing a damaged boot disk

- For upgrading software

- Adding disks, using LVM, etc.

*BOEING*®

# Customer takes responsibility for ALL maintenance and upgrades

- If customer needs own R/W /usr disk...

- If customer modifies the kernel…

- If customer modifies security arrangements

- If customer can't tolerate upgrades

- Will be notified by e-mail if new release/patches available on CD/Patch server

*BOEING*®

# Keep Policies and forms accessible on the web

- Written policy on how to obtain a server

- Lots of how-to documents for users (Make them do as much of the work as you can!)

  - Setting up KDE

  - Creating users and groups

  - Keeping Linux secure

  - Running Samba

  - Running Apache

  - Etc.

- Can use Linux Apache for this purpose!

*BOEING*

# Common 191 disk

- Owned by clone server

- PROFILE EXEC
  - Choose boot from DASD (default) or Reader
  - If boot from DASD, CMS FORMAT/RESERVE V-DISK
  - \<server\> EXEC to couple VCTCA's or set up OSA

- Contains files needed to boot from reader
  - INITRD
  - \<server\> PARMFILE
  - IMAGE

- See Appendix 1 in handouts

*BOEING*®

# Memory and Swap

- Linux swap to VM V-DISK in real/expanded storage
  - Means VM does real paging - more efficient!
  - Set up in PROFILE EXEC on common 191 disk
  - Have to modify one file on Linux
  - V-DISK defined in directory entry

- Keep Linux Memory to a minimum
  - Let it use its own swap
  - Only need 48MB for Apache, Samba server
  - Need 128 MB for Oracle beta
  - Need 750MB for WebSphere Commerce Suite!

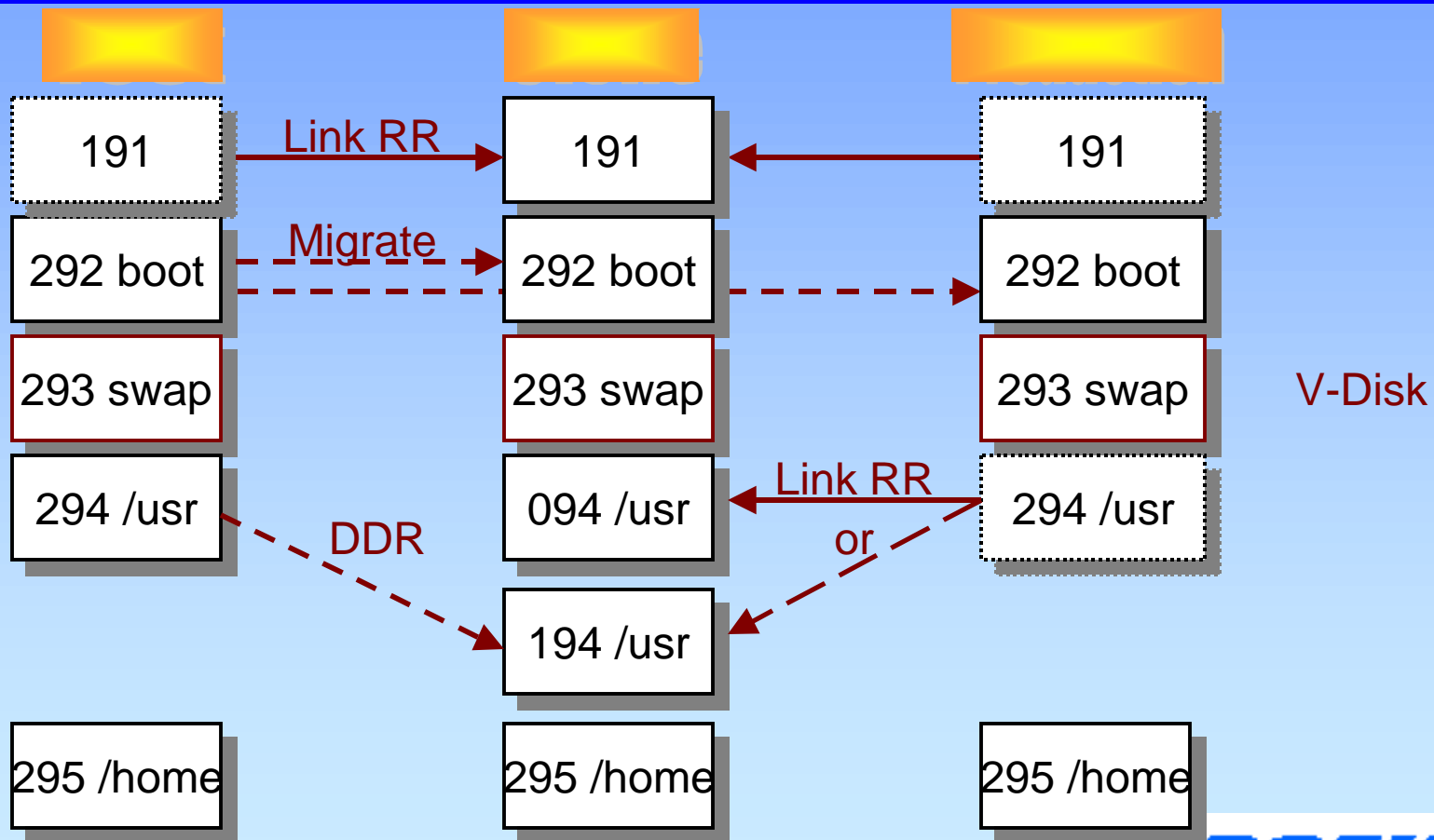- Booting from the reader requires at least 64MB!
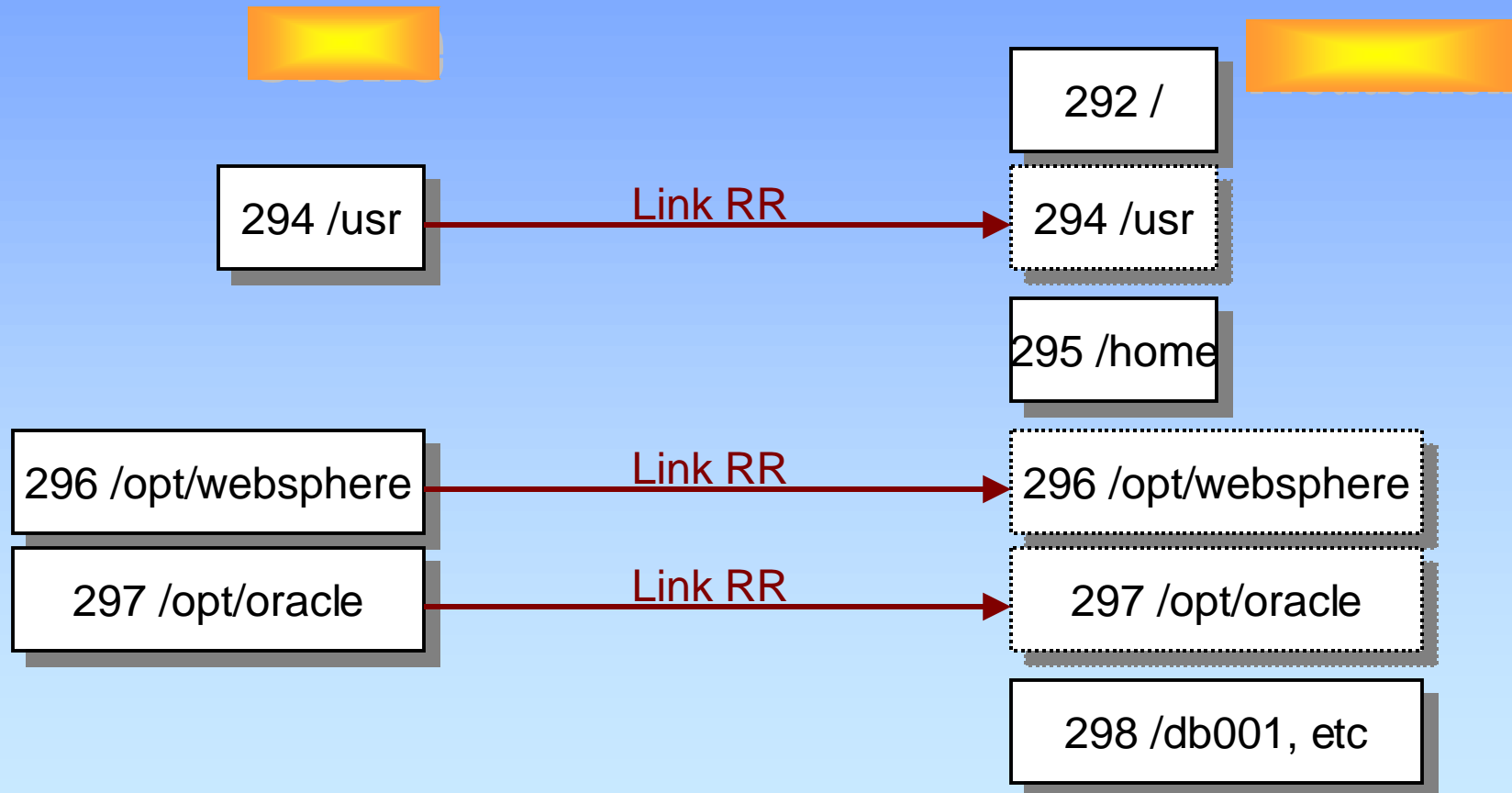
*BOEING*

# Patch and CD server

- Samba Server acessible r/o by any Linux administrator

  - Need to create userid with read privileges for FTP and SMBMOUNT on all servers

  - Use mount -t smbfs -o userid=xxx,password=yyy //patchsvr/share /mountpoint

- /cd     - to do installs of new software - copy install CDs to this directory

  - **/cd/SLES7**
    - **../CD1**
    - **../CD2**
  - **/cd/SLES7/develop.ed**
    - **../CD1**
    - **../CD2**

- /home/patches   - to install patches (Samba share /patches)

  - **Sudo-20020402.rp,**
  - **Gcc-20020326.rpm, etc.**

# Test, Clone and Production

# Different Filesystems for Different Applications

| | |
|---|---|
| | 292 / |
| 294 /usr | → Link RR → 294 /usr |
| | 295 /home |
| 296 /opt/websphere | → Link RR → 296 /opt/websphere |
| 297 /opt/oracle | → Link RR → 297 /opt/oracle |
| | 298 /db001, etc |

# Shared /usr and other Filesystems

- Disk(s) owned by clone server

- 094 for production , 194 for new/updated  self-link 194 as 294 RW

- Clone machine is shut down virtually all the time, except when updating files.

- Link clone 094 disk as 294 RR in directory of server

- /etc/fstab should mount /usr "ro"

- Parm line in /etc/zipl.conf should read dasd=0293,0292,294(ro),295-29F root=/dev/dasdb1

- Keep extra DASD devices in parm in case you need to add one later.

- When clone machine done updating 194, do
    - mount /usr -o ro,remount
    - Sync;sync
    - shutdown

- Same for other R/O filesystems

# Read/Write on a Read-Only Directory

- R/O user means some functions have to be moved to a R/W disk, e.g.

  - Apache Webserver

  - Move /usr/local/httpd to /home/httpd or other R/W  location

  - Update location in /etc/httpd/httpd.conf

*BOEING*

# Additional Files in a Read-Only Directory

- Create a new subdirectory on a R/W disk
  - Mkdir /home/mystuff

- Copy files from R/O directory (/usr/mystuff) to it
  - Cd /usr/mystuff
  - tar cf - . | tar xpf - -C /home/mystuff
  - Add or change anything you want to here

- Mount R/W subdirectory over R/O subdirectory cd /home
  - mount -o rw --bind /home/mystuff /usr/mystuff

# Cloning Penguins:  Part 1

- Create link exec, parm file on 191

- Set up TCPIP to talk to server

- Create the directory entry
  - Link RR to 191, /usr, other product disks

- DDR the boot disk from the clone server

# Cloning Penguins: Part 2

- **Boot the new server from the reader**
  - **Mount /dev/dasdb /mnt**
  - **Chroot /mnt**
  - **Mount /dev/dasdc /usr**
  - **dasdfmt /dev/dasdd (and fdasd, mke2fs)**
  - **Mount /dev/dasdd1 /home**
  - **Update files to make new server (SuSE)**
    - **/etc/rc.config**
    - **/etc/httpd/httpd.conf**
    - **/etc/route.conf**
    - **/etc/hosts**
    - **/etc/fstab**
    - **/etc/smb.conf**
    - **Run /sbin/SuSEconfig**
    - **Run zipl**
  - **Exit, umount everything and shut down**

*BOEING*

# Cloning Penguins:  Part 3

- Boot the new server from DASD

- Set up authorities to start and stop server

- Place DNS name in DNS server

- Whole thing takes about 2-3 hours.  Can be automated with CMS execs and shell scripts preinstalled on the clone server.

- Be Sure to keep a database of servers!

  - DNS, userid, IP address, owner-name, comm method, usage type

# Database

```
            LINUX SERVER DATABASE ENTRY AND DISPLAY SCREEN

  I INQUIRE, ADD, DELETE, CHANGE, EXIT


  SMK001                           LINUX SERVER NAME
  SMK001    VM USERID OF SERVER ON NODE  VLX1   VM MANAGER ID  IPSEDT
  DNS NAME OF SERVER  SMK001.CA.BOEING.COM


  IP ADDR  192.33.58.226        PEER IP ADDR  192.33.58.225
  CTC, OSA OR IUCV  CTCA  CTC/OSA ADDR IN  C16    CTC/OSA ADR OUT  C17
                   TCPIP CTC/OSA ADDR IN  C02    CTC/OSA ADR OUT  C03


  OWNER'S NAME  TRETTEVIK, ED
  OWNER'S EMAIL ADDR  ED.A.TRETTEVIK@BOEING.COM
  CWA CHARGED  BNLGAR     OWNER'S ORG NUMBER  G-4646


  LINUX RELEASE  SLES7       MANAGED BY  VM
      STANDARD?  Y


  DATE IMPLEMENTED  2001-07-03
  DATE DELETED      __/__/____


  ENTER - RUN PROCESS   PF2 - CLEAR SCREEN          PF3 - EXIT
```

# Cloning Software Available

- VMLINMAN
  http://www.glasshousesystems.com/home.html

- STK SNAPVANTAGE
  http://www.storagetek.com/prodserv/products/software/svan/

- Unsupported demo software from Tung-Sing Chong at IBM Endicott

  - Code at http://www.vm.ibm.com/devpages/chongts/

  - Instructions at http://www.vm.ibm.com/devpages/chongts/tscdemo.html

*BOEING*

# Starting Servers

- Create a server start/stop userid

- Allow SMSG commands via WAKEUP

- Have list of authorized users

- Exec to do actual AUTOLOG of server

- PROFILE EXEC in Clone machine defaults to "boot from disk" if no virtual console attached.

- User/owners of servers require second VM id to start system

# Stopping Servers (part 1)

- Put VMPOFF=LOGOFF into parm file /etc/zipl.conf to log off Linux userid when linux O/S quits

- User/owners of servers require second VM id to stop system

# Stopping Servers (Part 2)

- Use SECUSER facility, CP SEND command
  - Requires knowledge of root password (Require root password to be same as VM password for server, check through VMSECURE.  Advantage is VM sysprog always knows root password)
  - Have to test for completed login
  - Have to test for completed shutdown
  - Can be done using autolog server above, WAKEUP.  See Appendix 3

# Stopping Servers. Part 3

– With z/VM 4.3's SERVC facility and Linux 2.4.7,

- Linux can be patched to shut down automatically at CP SHUTDOWN or CP SIGNAL SHUTDOWN

- Put ca:12345:ctrlaltdel:/sbin/shutdown -t1 -h now in /etc/inittab

- Patch available from http://linuxvm.org/Patches/S390/linux-2.4.7-s390-quiesce.diffs

- For 2.4.17 kernel, go to http://www.marist.edu/htbin/wlvtype?LINUX-VM.28797

# TCPIP

- ## With CTCA and IUCV, use VM's TCPIP as virtual router

  - Linux guests must be subnet of VM IP address

  - Need one IP address for each server

  - Need only one additional address as "peer" address for all servers

  - No broadcast IP with this method

  - Linux prior to 2.4.7 has problems with FTP and IUCV.

  - VCTCA approx. 200 mb/sec, IUCV approx 400 mb/sec

# Multiple TCPIP stacks

- Use secondary TCPIP userids to main TCPIP

- Separate TCPMAINT 198 disk for each

- One IP address for each Linux, only one peer address for all Linuxes on this secondary stack.

- Multiple stacks for testing, production, alternate methods; better performance on multiprocessor machine

- Don't have to take down all Linux servers when testing methods on one server.

- Romney suggests optimum is 10-12 Linuxes per TCPIP stack

# Multiple TCPIP stacks
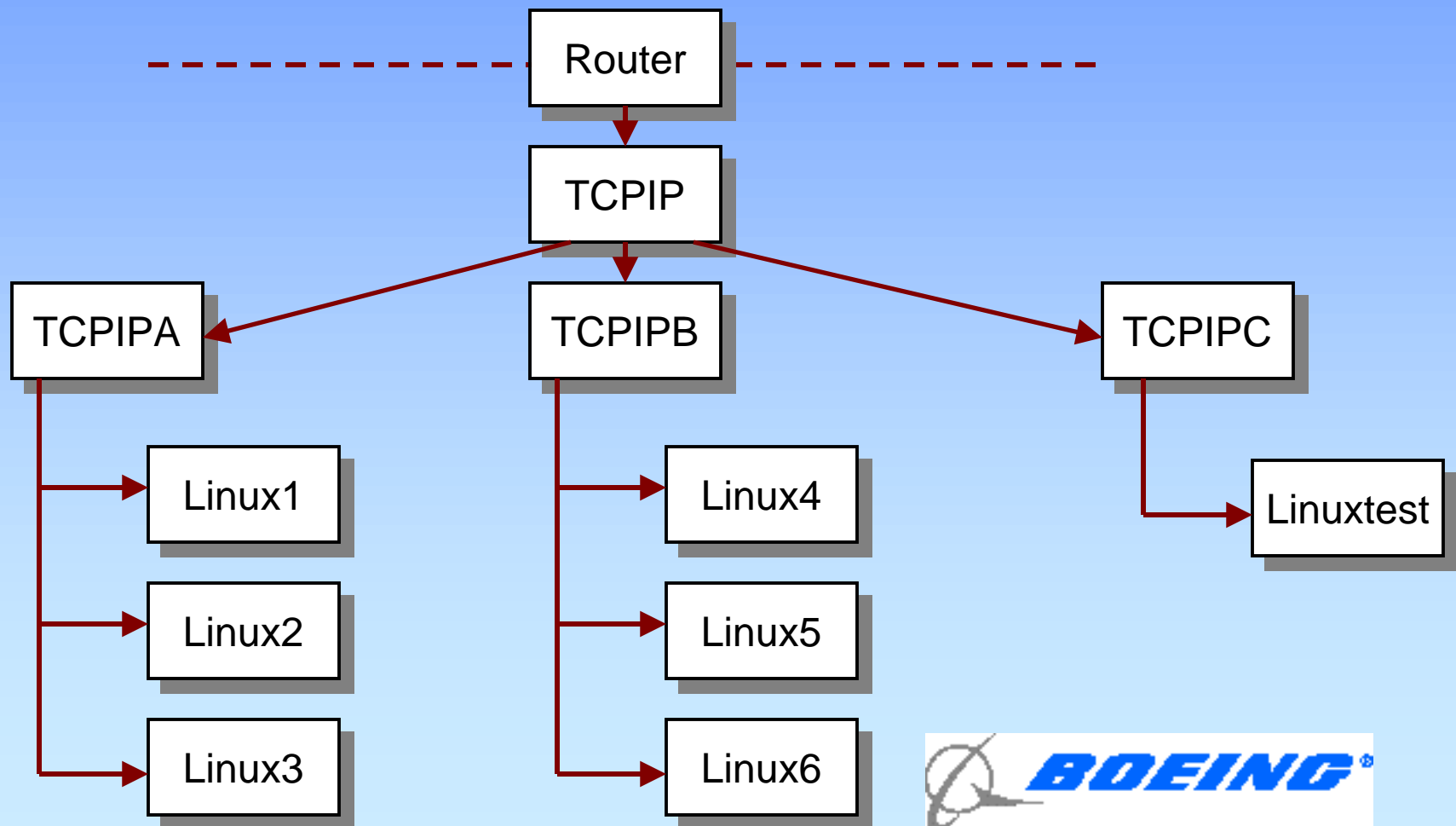
# Linux and OSA

- With OSA, let Linux own the OSA address and talk to real router
  - Requires extensive knowledge of OSA, which is a "black art".
  - Multiple Linux servers can share an OSA card
  - Gigabit speeds
  - Special parameters in /etc/zipl.conf
  - Red Hat does not use IBM's OCO code.  You have to add it.

*BOEING*

# Tuning Linux to work with VM

- Apply the on-demand timer patch from IBM Developerworks

- SuSE has it applied and ready to use as an rpm if you buy the service.

- Shared DASD such as /usr

- V-DISK for swap, minimum virtual storage (Requires change to Linux. See Appendix 2)

- MINIOPT CACHE RECORDMDC for real minidisks

- Use "dasdfmt cdl" or reserve first cylinder of pack for VM disk label:

  - MDISK 294 3390 1   3338 V163E1 MR LINUX   USR      DASDC

  - ........................^

- Use INCLUDE files and ACIGROUPS in the directory to better manage

- Stagger file-level backups of Linux servers.

# Updates and Changes

- Load patches, CD's onto a Linux Patch server

  - Easier to use than CD or Windows server

  - Makes it available to those Linux owners who do their own updates

- Update test server first

- When working, move to clone server, including 194 disk

- Route around to production servers

*BOEING*®

# Routing Updates and Changes

- Use Rsync and rsh to route new files around. (RPM -ql <package> gives list of all files contained in <package.rpm>)

- Ssh is more secure than rsync and rsh

- Sample scripts in appendix 4 for automated method.

- Shut down server and swap LINK LXCLONE 094 294 RR for LINK LXCLONE 194 294 RR

- Reboot

# Try to palm off Linux maintenance to someone else

- Unix gurus

- NT server people

- End user as last resort - use at least for creating own userids and groups

- Set up formal Service Level Agreements for "dumpees" to follow.

*BOEING*

# Backing up Linux with VM

- VM:Backup will do physical (track) backups of Linux, but must restore an entire filesystem disk set to recover one file.  E.g. all of /usr.

- Problem is compounded if using LVM for multiple-volume filesystem.  Have to restore ALL volumes of logical volume to restore one file!

- Same for DDR or any other VM backup system.  No file-level backups.

*BOEING*

# Backing up Linux with TSM

- TSM (Server on VM or z/OS) works well  for file-level backups

  - VM server release 3, z/OS at release 4 (maybe 5 by now)

  - but uses lots of network capacity, especially first time!

  - Don't use software compress in client machine.  CPU hog!

  - Can do full or incremental, keep multiple generations.

  - Stage to DASD first, then move to tape to minimize number of tape drives used.

*BOEING*

# Other ways to back up Linux at file level

- CA-Brightstar

- Other products on near horizon

# Roll your own Backups

- Give Linux server capability to use BFS, OpenExtensions in directory, BFS server

- Use Linux NFS, VMNFS server to mount BFS

- Use Tar to back up to NFS-mounted BFS

- Use VM:Backup to back up BFS

- Can do with 4 shell scripts and 3 files, but very labor intensive, requires root privilege, and uses LOTS of disk space!  (Contact me if you want the scripts)

- With Neale's cpint package and tape support in 2.4.7 (Both included in SuSE 2.4.7) can go direct from Linux direct to tape with "tar" or "dump", but tape scheduling is a nightmare.  Cpint can be used to SM VMTAPE MOUNT and CP DETACH

*BOEING*®

# Accounting

- Most Unix-type systems do not do job accounting

- Hooks and packages available but require extensive kernel mods

- Future kernel may have job accounting in it.

- Under VM it's simple!  Use a separate server for each account!  If you have to share data among servers, use NFS!

- VM:Account will create (with exits) and collect charge records for CPU, DASD, BFS, tape mounts, and so on.

- VM:Account will also do ad-hoc reports on usage.  Cumbersome to set up, but well worth it in the end.  Beware!  Users will want you to run reports for them!

*BOEING*®

# Close

- VM can be used as "hypervisor" for many linux guests

- The more guests you have, the more work maintaining them is.

- Many members of Linux-VM community have come up with some ideas for managing many servers.

- Commercial software solutions are still on the horizon.

- See Mark Post's LinuxVM page at http://linuxvm.org

- Join the Linux-390 Listserver! (address on LinuxVM page above)

*BOEING®*

# Appendix 1 - Starting up Linux

```
/*------------------------------------------------------------*/
/* Common Profile Exec for Linux Server Machine               */
/* By Gordon Wolfe, VM Technical Services          05/22/00*/
/*------------------------------------------------------------*/
address command

'SET RDYMSG SMSG'
'CP SET ACNT OFF'
'CP SET RUN ON'
'CP SET RELPAGE OFF'
'CP SET EMSG ON'
'CP LIMIT CLEAR CP'

'CP SET PF11 IMMED FILEL'
'CP SET PF12 RETRIEVE'
'CP TERMINAL LINESIZE 255'
'CP TERMINAL CHARDEL OFF'
'CP SET EMSG ON'

/* Set up for this particular linux server machine            */
'ESTATE' userid() 'EXEC A'
if rc <> 0 then exit rc
'EXEC' userid()

/* Determine if we start up Linux now.                        */
startflag = 'N'
iplflag   = 'D'

/* Are we running disconnected? if so, start linux.           */
'PIPE CP QUERY' userid() '| var usrline'
parse value usrline with . . term .
if term = 'DSC' then startflag = 'Y'
```

## Appendix 1 - Continued

```
/* Not disconnected? ask to start up.                          */
/* Also find out where to start up from.  Reader IPL or DASD ipl.    */
else do
   say 'Do you want to start up LINUX now? (Y/N)'
   pull ans .
   if left(ans,1) = "Y" then startflag = 'Y'
   if startflag = 'Y' then do
      say 'Do you want to IPL from DASD or from the reader? (D/R)'
      say 'The default is to IPL from DASD.'
      pull ans .
      select
        when left(ans,1) = 'R' then iplflag = 'R'
        when left(ans,1) = 'D' then iplflag = 'D'
        when ans         = ''  then iplflag = 'D'
        otherwise do
          say ans 'is an invalid choice.'
          exit 8
        end
      end
   end
end

if startflag = 'Y' then do
   if iplflag = 'R' then queue 'EXEC SLES7IPL'
   if iplflag = 'D' then do
     queue '1'
     queue 'LXSWAP'
     'FORMAT 293 E ( BLK 4096'          /* format/reserve V-disk      */
     if rc <> 0 then exit rc            /* for swap space             */
     queue '1'
     'RESERVE LINUX SWAP E6'
     if rc <> 0 then exit rc
     queue 'EXEC IPLDASD'
   end
end
exit
```

# Appendix 1 - Continued

```
/* Sample LINUX002 EXEC for Linux userid LINUX002  */
address command
'CP COUPLE C16 TO TCPIP C20'
'CP COUPLE C17 TO TCPIP C21'

(Parmfile file Linux userid LINUX002)
ramdisk_size=32768 root=/dev/ram0 ro ctc=0,0xC16,0xC17,ctc0

/* IPLDASD EXEC */
ADDRESS COMMAND
TRACE o
'CP CLOSE RDR'
'CP PURGE RDR ALL'
'CP DETACH 190'
'CP DETACH 19E'
'CP IPL 292 CLEAR'

/* SLES7IPL EXEC                                       */
/* exec to IPL Linux from the reader and run from a ramdisk */
/* by Gordon Wolfe, VM Technical Services      08/17/01  */
address command
trace o

/* Do we have enough virtual storage to do this?           */
'MAKEBUF'
buf1 = rc
'EXECIO 1 CP ( STRING Q V STOR'
pull . . stor .
'DROPBUF' buf1
stor = STRIP(stor,'L','0')
stor = STRIP(stor,'T','M')
if stor < 64 then do
  say 'Virtual storage must be 64M or greater to IPL from reader'
  say 'Perform CP DEF STOR 64M and IPL CMS.'
  exit 8
end
```

# Appendix 1 - Continued

```
/* Do we have the files we need?                              */
'ESTATE' userid() 'PARM *'
if rc <> 0 then do
  say 'File' userid() 'PARM * not found.'
  exit 28
end

/* Find the filemode for the files we need                    */
'PIPE CMS LISTF SLES7 IMAGE * |',
  'take 1 |',
  'var imageloc'
parse value imageloc with . . fm .
fm = left(fm,1)

/* All looks okay, proceed.                                   */
'CP CLOSE RDR'
'CP PURGE RDR ALL'
'CP SPOOL PUN * R'
'PUNCH SLES7 IMAGE' fm '( NOH'
'PUNCH' userid() 'PARM' fm '( NOH'
'PUNCH SLES7 INITRD' fm '( NOH'
'CP CHANGE RDR ALL KEEP NOHOLD'
'CP IPL 00C CLEAR'
```

# Appendix 2 - Modified File in Linux for swapping to V-Disk

```
SuSE Linux 2.2.16
   /sbin/init.d/boot
   At about line 153, a change was made to change swap from that listed
   in fstab to use the FB-512 V-disk at /dev/dasda1.  This device should
   have been CMS FORMAT/RESERVEd by CMS before IPLing linux.  Then at
   boot time a mkswap is done on /dev/dasda1 and swapon is used to start
   swapping on this device. The lines in /sbin/init.d/boot should read:


           #cho "Activating swap-devices in /etc/fstab..."
           #wapon -a &> /dev/null
           echo "Creating swap file signature"
           mkswap /dev/dasda1
           echo "Activating swap partition"
           swapon /dev/dasda1

SuSE SLES7 2.4.7
   This process has been moved to /etc/init.d/boot and is done at line
   200.
```

# Appendix 3 - Shutting Down Linux from VM

```
/* EXEC to send shutdown commands to a Linux guest                 */
/* Assumes root password is same as Linux VM userid password.      */
/* Also assumes server has a parmline containing vmpoff=LOGOFF      */
/* Userid running this exec must be a VMSECURE administrator        */
/* By Gordon Wolfe, VM Technical Services              03/09/2001*/

address command

arg server .
'PIPE CP Q SECUSER' server '| drop 2 | var bkupsecuser'
if rc <> 0 then do
  say server 'is an unknown Linux server.'
  exit 8
end
parse var bkupsecuser with oldsecuser .
if oldsecuser = 'not' | oldsecuser = 'NOT' then oldsecuser = 'OFF'


line = 'shutdown -h now'

/* Set the secondary userid to ourself.                            */
'CP SET SECUSER' server userid()

/* Get the console logged on as root                               */
call linuxpwd server
if result <> 0 then exit result

/* And send the shutdown command                                   */
'CP SEND' server line

/* Clean up and quit.                                              */
done:
'CP SET SECUSER' server oldsecuser
exit
```

# Appendix 3 - Continued

```
linuxpwd: procedure
address command

arg linuxmach .
if linuxmach = '' | linuxmach = 'LINUXMACH' then do
  say 'No Linux machine specified'
  return 8
end

/* Get the password for the server       in the proper case.        */
rootname = 'root'
call getpass linuxmach
if result = 28 then do
  say 'no password on file for' linuxmach
  return 8
end
else pw = lowercas(result)


do i=1 to 3
  'CP SEND' linuxmach rootname
  'CP SLEEP 1 SEC'
  'CP SEND' linuxmach pw
  'CP SLEEP 1 SEC'
end
return 0



getpass: procedure
/* procedure to query VMSECURE for the password of the server  */
arg finduser .
```

# Appendix 3 - Continued

```
'ERASE LINUX TEMP A'
'MAKEBUF'
buf1 = rc
queue 'SSAVE LINUX TEMP A'
queue 'QQUIT'
'VMSECURE EDIT' finduser '( NOPROF'
if rc <> 0 then do
  say finduser 'not found in VMSECURE'
  'ERASE LINUX TEMP A'
  return 28
end
'DROPBUF' buf1
'PIPE < LINUX TEMP A |',
  'locate /USER / |',
  'take 1 |',
  'specs word 3 1 |',
  'var pw'
'ERASE LINUX TEMP A'
return pw


/* Lowercas                                                   */
/* translates input argument to lowercase                     */
/* By Gordon Wolfe, Vm Technical Services          06/23/98 */
lowercas: procedure
arg inp
out = translate(inp,'abcdefghijklmnopqrstuvwxyz','ABCDEFGHIJKLMNOPQRSTUVWXYZ','.
return out
```

# Appendix 4 - Routing Updates to Servers
# From a Central Maintenance Server

```
File /root/updates/hosts - Place the names of the hosts to which updates will be routed
in this file.
#Nickname in hosts file or DNS name     VM userid
#-------------------------------        ---------
clone                                   LINUX000
test                                    LINUX001
patch                                   LINUX002
webserver                               LINUX003


File /root/updates/files - Place the fully-qualified filenames of files that will be routed
to the above hosts:
/etc/hosts
/etc/profile.local
product-20020505.rpm


File /root/updates/commands - Place commands that you want the above hosts to execute
id
rpm -Uvh product-20020505.rpm


Allow the central server in /etc/hosts.allow on each server to be updated
in.rsyncd:192.100.200.1
in.rshd:192.100.200.1


In EACH server to be updated, include a /root/.rhosts file with the name of the central
server:
(assuming being updated from the root userid)
test.lnx.boeing.com root

Then run /root/updates/update to send all files to all hosts then execute all commands on
all hosts:
```

# Appendix 4 - Continued

```
rm runupdate
rexx updates.rex $1
chmod 770 runupdate
/root/updates/runupdate

This calls the rexx program updates.rex, which is

/* updates.rex                                              */
/* An exec to create a shell script to take a list of files      */
/* from the file ./files and send them to a group of linux servers  */
/* listed in a file named ./hosts                           */
/* Then execute a number of commands taken from a file ./commands  */
/* Called from shell script "update"                        */
/* Assumes:                                                 */
/* file names in ./files are fully qualified path names        */
/* host names in ./hosts are resolvable through /etc/hosts     */
/* the shell script created will be run from userid root       */
/* Userid root on the receiving host has a /root/.rhosts file    */
/* Both transmitting and receiving host allow use of rsh.      */


trace off
signal off error

parse arg onehost .
if onehost <> '' then say "Processing for" onehost "only."

/* First, get names of files into stem variable              */
jfiles = 0
oldq = queued()
"cat files >FIFO"
do until queued() = oldq
  parse pull file
  if left(file,1) = "#" then iterate
  jfiles = jfiles + 1
  parse value file with fname.jfiles fnewname.jfiles .
  if fnewname.jfiles = '' then fnewname.jfiles = fname.jfiles
end
```

# Appendix 4 - Continued

```
/* Next, get commands to execute into stem variable                */
jcmds  = 0
oldq = queued()
"cat commands >FIFO"
do until queued() = oldq
  parse pull file
  if left(file,1) = "#" then iterate
  jcmds  = jcmds  + 1
  parse value file with command.jcmds
end


/* Next get list of hosts to send files to                         */
oldq = queued()
nhosts=0
"cat hosts >FIFO"
do until queued() = oldq
  parse pull line
  if left(line,1) = "#" then iterate
  nhosts=nhosts+1
  parse value line with hostname.nhosts vmuserid.nhosts .
  if onehost = hostname.nhosts then khost=nhosts
end
if onehost <> '' then do
  vmuserid.1 = vmuserid.khost
  nhosts = 1
  hostname.1 = onehost
end
```

# Appendix 4 - Continued

```
/* for those hosts that are logged on, build a script              */
/* Is the host logged on?                                          */
do j=1 to nhosts
  oldq = queued()
  "hcp q" vmuserid.j ">FIFO"
  do until queued() = oldq             /* get just last line       */
    parse pull line
  end
/*    Host is not logged on.  Ignore it for now.                   */
  if strip(line) <> 'Ready;' then say '+++' hostname.j 'not running.'
  else do
/*    Host is indeed running.                                      */
/*    Build the shell script to actually send the files            */
    host = hostname.j
    "echo -e echo -e                    "  ">> runupdate"
    "echo -e echo -e Sending files to" host ">> runupdate"
    do i=1 to jfiles
      newline = "rsync -pogt -r -l" fname.i
      newline = newline host":"fnewname.i
      "echo -e" newline ">> runupdate"
    end

/*    Build the shell script to execute the commands               */
    "echo -e echo -e Executing commands on" host ">> runupdate"
    do i=1 to jcmds
      newline = "rsh"
      newline = newline host command.i
      "echo -e" newline ">> runupdate"
    end                               /* do i=1 to jcmds          */
  end                                 /* else do                  */
end                                   /* do j=1 to nhosts         */
exit
```