Virtual Servers: An Overview

Ben Rockwood, Cuddletech <benr@cuddletech.com>

Revision History
Revision v1.0 August 10th 2002 br
Initial document

...

Table of Contents

Introduction

Virtual servers are being more and more popular, and the trend will only continue as computing power does. The concept is simple, the running of multiple operating enviroments (or OS is you prefer) on a single system. This is not at all a new idea, in fact it's what made Linux/Apache famous (Apache virtual servers), in the enterprise space anyway, and as old as time in the mainframe world. For years people have run emulators, namely for running Windows on MacOS or Solaris, using applications like SoftWindows. Then when VMware became popular on Linux many people used it not only to run Win32 on Linux, but also to run multiple independant Linux distributions ontop of their primary Linux system for testing and isolation purposes. But the way in which virtualization is accomplished is changing. Mainframes aside, full X86 emulation is the tradition but this requires a large ammount of resource and doesn't leverage the already running system. As systems are getting more powerful it is becoming increasingly appealing to virtualize.

Using Apache ISP's have the power to host 20 domains on 1 system, rather than hosting 20 domains on 20 diffrent servers. Each Apache virtual server has an independant root, as far as the user is concerned, providing no evidence that it's mearly one among many. What if we apply this same priciple to an operating enviroment such as Linux or Solaris. A common problem in development enviroments individual developers want root access to the development system, what if you had a way to provide each developer with their

own individual operating enviroment complete with root access? What if you would consolidate down 5 poorly utilized systems into 1 without having to worry about the applations playing nice with each other? What if you would create servers could be recovered from a root compromise in a matter of seconds? All these things and more are possible by leveraging server virtualization.

In this article I want to look at the current state of server virtualization, and look at some of the most popular and prevelent tools used today. From this point forward I will refer to server virtualization as operatiing enviroment virtualization (OEV), which is I think a better fitting term for the pracitcle of providing complete operating enviroments rather than simple services and to avoid confusions with other uses of the same term. While reading this please bear in mind that most of the things discused here are new or currently in developement and likely to change significantly. I do not espouse to be an expert on the topic, nor do I think anyone at this early stage can, but none-the-less I feel that an overview of this topic is warrented, even if it is a poor one. Lets start by looking at whats avalible on Linux and the move on to the "Big Three" in the enterprise space.

Linux Virtualization

Linux is naturally a hot bed for diffrent ideas on how to approach virtualization. Lets look at the three most popular and promising packages currently avalible: User-Mode Linux, VServer, and Xen. I should note that while VMware technically could be used for OEV, I do not consider it a viable option as it is not intended for replacing real servers for indefinate periods of time, nor does it provide an adiquate mechanism for managing a large number of concerant virtual systems.

User-Mode Linux

User-Mode Linux (UML) was introduced to assist in developing and debuging the Linux kernel. UML can be done by patching any given Linux kernel (some don't even need patching) source tree, and building it using the "um" architecture instead of your normal hardware arch. Once you've built a UML kernel it can be started just like any program reguardless of your permissions on the primary system. This makes UML particularly interesting to users without root access to the system. A filesystem is provided to the UML kernel by way of a root filesystem image. This also makes UML ideal for testing diffrent distributions without ever having to reboot the primary system.

Here is a simple example of UML in action:

```
benr@nexus6 UML$ ls -lh
total 711M
-rwxr-xr-x    1 benr     benr          32M Mar 10 02:42 linux
-rw-------    1 benr     benr         679M Mar 14 17:23 root_fs.rh-7.2-full.pristine.20020
benr@nexus6 UML$ ./linux ubd0=root_fs.rh-7.2-full.pristine.20020312 ubd1=swap mem=92M
Checking for the skas3 patch in the host...not found
Checking for /proc/mm...not found
```

```
tracing thread pid = 4297
Checking for /dev/anon on the host...Not available (open failed with errno 2)
Checking for /dev/anon on the host...Not available (open failed with errno 2)
Checking for /dev/anon on the host...Not available (open failed with errno 2)
Checking for /dev/anon on the host...Not available (open failed with errno 2)
Linux version 2.4.24-1um (benr@nexus6) (gcc version 3.3.2 20031218 (Gentoo Linux 3.3.2-r5,
On node 0 totalpages: 23552
zone(0): 23552 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: ubd0=root_fs.rh-7.2-full.pristine.20020312 ubd1=swap mem=92M root=/de
Calibrating delay loop... 1192.75 BogoMIPS
Memory: 88976k available
Dentry cache hash table entries: 16384 (order: 5, 131072 bytes)
Inode cache hash table entries: 8192 (order: 4, 65536 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 32768 (order: 5, 131072 bytes)
Checking for host processor cmov support...Yes
Checking for host processor xmm support...No
Checking that ptrace can change system call numbers...OK
Checking that host ptys support output SIGIO...Yes
Checking that host ptys support SIGIO on close...No, enabling workaround
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
VFS: Disk quotas vdquot_6.5.1
devfs: v1.12c (20020818) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x1
JFFS version 1.0, (C) 1999, 2000  Axis Communications AB
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis Communications AB.
pty: 256 Unix98 ptys configured
SLIP: version 0.8.4-NET3.019-NEWTTY (dynamic channels, max=256).
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
PPP generic driver version 2.4.2
Universal TUN/TAP device driver 1.5 (C)1999-2002 Maxim Krasnyansky
SCSI subsystem driver Revision: 1.00
scsi0 : scsi_debug, Version: 0.61 (20020815), num_devs=1, dev_size_mb=8, opts=0x0
  Vendor: Linux     Model: scsi_debug        Rev: 0004
  Type:   Direct-Access                      ANSI SCSI revision: 03
blkmtd: error: missing `device' name

Initializing software serial port version 1
mconsole (version 2) initialized on /home/benr/.uml/J0ytHU/mconsole
Partition check:
 ubda: unknown partition table
unable to open swap for validation
UML Audio Relay (host dsp = /dev/sound/dsp, host mixer = /dev/sound/mixer)
Initializing stdio console driver
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 8192 bind 16384)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
VFS: Mounted root (ext2 filesystem) readonly.
Mounted devfs on /dev
INIT: version 2.78 booting
                       Welcome to Red Hat Linux
                Press 'I' to enter interactive startup.
```

```
Mounting proc filesystem:  [  OK  ]
Configuring kernel parameters:  [  OK  ]
Setting clock : Mon Mar 15 05:54:58 EST 2004 [  OK  ]
Activating swap partitions:  [  OK  ]
Setting hostname redhat72.goober.org:  [  OK  ]
Your system appears to have shut down uncleanly
Press Y within 1 seconds to force file system integrity check...
Checking root filesystem
/dev/ubd/0 was not cleanly unmounted, check forced.
/dev/ubd/0: Inode 23855, i_blocks is 88, should be 32.  FIXED.
/dev/ubd/0: Inode 59046, i_blocks is 128, should be 96.  FIXED.
/dev/ubd/0: Inode 66780, i_blocks is 64, should be 8.  FIXED.
/dev/ubd/0: Inode 67351, i_blocks is 72, should be 16.  FIXED.
/dev/ubd/0: Inode 67357, i_blocks is 64, should be 8.  FIXED.
/dev/ubd/0: 56179/86976 files (0.1% non-contiguous), 154522/173824 blocks
[/sbin/fsck.ext2 (1) -- /] fsck.ext2 -a /dev/ubd/0
[PASSED]
Remounting root filesystem in read-write mode:  [  OK  ]
Finding module dependencies:  depmod: cannot read ELF header from /lib/modules/2.4.24-1um/
depmod: cannot read ELF header from /lib/modules/2.4.24-1um/modules.generic_string
depmod: /lib/modules/2.4.24-1um/modules.ieee1394map is not an ELF file
depmod: /lib/modules/2.4.24-1um/modules.isapnpmap is not an ELF file
depmod: cannot read ELF header from /lib/modules/2.4.24-1um/modules.parportmap
depmod: /lib/modules/2.4.24-1um/modules.pcimap is not an ELF file
depmod: cannot read ELF header from /lib/modules/2.4.24-1um/modules.pnpbiosmap
depmod: /lib/modules/2.4.24-1um/modules.usbmap is not an ELF file
[FAILED]
Checking filesystems
Checking all file systems.
[  OK  ]
Mounting local filesystems:  [  OK  ]
Enabling local filesystem quotas:  [  OK  ]
swapon: cannot stat /dev/ubd/1: No such file or directory
Enabling swap space:  [  OK  ]
INIT: Entering runlevel: 3
Entering non-interactive startup
Setting network parameters:  [  OK  ]
Bringing up interface lo:  [  OK  ]
SIOCADDRT: No such device
SIOCADDRT: Network is unreachable
Starting system logger: [  OK  ]
Starting kernel logger: [  OK  ]
Starting portmapper: [  OK  ]
Loading system font:  [  OK  ]
Initializing random number generator:  [  OK  ]
Mounting other filesystems:  [  OK  ]
Starting identd: [  OK  ]
Starting snmpd: [  OK  ]
Starting named: [  OK  ]
Starting sshd:
Starting xinetd:
Starting sendmail: [  OK  ]
Starting console mouse services: (no mouse is configured)
Starting httpd: [  OK  ]
Starting crond: [  OK  ]
Starting squid:  [  OK  ]
Starting xfs: [  OK  ]
Starting SMB services: [  OK  ]
Starting NMB services: [  OK  ]
Please run makehistory and/or makedbz before starting innd.
Running Linuxconf hooks:  [  OK  ]
```

```
Unauthorized access to this system is strictly prohibited.
redhat72 login: root
Password:
Last login: Fri Mar 12 03:20:48 on vc/0
bash-2.05# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/ubd/0            668M  594M   41M  94% /
bash-2.05# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 05:54 ?        00:00:00 init [3]
root         2     1  0 05:54 ?        00:00:00 [keventd]
root         3     1  0 05:54 ?        00:00:00 [ksoftirqd_CPU0]
root         4     1  0 05:54 ?        00:00:00 [kswapd]
root         5     1  0 05:54 ?        00:00:00 [bdflush]
root         6     1  0 05:54 ?        00:00:00 [kupdated]
root         7     1  0 05:54 ?        00:00:00 [scsi_eh_0]
root         8     1  0 05:54 ?        00:00:00 [mtdblockd]
root       317     1  0 05:55 ?        00:00:00 syslogd -m 0
root       322     1  0 05:55 ?        00:00:00 klogd -2
rpc        332     1  0 05:55 ?        00:00:00 portmap
ident      390     1  0 05:56 ?        00:00:00 identd -e -o
ident      393   390  0 05:56 ?        00:00:00 identd -e -o
ident      397   393  0 05:56 ?        00:00:00 identd -e -o
ident      399   393  0 05:56 ?        00:00:00 identd -e -o
ident      400   393  0 05:56 ?        00:00:00 identd -e -o
root       404     1  0 05:56 ?        00:00:00 /usr/sbin/snmpd -s -l /dev/null -P /var/ru
named      413     1  0 05:56 ?        00:00:00 named -u named
named      415   413  0 05:56 ?        00:00:00 named -u named
named      416   415  0 05:56 ?        00:00:00 named -u named
named      417   415  0 05:56 ?        00:00:00 named -u named
named      418   415  0 05:56 ?        00:00:00 named -u named
root       429     1  0 05:56 ?        00:00:01 /usr/sbin/sshd
root       445     1  0 05:56 ?        00:00:00 xinetd -stayalive -reuse -pidfile /var/run
root       462     1  0 05:56 ?        00:00:00 sendmail: accepting connections
root       482     1  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     483   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     484   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     485   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     486   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     487   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     490   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     491   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
apache     492   482  0 05:56 ?        00:00:00 /usr/sbin/httpd -DHAVE_PROXY -DHAVE_ACCESS
root       500     1  0 05:56 ?        00:00:00 crond
root       518     1  0 05:56 ?        00:00:00 squid -D
squid      524   518  1 05:56 ?        00:00:02 (squid) -D
squid      531   524  0 05:56 ?        00:00:00 (unlinkd)
xfs        556     1  0 05:56 ?        00:00:00 xfs -droppriv -daemon
root       566     1  0 05:56 ?        00:00:00 smbd -D
root       600     1  0 05:56 vc/1     00:00:00 /sbin/mingetty ttys/1
root       601     1  0 05:56 vc/2     00:00:00 /sbin/mingetty ttys/2
root       602     1  0 05:56 tts/0    00:00:00 /sbin/mingetty serial/0
root       604     1  0 05:58 vc/0     00:00:00 login -- root
root       605   604  9 05:58 vc/0     00:00:00 -bash
root       638   605  0 05:59 vc/0     00:00:00 ps -ef
bash-2.05# cat /proc/cpuinfo
processor       : 0
vendor_id       : User Mode Linux
model name      : UML
mode            : tt
host            : Linux nexus6 2.4.25-vs1.26 #2 SMP Wed Mar 10 03:20:29 PST 2004 i686
bogomips        : 1192.75
```

```
bash-2.05# shutdown -h now

Broadcast message from root (vc/0) Mon Mar 15 06:02:50 2004...

The system is going down for system halt NOW !!
INIT: Switching to runlevel: 0
INIT: Sending processes the TERM signal
Shutting down xfs: [  OK  ]
Stopping httpd: [  OK  ]
Stopping squid:  [  OK  ]
Stopping sshd:[  OK  ]
Shutting down sendmail: [  OK  ]
Shutting down SMB services: [  OK  ]
Shutting down NMB services: [FAILED]
Stopping named: [  OK  ]
Stopping snmpd: [  OK  ]
Stopping xinetd: [  OK  ]
Stopping crond: [  OK  ]
Stopping identd: [  OK  ]
Saving random seed:  [  OK  ]
Stopping portmapper: [  OK  ]
Shutting down kernel logger: [  OK  ]
Shutting down system logger: [  OK  ]
Starting killall:  Shutting down SMB services: [FAILED]
Shutting down NMB services: [FAILED]
[FAILED]
Sending all processes the TERM signal...
Sending all processes the KILL signal...
Saving mixer settings aumix:  error opening mixer

Syncing hardware clock to system time
Turning off quotas:
umount2: Device or resource busy
umount: devfs: not found
umount: /dev: Illegal seek
Halting system...
Power down.

benr@nexus6 UML$
```

As you can see above, you execute the kernel just like any ordinary program, and as a benifit of that, ordinary debuggins tools like GDB can be used to debug kernels as they boot. Setup is fairly straight forward and simple, but networking can be alittle tricky.

Setting Up UML

Here is a quick example of setting up UML.

1. Start by downloading the latest UML patch from the UML patch page [http://user-mode-linux.sourceforge.net/dl-2.4-patches-sf.html] and then it's matching kernel from your favorite kernel mirror.

2. Unpack the kernel source and patch it.

```
benr@nexus6 user-mode$ tar xfvj linux-2.4.24.tar.bz2
 [ Output removed for clarity ]
benr@nexus6 user-mode$ cd linux-2.4.24
benr@nexus6 linux-2.4.24$ cd linux-2.4.24
benr@nexus6 linux-2.4.24$ patch -p1 < ../uml-patch-2.4.24-1
 ...
```

3. Now you can build you UML kernel using the "um" architecture. Make sure to build in virtual networking devices along with any other options you want built in. Modules can be used with UML, but are beyond the scope of this paper. Stripping (removing the debugging info) your kernel once built will bring the size of the kernel down to a more normal size.

```
benr@nexus6 linux-2.4.24$ make menuconfig ARCH=um
 [ Configure the kernel like your used to ]
benr@nexus6 linux-2.4.24$ make linux ARCH=um
 [ Output removed for clarity ]
benr@nexus6 linux-2.4.24$ ls -lh linux
-rwxr-xr-x    1 benr     benr          32M Mar 16 01:36 linux
benr@nexus6 linux-2.4.24$ strip linux
benr@nexus6 linux-2.4.24$ ls -lh linux
-rwxr-xr-x    1 benr     benr         2.2M Mar 16 01:50 linux
```

4. Download the latest UML-Utilities [http://user-mode-linux.sourceforge.net/dl-tools-sf.html] source, unpack, build and install it.

```
benr@nexus6 user-mode$ tar xfvj uml_utilities_20040114.tar.bz2
 [ Output removed for clarity ]
benr@nexus6 user-mode$ cd tools/
benr@nexus6 tools$ make
benr@nexus6 tools$ su
Password:
root@nexus6 tools$ make install
```

5. UML is generally uses a root image for it's root file system. Because UML will appear as a complete system we'll need all the usual things in the root filesystem. The easiest method to get started is to download a prebuilt root image. A wide variety are provided for download. [http://user-mode-linux.sourceforge.net/dl-fs-sf.html] Choose and download an image, then uncompress it and preferably put it in the same directory as your UML kernel. If your just experimenting I'd recommend root_fs_toms, the *tomsrtbt* single floppy distribution weighing in at 1.4MB.

6. You are now ready to start UML, at a mimimum you must specify a block devices

for the root file system. Swap devices can added as a zero'ed file (swap image) or use system swap. You can constrain memory usage using the "mem=32M" argument. Also add any boot arguments you need.

```
benr@nexus6 user-mode$ dd if=/dev/zero of=swap.img bs=1M count=16
16+0 records in
16+0 records out
benr@nexus6 user-mode$ mkswap -f swap.img
Setting up swapspace version 1, size = 16773 kB
benr@nexus6 user-mode$ ./linux ubd0=root_fs_toms1.7.205 ubd1=swap.img mem=32M
 [ Output removed for clarity ]
Linux version 2.4.24 (benr@nexus6) (gcc version 3.3.2 20031218 (Gentoo Linux 3.3.2-r5,
 [ Output removed for clarity ]
INIT: Entering runlevel: 5


Welcome to the uml version of Tom's root/boot.
 ttys/0 tomsrtbt login: root
Password: (root)
Today is Setting Orange, the 2nd day of Discord in the YOLD 3170

# swapon /dev/ubd/1
Adding Swap: 16376k swap-space (priority -1)
# cat /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  29552640 12472320 17080320        0   126976  1372160
Swap: 16769024        0 16769024
 [ Output removed for clarity ]
```

This procedure can be changed significantly to provide for diffrent needs, but should get you started with UML. Networking can also be provided to UML, but the proccess is more involved than I'd like to discuss here.

Thoughts on UML

User-Mode Linux provides an "easy" way to get started with EOV on Linux. While useful for a variety of diffrent testing and development needs it still has that hackish feel. Setting up networking for UML is complicated and confusing, and not at all well suited for production OEV. Lacking a central management interface provides some flexablity, such as running it as an unprivelaged user, however makes managing multiple UMLs exceedingly difficult. After playing with UML you get the feeling that it's best used as a tool, rather than a solution. This isn't to say that with some serious scripting and loving care you couldn't overcome some of these problems, but doing so would push the limits of what UML was really ment to do, making other solutions much more appealing.

The Linux VServer Project

The Linux Vserver project is.....