User-mode Linux (UML)

January 15, 2005
Ryan Brown
The Burgh Live, LLC
What is UML?

- UML is a “safe and secure way of running Linux versions and Linux processes.”
- UML provides you with a virtual machine that may have more resources than your physical computer.
- UML allows you to run Linux in Linux.
- UML is FREE!!!
Where Did UML Come From?

February 1999 by Jeff Dike (jdiike on IRC).
Constant new development and updates (new kernels).
Open Source Project
Why use UML?

- Safe and secure way for development and testing software upgrades
- Maximize resources
- Test latest kernels
- Training and education
- Honeypots
- Virtual Private Server Provider
What is Required for UML?

- 2.2.15 or newer kernel on a Linux machine

*The requirements are very basic as you can see. The more RAM, CPU speed, and disk space you have the better, as you can run large and multiple virtual servers.*
UML Vocab

Host – the machine which runs the virtual servers.

Guest – the virtual machine produced from UML.

SKAS – Separate Kernel Address Space
Kernel Modifications

- Download the kernel source code you wish to use (2.4 or 2.6).
- Download UML patch from UML web site (see resources at end).
- Patch your kernel source, and configure kernel: make menuconfig ARCH=um.
- Finally, make linux ARCH=um, to create the UML binary.
Add-on Patches

SKAS – Separate Kernel Address Space

Originally each process in a UML guest session had its own process (TT or tracing Thread) on the host. With SKAS a guest only has 5 processes. Huge speed increase as SKAS eliminates the signal delivery that was previously seen on every UML guest process. SKAS is a host kernel patch.
SKAS

- Processes cannot address kernel memory
- 2 context switches vs 4 with TT mode, so much faster.
### Host processes for a UML guest named “tblive”

<table>
<thead>
<tr>
<th>Process ID</th>
<th>Priority</th>
<th>Status</th>
<th>Time</th>
<th>Version</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>12964 pts/4</td>
<td>S+</td>
<td></td>
<td>2:27</td>
<td>linux-2.4.28 (tblive)</td>
<td>[rm]</td>
</tr>
<tr>
<td>12966 pts/4</td>
<td>T+</td>
<td></td>
<td>0:28</td>
<td>[linux-2.4.28]</td>
<td></td>
</tr>
<tr>
<td>12972 pts/4</td>
<td>S+</td>
<td></td>
<td>0:00</td>
<td>linux-2.4.28 (tblive)</td>
<td>[rm]</td>
</tr>
<tr>
<td>12973 pts/4</td>
<td>S+</td>
<td></td>
<td>0:02</td>
<td>linux-2.4.28 (tblive)</td>
<td>[rm]</td>
</tr>
<tr>
<td>12974 pts/4</td>
<td>S+</td>
<td></td>
<td>0:00</td>
<td>linux-2.4.28 (tblive)</td>
<td>[rm]</td>
</tr>
</tbody>
</table>
RAM and Storage

UML guests use a single file for each partition. Most UML guests have a root_fs file containing / and a swap_fs file for swap. Can be stored on NFS or SANs.

RAM is allocated during startup. A RAM disk is also used on the host machine, mounted as /tmp which drastically increases the guest’s performance.
Networking

- There are currently five transport types available for a UML virtual machine to exchange packets with other hosts:
  - ethertap
  - TUN/TAP
  - Multicast
  - a switch daemon
  - slip
  - slirp
  - pcap

- The TUN/TAP, ethertap, slip, and slirp transports allow a UML instance to exchange packets with the host. They may be directed to the host or the host may just act as a router to provide access to other physical or virtual machines.
Networking Cont.

With so many host transports, which one should you use? Here's when you should use each one:
- **ethertap** - if you want access to the host networking and it is running 2.2
- **TUN/TAP** - if you want access to the host networking and it is running 2.4. Also, the TUN/TAP transport is able to use a preconfigured device, allowing it to avoid using the setuid uml_net helper, which is a security advantage.
- **Multicast** - if you want a purely virtual network and you don't want to set up anything but the UML
- **a switch daemon** - if you want a purely virtual network and you don't mind running the daemon in order to get somewhat better performance
- **slip** - there is no particular reason to run the slip backend unless ethertap and TUN/TAP are just not available for some reason
- **slirp** - if you don't have root access on the host to setup networking, or if you don't want to allocate an IP to your UML
- **pcap** - not much use for actual network connectivity, but great for monitoring traffic on the host
Host Setup for Networking

- Make sure TUN/TAP is in kernel

```bash
ifconfig eth0 0.0.0.0 up promisc
brctl addbr vds-bridge
brctl setfd vds-bridge 0
brctl sethelo vds-bridge 0
brctl stp vds-bridge off
ifconfig vds-bridge 192.168.1.10 netmask 255.255.255.0 up

brctl addif vds-bridge eth0
route add default gw 192.168.1.1 dev vds-bridge

tnct1 -u tblive -t vds-tblive0
ifconfig vds-tblive0 0.0.0.0 promisc up
brctl addif vds-bridge vds-tblive0
```
Making the root_fs

Easiest way is to download a root_fs. There are a lot of guides about creating your own and utilities (GUI and console) for making them.

Next, create a new empty file system using “dd” (any size you want) and format the file with your desired (ext2, ext3, etc).

Mount original to /mnt/loop and new file system to /mnt/loop2.

cp –a /mnt/loop /mnt/loop2
COW

- Copy on Write
- 1 master root_fs to boot everyone’s (assuming same distro) UML guest
- Individual storage files used to record only changes
- The more guest sessions you have, the more disk space you will save. For example, a 1 GB root_fs without using COW will use 30 GB for 30 guests, or only 1 GB if you use COW.
Starting UML

- Start in a screen session

2.4 Kernel
linux-2.4.28 con0=fd:0,fd:1 ubd0=/home/tblive/tblive_root_fs
    ubd1=/home/tblive/tblive_swap_fs mem=64M eth0=tuntap,vds-tblive0 umid=tblive

2.6 Kernel
linux-2.6.9 con0=fd:0,fd:1 ubd0=/home/tblive/tblive_root_fs
    ubd1=/home/tblive/tblive_swap_fs mem=64M eth0=tuntap,vds-tblive0 umid=tblive
Inside the Guest UML

Kernel

rbrown@shadowcat [~]# uname -a
Linux shadowcat.lax.tblive.com 2.6.7-1um #2 Fri Jul 2 07:35:01 PDT 2004 i686 i686 i386 GNU/Linux

Top

rbrown@shadowcat [~]# top
18:13:00 up 23:46, 1 user, load average: 0.28, 0.16, 0.08
51 processes: 49 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 0.8% user 2.9% system 0.1% nice 0.0% iowait 96.0% idle
Mem: 125760k av, 118248k used, 7512k free, 0k shrd, 17560k
Problems with UML

Debian Testing with 2.6 kernel. MAJOR problems and crashes.
You must rename /lib/tls to /lib/tls-disabled.
UML does not yet support Thread Local Storage (TLS) which is required by Native POSIX Thread Library (NPTL). You need to disable TLS and everything works.
Network bridging
You must make the network bridges BEFORE starting the UML guest. If you add the bridges after the guest has been started, networking will work, but once the guest is rebooted the bridge will fail to work and network access is lost.
MConsole
get the kernel version
add and remove devices
halt or reboot the machine
send SysRq commands
pause and resume the UML
make online backups without shutting down the UML
receive notifications of events of interest from within UML
monitor the internal state of the UML
More Utilities

- UML_Switch (networking)
- UML_Moo

Resize and manage COW files.

- UMLazi (http://www.umlazi.org)

Management tool for configuring and deploying new UML guests.
UML Competition

- VMWare ($1700 - $30,000)
- Virtuozzo ($1000 + )
- Linux Vserver (Free)
UML Resources

Official Site
http://user-mode-linux.sf.net

http://uml.openconsultancy.com
http://www.usermodellinux.org

IRC
irc.oftc.net  #uml