Massimiliano Leone
kosmiko@logorroici.org

# "Quelli che il uml... attività varie ed eventuali..."

hackit 2005
Napoli, 17-06-2005

# What is this ?

user-mode-linux è una patch per il kernel linux, che permette di eseguirlo come un processo user-space, già all'interno di un host linux.

Con appositi tool che emulano switch e cavi di rete, si può addirittura simulare una rete di host linux uml che dialogano tra loro.

# Perche UML ?

L'utilità è ovvia:

(1) possibilita' di effettuare prove in un ambiente protetto

(2) possibilita' di simulare reti con tanti nodi

(3) in ambito didattico: e' possibile far provare agli utenti l'ebbrezza dell'essere amministratori di una loro macchina, pur non disponendo in tal senso della macchina ospite (UML infatti può essere eseguito da qualunque utente)

(4) possibilita' di creare "gabbie" dentro cui far girare servizi che possono dare problemi di sicurezza, ancor piu' completamente che con chroot.

# Theory (1)

UML non è una macchina virtuale.

UML, invece, provvede solo a proseguire le chiamate di sistema del sistema ospite al kernel del sistema reale; fornisce, altresi, hardware emulato sulla base di quello reale.

# Theory (2)

UML prevede 2 modalità di “funzionamento”:

(1) TracingThread: per ogni processo interno ad uml, nell'host reale viene incrementato il numero di istanze uml: istanze, peraltro, mappate nella parte alta della memoria dell'host ospite.

...avido di risorse e poco sicuro...

# Theory (3.1)

(2) **SKAS**: separa il codice di da quello dei suoi processi, fornendo 2 spazi di indirizzamento appositi, come avviene in un host reale.
In tal modo i processi interni a uml non possono accedere al kernel stesso uml, e non si ha una mappatura 1:1 dei processi uml nel sistema reale, ma saranno solo 4 i processi avviati da uml:

- L'<*UML kernel thread*>, che gira in kernel-space in un'area di memoria riservata, esegue il codice del kernel del sistema emulato, e provvede ad intercettare le chiamate di sistema dei processi interni a UML.

# Theory (3.2)

- L'<*UML userspace thread*>, che esegue tutti i processi UML e provvede a mapparli in user-space nel sistema reale ad ogni context-switch (interno ad UML).
- L'<*ubd driver asynchronous IO thread*>, che si occupa della gestione dell'immagine del file system che usa UML.
- The <*write SIGIO emulation thread*>, che infine si occupa di gestire l'interazione con l'utente, proseguendo i comandi digitati da quest'ultimo al sistema UML.

# Theory (4)

La modalità SKAS, tuttavia, non è ancora presente nel kernel vanilla, ed è necessario patchare appositamente quest'ultimo: sicchè, per motivi di retro-compatibilità, la TT-mode  non è stata ancora definitivamente eliminata. Si può, d'altra parte, compilare un kernel UML con entrambi i supporti, e se il supporto SKAS sarà rilevato, UML verrà avviato in questa modalità.

**SKAS è cosa buona e giusta!**

In SKAS-mode:

- l'avvio di ogni kernel si riduce dai circa 50 ai circa 15 sec
- il numero di istanze simultanee aumenta da 3-5 ai 15-30, senza che l'host ospite ne risenta particolarmente.

# ...into the game: cow fs

La modalità “cow" è una funzionalità che permette di  emulare intere reti locali con una sola immagine del filesystem.

Il <fs> è, infatti, un file aperto in scrittura sul quale il processo agente crea un “lock”, e l'unica soluzione per avviare molti kernel uml insieme è avere tanti fs quanti i processi avviati (e...ARGH!, l'fs può essere anche di svariate centinaia di Mb).

Con la modalità “cow”, invece, si può replicare sempre l'unico filesystem esistente, usando come istanza un file di estensione .cow, dove vengono salvate le "differenze": peraltro, non più di qualche Mb...

esplicativo l'esempio seguente:

- normal mode:: "./linux ubd0=immagineFilesystem <altreOpzioni>"
- cow mode: "./linux ubd0=fs1.cow,immagineFilesystem"

# ...into the game: net(1)

Esistono varie modalità per
creare una rete di host uml;
le più diffuse sono:

- TUN/TAP:
  tramite un interfaccia virtuale tun sull'host ospite, l'uml parla con “il resto del mondo;
  mini-howto:
  - <host>modprobe tun;
  - uml cmdline: ethX=tuntap,,,IP_GATEWAY
  - <uml> ifconfig DEV IP up

# ...into the game: net (2)

- switch daemon:
  si possono far comunicare più di 1 host,
  tramite una pipe;
  il team di uml fornisce umlnet;
  in alternativa c'è o vde (http://vde.sf.net);
  mini-howto:
  - <host> vde_switch -daemon;
  - uml cmdline: ethX=daemon,,,$pipe_file
  - <uml>: ifconfit DEV IP up

# INTERMEZZO

INTERMEZZO:

menuconfig && compile

src e co. su

ftp://saltatempo.hackit.05
www.kernel.org
http://user-mode-linux.sf.net
http://kem.p.lodz.pl/~peter/qnet/

# umlhandler

umlhandler è un tool che si compone di 4 script, utili se si preferisce non impazzire quando si vuole avviare un numero (più o meno) elevato di uml, con certi parametri e altro ancora

# umlhandler (2)

(1) umlHandler:
    a command line interface to start and stop a certain number of uml kernels.
(2) umlStop:
    a command line interface to stop a certain number of uml kernels
(3) umlSetup-1:
    parse certain arguments from /proc/cmdline and set-up system
(4) umlSetup-2:
    parse a “command” argument from /proc/cmdline and exec it on boot

# umlhandler (3)

http://utenti.lycos.it/k0smik0/area51/umlhandler/

e su:

http://uml-handler.sourceforge.net/

# Bibliografia

http://user-mode-linux.sourceforge.net

# Bibliografia

http://user-mode-linux.sourceforge.net/

http://www.google.com

.... :-)

# License