Eines de virtualització lliures per a sistemes GNU/Linux

VII Jornades Programari de Lliure

Sergio Talens-Oliag
sto@iti.upv.es
Institut Tecnològic d'Informàtica (ITI)

4 de juliol de 2008

Resum

En aquest document parlarem de què són i com funcionen els sistemes de virtualització i les eines relacionades. Un cop introduïts els conceptes bàsics comentarem en més detall quines són algunes de les solucions de virtualització disponibles en les distribucions de GNU/Linux actuals (fonamentalment per a **Debian GNU/Linux** i derivades, tot i que també funcionen en altres distribucions), amb una petita explicació de les seues característiques, casos d'ús i mode de funcionament.

Índex

1 Què és la virtualització?

Abans de començar a parlar dels diferents tipus de virtualització caldria concretar a què ens referim quan parlem de *virtualització*, ja que es tracta d'un terme molt genèric que es por fer servir per a referir-se a coses diferents (veure http://en.wikipedia.org/wiki/Virtualization).

En aquest document parlarem de la *virtualització de plataforma* o *virtualització de servidors*, és a dir, la capacitat d'executar en un únic equip físic, l'*amfitrió* (*host* en anglès), múltiples *sistemes operatius convidats* (*guests* en anglès).

La idea bàsica és tindre la possibilitat d'executar programari d'usuari dins d'un entorn virtual sense tindre que modificar-lo.

Parlem de programes d'usuari perquè en alguns dels models de virtualització dels quals parlarem pot ser necessari modificar el sistema operatiu *convidat*, tot i que els programes d'usuari funcionen sense fer cap canvi.

2 Tipus de virtualització

En els punts següents enumerarem els diferents tipus de virtualització existents explicant com funcionen i donant exemples de sistemes reals que els implementen.

2.1 Emulació o simulació del hardware a nivell d'aplicació

Una aplicació simula el *hardware* complet, permetent l'execució de sistemes operatius sense modificar.

L'execució es fa sota el control de l'emulador que fins i tot simula l'execució de les instruccions a nivell de la CPU, és a dir, executa codi binari per a un tipus de CPU concret en un sistema real que fa servir un processador i un joc d'instruccions diferent al del sistema emulat.

El inconvenient d'aquest model de virtualització és que la simulació és molt lenta (per a cada instrucció del sistema emulat pot ser necessari executar entre 100 i 1000 instruccions a la CPU real), tot i que en alguns casos no és un problema gran (per exemple l'emulació de sistemes dels anys 80 en hardware actual funciona molt més ràpidament que en els equips originals).

Exemples:

- **Bochs**: http://bochs.sourceforge.net/
- **MAME**: http://mamedev.org/
- **QEMU**: http://bellard.org/qemu/

2.2 Virtualització completa o nativa sense suport hardware

Aquest tipus de sistemes fan servir una màquina virtual que fa d'intermediària entre el sistema convidat i el hardware real.

El *software* de virtualització és conegut generalment com a *monitor de màquina virtual* (VMM, *Virtual Machine Monitor*) o *hipervisor* (*hypervisor*).

En aquest tipus de sistemes l'*hipervisor* s'encarrega d'emular un sistema complet i analitza dinàmicament el codi que vol executar el sistema *convidat*, reemplaçant les instruccions crítiques (les quals cal virtualitzar) per noves seqüències d'instruccions que tenen l'efecte desitjat en el hardware virtual, mentre que les instruccions no crítiques s'executen tal qual a la CPU real.

Aquest tipus de sistemes permeten l'execució de sistemes operatius sense modificar.

Exemples:

- **VirtualBox**: http://www.virtualbox.org/
- **VMWare**: http://www.vmware.com/

2.3 Virtualització completa o nativa amb suport hardware

Aquest tipus de sistemes funcionen de manera similar als *sistemes de virtualització completa sense suport hardware*, però aprofiten tecnologies incorporades a les noves generacions de microprocessadors com les d'*Intel* (Intel-VT, VT-x per a 32 bit i VT-i per a 64 bit) i *AMD* (AMD-V), de manera que és possible executar el codi del sistema operatiu *convidat* sense modificar-lo.

En aquests sistemes el que es fa és executar l'*hipervisor* o VMM amb el màxim nivell d'accés a la CPU (*Anell -1* en processadors AMD i Intel) i els sistemes *convidats* s'executen a un nivell inferior (*Anell 0* en processadors AMD i Intel, que era el màxim nivell d'execució quan els processadors no incorporaven suport per a la virtualització).

Amb la introducció d'un nivell superior al que ja feien servir els sistemes reals aconseguim que no siga necessari fer cap canvi als sistemes *convidats*, però ara eixos sistemes no tenen accés als dispositius reals i és la CPU qui avisa al VMM quan es volen executar instruccions per accedir als dispositius des dels sistemes *convidats* i és l'*hipervisor* qui s'encarrega de donar l'accés als dispositius virtuals o reals que corresponguen.

Exemples:

- **KVM**: http://kvm.qumranet.com/

2.4 Paravirtualització

Són sistemes similars als de *virtualització completa*, que executen el sistema *convidat* amb un *hipervisor* que s'executa sobre el sistema real.

La diferència amb el model anterior és que en aquest tipus de virtualització es modifica el sistema operatiu *convidat* per incloure instruccions relacionades amb la virtualització, de manera que en lloc que l'*hipervisor* capture les instruccions problemàtiques, és el sistema *convidat* qui crida directament a l'*hipervisor* quan es necessari.

Evidentment, amb independència de les modificacions del *nucli del sistema convidat*, els programes d'usuari es poden executar sense cap canvi.

Un cas extrem d'aquest model de virtualització és el de l'UML (*User-mode Linux*), en el qual el nucli del sistema *convidat* es transforma en una aplicació a nivell d'usuari que fa la funció d'*hipervisor* i s'encarrega d'emular el hardware, tot i que ho fa a nivell de la interfície del sistema operatiu i no a nivell d'interfície física (com l'accés als dispositius dins del sistema *convidat* sempre es fa a través del *nucli* no cal emular el hardware, només la visió que tenen d'ell els programes d'usuari).

El gran problema d'aquest model és que cal modificar el sistema operatiu *convidat*, cosa que no és possible en alguns casos (per exemple, com modifiquem el codi del **Windows XP** per a que funcione amb *paravitualització*?).

Exemples:

- **User-mode Linux**: http://user-mode-linux.sourceforge.net/
- **Xen**: http://www.xen.org/

2.5 Virtualització a nivell de sistema operatiu

En aquest tipus de sistemes només executem un nucli (el de l'*amfitrió*) i aquest nucli crea *entorns d'execució* que les aplicacions veuen com a *màquines virtuals*.

En principi en aquest tipus de sistemes no cal emular el hardware a baix nivell, ja que en realitat és el mateix sistema operatiu qui controla els dispositius físics. El que sí es sol fer és incloure suport per a tindre dispositius virtuals com discs o targetes de xarxa dins de cada entorn d'execució.

La idea és que els programes s'executen en un entorn que fa creure a les aplicacions que es troben en un sistema independent quan en realitat comparteixen recursos amb altres màquines virtuals, tot i que el sistema organitza les coses per evitar que els entorns s'interfereixen entre ells.

Aquest és un dels models de virtualització més econòmic, ja que no necessita suport del *hardware* ni fa falta supervisar el codi a baix nivell, però té l'inconvenient que només permet executar entorns

virtuals per a la mateixa CPU i Sistema Operatiu i en realitat només hi ha un nucli, de manera que si eixe nucli té un problema totes les màquines virtuals es veuen afectades.

Exemples:

- **Linux-VServer**: http://www.linux-vserver.org/
- **OpenVZ**: http://www.openvz.org/

3 Sistemes de virtualització lliures per a GNU/Linux

En aquest apartat comentarem les característiques mes interessants d'alguns dels sistemes de virtualització disponibles per a GNU/Linux.

3.1 Bochs (<http://bochs.sourceforge.net/>)

És un emulador d'arquitectures basades en `x86` que funciona en múltiples plataformes; el major interès de **Bochs** és que és capaç d'emular un PC complet, incloent-hi els perifèrics i funciona en pràcticament qualsevol sistema amfitrió (per exemple es pot fer servir per a emular un PC en un Linux que s'execute en una arquitectura *PowerPC*, *Alpha*, *SPARC* o *MIPS*).

El problema d'aquest sistema és que és molt lent, tot i que les darreres versions van millorant la velocitat d'emulació fent servir tècniques d'optimització com les descrites l'article *Virtualization without Direct Execution or Jitting: Designing a Portable Virtual Machine Infrastructure* de Darek Mihocka i Stanislav Shwartsman, disponible des de l'URL:

http://bochs.sourceforge.net/Virtualization_Without_Hardware_Final.pdf

3.2 QEMU (<http://bellard.org/qemu/>)

QEMU és un emulador similar a **Bochs** que té unes quantes característiques interessants.

Per començar, QEMU té dos modes de funcionament: emulació de sistema complet i emulació en mode usuari.

En el mode d'emulació de sistema complet el programa emula un equip sencer (per exemple un PC basat en microprocessadors `x86` o `x86_64`) incloent-hi múltiples processadors i perifèrics. Aquest mode es fa servir per a executar sistemes operatius complets. En les darreres versions del programa es suporten més de 15 arquitectures diferents.

En el mode d'emulació a nivell d'usuari el programa pot executar programes compilats per una CPU concreta en un sistema que fa servir una CPU diferent; això es pot fer servir, per exemple, per a executar el **Wine** en una arquitectura no *Intel*.

Per a les arquitectures `x86` el QEMU suporta l'ús d'un mòdul d'acceleració per a sistemes amfitrions *Linux* i *Windows* que permet l'execució de part del codi que s'executa en els sistemes *convidats* directament en la CPU real, fent que el *QEMU* funcione com un sistema de *virtualització nativa* en lloc de com un emulador.

3.3 KVM (<http://kvm.qumranet.com/>)

KVM (*Kernel Virtual Machine*) és una solució de virtualització completa en la que fem servir el nucli de Linux com a *hipervisor*, de manera que tant el control dels dispositius reals com la planificació de tasques i la gestió de memòria del sistema *amfitrió* les fa el nucli de Linux.

En aquest model les màquines virtuals són processos normals del sistema (per això la gestió de memòria i la planificació de processos son les estàndard del sistema) als que afegim un mode d'execució addicional (*convidat*) a banda dels modes d'execució estàndards de Linux (*usuari* i *nucli*).

Així, una màquina virtual tindrà tres modes d'execució:

- Mode *convidat*: serà el mode d'execució normal per al codi del sistema *convidat* sempre que no tinga operacions d'entrada/eixida.
- Mode *usuari*: només el farem servir per a executar les operacions d'entrada/eixida del sistema *convidat*, ens permetrà gestionar dispositius virtuals a nivell d'usuari.
- Mode *nucli*: es farà servir per a entrar a treballar en mode *convidat* i per gestionar les eixides des de mode *usuari* causades per operacions especials o d'entrada/eixida.

Quant a la implementació del sistema, el KVM està format per dos components:

- Un controlador de dispositius per a gestionar el hardware de virtualització, accessible des del dispositiu `/dev/kvm` (inclòs al nucli de Linux des de la versió 2.6.20, amb suport per a microprocessadors Intel i AMD amb suport de virtualització).
- Un programa d'usuari que emula el hardware del PC (actualment es fa servir una versió modificada de `qemu`) que s'encarrega de reservar la memòria de la màquina virtual i crida al controlador anterior per executar codi en mode *convidat*.

Un dels avantatges d'haver fet servir el **QEMU** com a component d'espai d'usuari és que la gestió de l'entrada/eixida és la mateixa que a l'emulador i per tant podem fer servir els mateixos dispositius virtuals que funcionen amb el **QEMU**.

El fet de que el suport per al KVM estiga integrat en les versions oficials del nucli i que siga el sistema de virtualització preferit de distribucions com **RedHat** o **Ubuntu** fan que KVM siga una tecnologia a considerar a curt i mig termini per a fer *virtualització nativa* amb Linux.

3.4 User-mode Linux (<http://user-mode-linux.sourceforge.net/>)

Podríem dir que **User-mode Linux** (UML) és una aplicació que només es pot executar sobre sistemes GNU/Linux i que ens proporciona un sistema operatiu Linux virtual.

Tècnicament l'UML és una adaptació del nucli de Linux com les que es fan per a poder executar-lo en diferents processadors, amb la diferència que en aquest cas és una adaptació a la interfície software definida pel nucli i no a la interfície hardware definida per l'arquitectura física.

En realitat l'UML el que fa es transformar un nucli pensat per a executar-se sobre un sistema físic en una aplicació de nivell d'usuari en la que tots els dispositius son virtuals, amb els seus avantatges e inconvenients.

Un sistema virtualitzat amb UML és més lent que un sistema de virtualització a nivell del sistema operatiu, ja que estem executant el nucli com a procés, però per altra banda tenim l'avantatge de que estem segurs de que la màquina virtual està clarament aïllada del sistema real i d'altres màquines virtuals com ella, la qual cosa ens dóna moltes garanties respecte a les conseqüències dels problemes generats pel codi que s'executa dins de cadascuna de les màquines virtuals.

3.5 Xen (<http://www.xen.org/>)

Xen és una solució de *paravitualització* que implementa un *hipervisor* que s'executa en el nivell més privilegiat de la màquina i que bàsicament es fa càrrec de la planificació de tasques i de la gestió de memòria, delegant la gestió de l'Entrada/Eixida en un *convidat privilegiat especial* (anomenat *domain 0* o *dom0* en **Xen**) que arranca sempre que llancem l'*hipervisor* i que en les distribucions de GNU/Linux que inclouen **Xen** és normalment una versió modificada del nucli de Linux. Tot i que no ens afecta, és important indicar que també es possible fer servir versions modificades de **NetBSD** i **Solaris** com a nucli per al *dom 0*, és a dir, que en la pràctica **Xen** no és un sistema de virtualització lligat al nucli de Linux.

La idea bàsica darrere d'aquest model de funcionament és que així el codi de l'*hipervisor* és més senzill i lleuger, tot i que actualment i donada la complexitat de les CPU (*multithreading*, *multicore*, etc.) i de la gestió de memòria, cada cop el tema de la simplicitat és menys evident.

Quan es fa servir en una CPU que no suporta virtualització a nivell hardware, **Xen** necessita que es modifique el codi del sistema operatiu que s'executarà damunt seu, de manera que no és possible executar sistemes com **Windows XP** en una CPU que no done suport hardware a la virtualització.

Si la CPU suporta virtualització l'*hipervisor* de **Xen** s'executa en l'anell de màxima prioritat (*anell -1* en Intel/AMD) i en eixe cas sí que podem executar sistemes operatius sense cap modificació.

És important indicar que **Xen** o més concretament el seu *hipervisor* mai serà integrat al nucli de Linux, ja que en realitat es tracta d'un programa independent que no està integrat de cap manera amb el codi del nucli de Linux.

El que si podría incorporar-se al nucli de Linux serien els pedaços que fan que es puga fer servir el Linux com a *convidat privilegiat* (*dom0*) o com a *convidat* a seques (*domU* en terminologia **Xen**).

3.6 Linux-VServer (http://www.linux-vserver.org/)

El **Linux-VServer** és un sistema de virtualització a nivell de sistema operatiu que s'implementa com una sèrie de pedaços sobre el nucli de Linux.

El que fa aquest sistema és incloure suport en el nucli per a crear i mantindre múltiples entorns d'usuari independents (coneguts com a VPS o *Virtual Private Servers*) sense que tinguen cap interferència entre ells.

Per a independitzar els espais d'usuari es defineix el concepte de *context*, que no és més que un *contenidor* (*container*) de processos relacionats amb un únic VPS. Quan el sistema arranca defineix un context per defecte que és el que fan servir tots els processos que pertanyen al sistema *amfitrió*.

A banda dels *contextos*, el **Linux-VServer** també fa servir una crida similar a `chroot` per redefinir el directori arrel dels processos que s'executen dins d'un context determinat i evita que puguen accedir als directoris que hi ha per sota de l'arrel.

Com aquesta tecnologia no té cap dependència relacionada amb la CPU de l'amfitrió, el sistema està suportat per a múltiples famílies de microprocessadors (`x86`, `x86-64`, `PowerPC`, `ARM`, etc.).

El problema d'aquest sistema és que no gestiona adequadament la utilització compartida de recursos virtuals com les targetes de xarxa (o en aquest cas seria més correcte parlar de dispositius virtuals de xarxa), ja que el que fa és fer servir els recursos de l'*amfitrió* sense aïllar-los dels que fa servir la màquina virtual (per exemple si llancem una operació de *bind()* contra un port dins d'una màquina virtual i el port està ocupat en totes les interfícies del sistema per un procés que s'executa en l'*amfitrió* el *bind()* falla, cosa que no pasaria si l'aïllament entre dispositius virtuals de xarxa fora l'adequat.

3.7 OpenVZ (http://www.openvz.org/)

L'**OpenVZ** és un sistema de virtualització similar al **LinuxVServer**, tot i que inclou capacitats i eines d'administració més avançades que les del sistema anterior.

En concret l'**OpenVZ** afegeix virtualització (permet la existència de múltiples entrorns virtuals aïllats dins del mateix nucli), gestió de recursos (proporciona mecanismes per limitar i en ocasions garantir la disponibilitat de recursos com la CPU, la memòria o l'espai de disc per a cada entorn virtual) i capacitat de *checkpointing* (possibilitat de *congelar* un entorn virtual, emmagatzemar el seu estat complet en un fitxer que podem fer servir més tard per a *descongelar* l'entorn virtual en la mateixa o en una altra màquina real deixant-lo en el mateix estat que tenia abans de la congelació).

Addicionalment el **OpenVZ** es distribueix amb un conjunt d'utilitats que simplifiquen molt la creació i manteniment dels entorns virtuals (VE, *Virtual Environments* en la documentació de **OpenVZ**), incloent-hi la possibilitat de treballar en plantilles de entorns virtuals preinstalats (bàsicament les plantilles són arxius `tar.gz` que contenen una imatge del sistema d'arxius arrel d'un VE).

Com en el cas de **Linux-VServer** aquest sistema de virtualització suporta múltiples arquitectures x86, x86-64, PowerPC, etc.

4 Eines relacionades amb els sistemes de virtualització

Tots els sistemes de virtualització que hem comentat abans inclouen eines d'administració per a crear i configurar màquines virtuals i per a executar i aturar eixes màquines.

Evidentment, sobre tot quan parlem de programari lliure, en molts casos les eines no són molt sofisticades, raó per la qual és habitual que es desenvolupen aplicacions que fan de *frontend* dels programes reals, simplificant i fent més còmode el treball amb els sistemes de virtualització.

En l'àmbit del les interfícies gràfiques trobem unes quantes aplicacions per a sistemes com **QEMU** (`qemu-launcher`, `qemuctl`, `qemulator`, `qtemu`), **User-mode Linux** (`guml`) o **Xen** (`xenman`).

A banda de les aplicacions de *frontend* també s'han desenvolupat biblioteques com **libvirt** (http://libvirt.org) que proporcionen una caixa d'eines per a gestionar de manera unificada múltiples sistemes de virtualització (**libvirt** suporta **Xen**, **QEMU**, **KVM** i **OpenVZ** entre altres tecnologies).

Basades en aquesta biblioteca trobem eines de línia d'ordres, aplicacions d'escriptori com **Virtual Machine Manager** (http://virt-manager.org/) i eines web com les proporcionades per **oVirt** (http://ovirt.org/).

A banda de les aplicacions independents també estan començant a aparèixer distribucions com **Proxmox VE** (http://pve.proxmox.com/) que proporcionen el suport necessari per a instal·lar i gestionar clústers de servidors virtuals (en concret aquesta distribució suporta l'ús de **KVM** i **OpenVZ**).

També cal mencionar que al voltant de la virtualització de sistemes també es desenvolupen sistemes per a virtualitzar altres tipus de recursos, com els discs o els dispositius de xarxa a nivell del sistema *amfitrió*, com per exemple el sistema de *ethernet distribuïda virtual* (*Virtual Distributed Ethernet* <http://wiki.virtualsquare.org/index.php/VDE_Basic_Networking>), que es pot fer servir per a crear una xarxa ethernet virtual entre sistemes UML, QEMU y KVM que s'executen en el mateix o en diferents sistemes reals.

Per últim és interessant indicar que el fet de fer servir un sistema com Linux com a *amfitrió* per a la virtualització de sistemes ens permet fer servir moltes altres tecnologies que són o poden ser interessants per a muntar infraestructures virtuals com els sistemes de fitxers amb funcionalitats avançades (*RAID*, *LVM2*) o les eines de gestió de xarxa avançades (*iptables*, *iproute2* ó *brctl*).

5 Exemples d'ús

A continuació comentarem uns quants exemples d'ús de tecnologies de virtualització, explicant les raons per les que es fa servir una tecnologia i no un altra i com s'organitza la gestió i el manteniment dels sistemes de virtualització dins de les infraestructures en les quals s'integren.

5.1 Pràctiques d'administració de sistemes amb user-mode-linux

Hi ha situacions en les que val la pena fer servir tecnologies com el `user-mode-linux` encara que no tinga les mateixes prestacions que altres sistemes de virtualització.

Un bon exemple es dona quan el fet de que el sistema s'execute a nivell d'usuari siga desitjable, com succeïx quan ens plantegem que hem de fer pràctiques d'administració de sistemes GNU/Linux i volem estalviar-nos la reinstal·lació del sistema després de cada sessió.

Gràcies a les capacitats de l'UML i la existència del SLIRP (un programa a nivell d'usuari que ens permet per a donar accés a la xarxa als servidors UML) resulta prou ràpid i econòmic donar a cada alumne una o més màquines virtuals per poder treballar amb elles com administrador sense haver de donar-li cap permís especial en el sistema *amfitrió*.

Un altre avantatge de fer servir el UML és que un cop instal·lat el programari necessari al sistema *amfitrió* no és necessària cap intervenció d'un administrador per a crear, esborrar o modificar les màquines virtuals i resulta prou senzill compartir dades entre la màquina virtual i l'*amfitrió*, ja que UML també suporta l'accés a directoris del sistema *amfitrió* des del sistema *convidat*.

I per a acabar amb aquest exemple comentar que hi ha una característica prou interessant de l'UML quan volem economitzar recursos, en concret ens referim al suport d'arxius COW amb capes, que ens

permet generar un arxiu COW (que per cert pot ser un arxiu amb forats si el sistema d'arxius de l'amfitrió ho suporta) en el qual fem una instal·lació inicial de la màquina virtual que després tots els alumnes poden fer servir com a base, de manera que ells només han de guardar al seu directori un fitxer amb els canvis del sistema d'arxius original, que en sessions de pràctiques solen ser prou xicotets, ja que el normal és únicament editar o afegir uns pocs fitxers.

5.2 Infraestructura actual de virtualització a l'ITI (linux-vserver)

Fa mes d'un any que a l'ITI ens vam plantejar la necesitat de renovar les infraestructures informàtiques d'ús comú per al personal del centre i vam decidir que el més ens interessava era consolidar els servidors per a poder retirar maquinari antic i fer servir la virtualització per a facilitar la escalabilitat del sistemes.

El que vam fer és comprar servidors amb en microprocessadors Intel i AMD de 64 bits sobre els que instalariem la versió estable del sistema operatiu Debian GNU/Linux (ens referim a la versió 4.0, amb nom en codi *etch*).

Aquesta versió inclou de sèrie paquets binaris de la versió 2.6.18 de Linux amb suport per a **Xen** i **Linux-VServer**.

Inicialment només voliem instal·lar servidors virtuals Linux, així que vam triar el **Linux-VServer**, bàsicament perqué és una tecnologia més lleugera i senzilla que **Xen** i de tota manera en un futur podem fer servir les dues tecnologies (hi ha disponibles versions del nucli que donen suport de les dues tecnologies anteriors al mateix temps).

Des de que vam començar a fer servir el **Linux-VServer** vam començar a automatitizar i definir procediments per a instal·lar i configurar els sistemes *amfitrions* i *convidats* per al nostre entorn.

El model que fem servir asumeix que el sistema *amfitrió* ha de donar la mateixa imatge als servidors virtuals que s'executen damunt seu, per la qual cosa hem definit una sèrie de serveis i configuracions que sempre han de ser iguals:

- En primer lloc fem que tots els vservers tinguen les seues interfícies de xarxa lligades al dispositiu virtual de xarxa `dummy0` que està configurat en el sistema amfitrió per a tindre l'adreça IP 10.0.0.1 (eixa adreça la fem servir com a IP local del *amfitrio* des de tots els servidors virtuals). Les adreces dels servidors virtuals estan sempre en la xarxa privada `10.0.0.0/8`.

- Per a donar accés a l'exterior als vservers fem servir un script d'*iptables* que fa NAT per a donar accés a l'exterior als servidors virtuals.

  Per a donar accés als serveis públics que s'executen en els sevidors virtuals fem servir regles de *port forwarding* que també es configuren en el mateix script d*iptables*.

- L'*amfitrió* sempre executa un servidor de SMTP local que escolta en la IP 10.0.0.1 i s'encarrega de modificar les adreces dels missatges que ixen dels servidors virtuals (fem servir la reescriptura *canonical* de *postfix*, que canvia adreces d'entrada i eixida) i enviar-los als *smarthosts* que corresponga (dins de l'institut s'envien al nostre servidor de correu principal).

- Opcionalment l'*amfitrió* pot executar un servidor proxy de DNS (normalment fem servir el `pdnsd`) que també funciona amb l'adreça 10.0.0.1 i ens independitza de la xarxa en la qual s'execute el servidor virtual, ja que no li cal ni saber quina és l'adreça del servidor de noms.

- Com tots els servidors els comprem amb dos discs, sempre els montem en RAID-1 (*mirroring*) per a garantir que tenim redundància davant de problemes físics en un dels discs.

- Damunt dels RAID montem sistemes de fitxers amb LVM2, deixant sempre espai lliure per a poder fer servir *snapshots*, ja que ens resulten molt útils per a fer còpies de seguretat dels sistemes de fitxers sense haver de parar els serveis. El que fem és garantir que els fitxers al disc són bons, prenem una foto fixa del sistema de fitxers (un *snapshot* amb LVM2) i continuem treballant normalment. El procés de còpia de seguretat es fa contra la imatge fixa del sistema d'arxius (en el nostre cas fem servir el `rdiff-backup`), que sabem que conté dades bones i quan el procés de còpia acaba eliminem la foto fixa i els sistema continua treballant normalment sense cap interrupció.

- Per a monitoritzar el bon funcionament dels serveis locals de l'*amfitrió* fem servir el programa `monit`, que configurem per a monitoritzar diferents paràmetres del sistema i la disponibilitat dels serveis que considerem necessaris.

Quant als sistemes *convidats* el que hem fet és aplicar un model semblant al d'**OpenVZ**, és a dir, hem preparat un script per replicar una plantilla d'un servidor virtual i modificar els quatre paràmetres bàsics per a que funcione de manera independent.

Dins del sistema plantilla, que també és una **Debian Etch** instal·lem sempre uns quants serveis ja configurats, incloent-hi un servidor de `ssh` per a poder connectar als servidors virtuals des de la xarxa, un servidor `postfix` preparat per a interactuar amb el servidor del sistema *amfitrió* i un `monit` per a supervisar el funcionament dels serveis anteriors.

Per a migrar o canviar un servidor virtual de màquina fem servir un procés manual amb el qual copiem les dades del servidor virtual, el parem, tornem a sincronitzar-lo i l'aixequem en una màquina nova. Per a que els serveis estiguen accessibles des de l'exterior també és necessari a vegades canviar la configuració dels tallafocs dels *amfitrions* tot i que actualment en realitat no sempre és necessari, ja que fem servir la segona targeta de xarxa dels servidors físics per a intercomunicar els servidors virtuals.

5.3 Infraestructura futura de virtualització a l'ITI (OpenVZ i KVM)

Tot i que la infraestructura amb `linux-vserver` funciona prou bé, tenim uns quants problemes amb el sistema que ens han fet decidir-nos a migrar a un sistema de virtualització diferent aprofitant tot el que puguem del sistema actual:

- Sembla que el projecte està una mica parat (no hi ha pedaç oficial des de la versió 2.6.22 del nucli de Linux), la qual cosa implica que si la situació no canvia podem perdre el suport de seguretat del nucli dels sistemes instal·lats i és fàcil que arribe un moment en què no siga possible instal·lar el sistema base en màquines noves per falta de controladors de dispositiu adequats.

- No es pot fer una migració *en viu* dels servidors virtuals. Tot i que no és un problema greu, és una funcionalitat que resulta molt útil quan volem ajustar la càrrega dels servidors o per a posar en marxa un sistema de alta disponibilitat per als servidors o serveis.

- Necessitem podem executar sistemes operatius distints al GNU/Linux, però amb **Linux-VServer** no és possible.

El problema més greu a mig termini és el primer, així que el primer que vam avaluar són les alternatives dintre del món dels sistemes de virtualització a nivell de sistema operatiu.

La primera opció que vam revisar va ser l'**OpenVZ**, ja que en realitat no va ser la tecnologia emprada inicialment simplement perquè la versió 4.0 de Debian inclou paquets binaris del nucli de Linux amb suport de **Linux-VServer**, mentre que per a generar nuclis amb suport de **OpenVZ** és necessari aplicar el pedaç i generar manualment els paquets. En el seu moment vam considerar que les diferencies de funcionalitat no justificaven eixa petita feina addicional i ens vam quedar amb **Linux-VServer**, pensant també que com que es tractava d'un sistema més simple seria més fàcil que entrara dins del nucli estàndard de Linux, la qual cosa amb el temps no ha passat.

A banda de per les seues millores tècniques (suport de migració en viu, eines de gestió més completes i amb mes funcionalitats, etc.), també vam considerar que l'**OpenVZ** era una bona opció per al futur després de revisar el *blog* del projecte (http://community.livejournal.com/openvz/) on es parla de la quantitat de pedaços relacionats amb el sistema que van entrant dins del nucli estàndard de Linux i és veu clarament que la tendència actual sembla prou bona.

Per a l'últim problema ja havíem decidit avaluar l'ús de nuclis amb suport de **Xen** (de fet en la versió 4.0 de Debian hi ha nuclis amb suport de `Xen` i `linux-vserver`), però abans de començar amb les proves reals vam abandonar l'idea de fer servir Xen i vam optar per KVM, bàsicament per que es tracta d'un sistema molt més simple d'instal·lar i configurar i per que es tracta de un sistema que ha entrat dins de la versió estàndard de Linux, la qual cosa ens dóna molta més confiança que anar depenent sempre dels pedaços i les adaptacions de les distribucions.

En definitiva, els sistemes que posarem en marxa en el futur faran servir les mateixes tecnologies a nivell d'*amfitrió* (com per exemple el LVM2 per a muntar els sistemes de fitxers) i incorporaran suport per a **OpenVZ** i **KVM**.

Per als dos sistemes de virtualització farem servir *bridging* per a les interfícies de xarxa virtuals, tot i que en els nostres sistemes els *ponts* es faran només sobre les interfícies internes dels servidors, de manera que continuarem fent servir el *port forwarding* per a publicar els serveis interns en la xarxa pública.

En quant a la migració dels sistemes actuals els farem fent servir dos tècniques bàsiques:

1. Instal·lacions noves i migració, actualització o canvi dels serveis. La idea és, en moltes ocasions, aprofitar el canvi de màquina virtual per a fer una migració a una instal·lació nova (per exemple per a migrar un sistema que funciona en una distribució de 32 bits a la darrera versió de la distribució amb arquitectura amd64).

2. Adaptació de l'*vserver* per a que funcione amb **OpenVZ**, que en general i per sort només implica reajustar la configuració de la xarxa i poc més.

Quant a la gestió dels sistemes de moment estem fent proves amb el sistema que porta incorporat la distribució **Proxmox VE**, tot i que és molt probable que acabem fent servir directament l'`ssh` i el"vnc".