ISTITUTO NAZIONALE DI FISICA NUCLEARE

CNAF

INFN/CCR-07/8
November 26, 2007

CCR/15/07/P

XEN E I BENEFICI DELLA VIRTUALIZZAZIONE HVM

Riccardo Veraldi[1]

[1)] *INFN-CNAF viale Berti Pichat 6/, I-40127 Bologna, Italy*

Abstract

Xen è un Virtual Machine Monitor che consente l'implementazione di macchine virtuali. Grazie alla tecnologia presente nelle CPU di ultima generazione è possibile girare un sistema operativo nativo come macchina virtuale su Xen. I benefici sono molteplici e vanno dalla possibilità di girare sistemi operativi misti a 32 e 64 bit su un unico hardware, all'implementazione dell'alta affidabilità, al test di nuovi sistemi operativi. Il seguente documento descrive Xen nell'ottica di hypervisor di macchine virtuali utilizzando come base (dom0) la distribuzione Scientific Linux 5.0 x86_64.

*Published by **SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

1 XEN

Xen è un prodotto opensource per la virtualizzazione dei sistemi operativi. E' un hypervisor che consente a più sistemi operativi, chiamati *guest*, di essere eseguiti contemporaneamente su una singola macchina fisica. Xen può funzionare in modalità *paravirtualizzata* o *completamente virtualizzata*.

Nella paravirtualizzazione il sistema operativo *guest* deve contenere delle opportune patch che implementano le hypercall di Xen. Diversi tipi di CPU sono supportati in questo caso, in generale tutte quelle con architettura x86. Sono supportati anche multiprocessori e tecnologia HyperThreading.

Nel caso della virtualizzazione completa (hvm), la CPU della macchina possiede al suo interno delle estensioni apposite (processori Intel VT e AMD Pacifica technology) che consentono al sistema operativo guest di essere eseguito in modo nativo senza ulteriori patch al kernel. Un sistema Xen ha diversi livelli. Il livello più basso (quello che ha i massimi privilegi) è Xen stesso ed è chiamato *domain0* o *dom0*. I sistemi operativi (*guest*) che girano sopra Xen stesso, sono le macchine virtuali e vengono chiamati *domainN*, dove *N* identifica la macchina virtuale in esecuzione sul dom0.

Il domain0 ha le proprietà di supervisore nei confronti dei domini non privilegiati, gestisce i dispositivi virtuali che sono contesi dai vari sistemi guest, e gestisce la sospensione, il resume e la migrazione delle diverse macchine virtuali che possono essere in esecuzione, oltre che gestirne la creazione e la distruzione. Per ulteriori dettagli sulla tecnologia Xen si rimanda al sito ufficiale http://xen.xensource.com

2 XEN 3.1.0 E SCIENTIFIC LINUX 5.0

La versione opensource di Xen 3.1.0 contiene alcune novità, tra le quali la possibilità di potere eseguire istanze di macchine virtuali miste a 32 e 64 bit su un dom0 a 64 bit e la possibilità di effettuare una live migration per macchine virtualizzate hvm. In questo documento quando parliamo di Xen hypervisor e di macchine virtuali facciamo riferimento comunque alla virtualizzazione con supporto hardware da parte delle CPU. La virtualizzazione completa consente ai sitemi operativi guest di girare in modo nativo sul dom0.

Scientific Linux 5.0 include Xen 3.0.3 nella distribuzione.

La distribuzione binaria pre-compilata di Xen 3.1.0 scaricabile da xensource non funziona correttamente su Scientific Linux 5.0 provocando un kernel panic.

Scaricando i sorgenti di Xen da xensource, ho realizzato un pacchetto di distribuzione pronto per essere installato su SL 5.0 x86_64.

2.1 Prerequisiti

Per trarre i massimi vantaggi da Xen 3.1.0 è opportuno avere i seguenti requisiti da parte dell'hardware:

- Processori che supportano la Virtualizzazzione in hardware
  - Intel VT (IVT)
  - AMD-V
- Una discreta quantità di RAM che deve poi essere suddivisa tra le varie macchine virtuali in esecuzione

- Spazio disco da dedicare ai sistemi operativi delle macchine virtuali. Ogni macchina virtuale può risiedere all'interno di un file che funge da contenitore per tutto il sistema, oppure usare una partizione fisica dedicata del disco

Per costruire il pacchetto Xen 3.1.0 installabile su SL 5.0 e per i successivi test effettuati ho utilizzato una macchina con 2 CPU Intel(R) Xeon(R) 5110 @ 1.60GHz e 4GB di RAM.
Inoltre uno storage su disco configurato in RAID5 Areca per potere gestire uno spazio adeguato su cui risiedono le macchine virtuali. Per questo tipo di storage i driver non sono presenti all'interno del kernel 2.6.18 di Scientific Linux. Il kernel module arcmsr è stato compilato e caricato separatamente.

2.2 Installazione di Xen 3.1.0

Prima di tutto è necessario installare il sistema di base, cioè la distribuzione SL 5.0 x86_64 con kernel NON Xen. E' consigliata un'installazione minimale. Il sistema di base, una volta installato Xen, dovrà soltanto gestire le macchine virtuali, e non dovrà fare altro. Successivamente occorre scaricare i pacchetti contenenti Xen 3.1.0 e installarli.

Nel caso in cui si voglia effettuare l'installazione di Xen su una SL 5.0 che supporta già la virtualizzazione con il pacchetto nativo Xen 3.0.3, occorre disinstallare tutte le RPM relative a questa versione di Xen, compresi i tool di xen e le librerie.

L'installazione può essere effettuata utilizzando un pacchetto pre-compilato che deve essere scompattato e successivamente installato con uno script, o alternativamente installando le opportune RPM che sono state create in fase di build della release di Xen 3.1.0 dai sorgenti. Di seguito sono riportate entrambe le metodologie di installazione. Di solito il pacchetto RPM è il metodo preferito per le installazioni su sistemi RHEL.

2.2.1 Installazione del pacchetto tgz

Si scarica il pacchetto configurato e ricompilato su SL 5.0 x86_64 da :

https://calcolo.infn.it/xen/xen-3.1.0-CNAF-bin.tgz

Verrà creata una directory *xen-3.1.0-CNAF-bin*. Seguire le istruzione in Install.txt.
In particolare:

- Eseguire lo script di installazione *install.sh* che si trova all'interno di *xen-3.1.0-CNAF-bin*
- aggiungere una entry in */boot/grub/menu.lst* per il kernel 2.6.18-xen in modo che al boot parta il kernel xen:

```
title Scientific Linux SL (2.6.18-xen-3.1.0)
  root (hd0,0)
  kernel /xen-3.1.0.gz
  module /vmlinuz-2.6.18-xen ro root=/dev/VolGroup00/LogVol01
  module /initrd-2.6.18-xen-3.1.0.img
```

E' importante specificare il device corretto nella entry

```
root= ...
```

in questo caso è una partizione LVM.

- abilitare i servizi *xend* e *xendomains* al boot
- riavviare e partire con il nuovo kernel 2.6.18-xen-3.1.0
- settare il kernel Xen come default in *grub.conf*

2.2.2 Installazione delle rpm

Si scaricano i pacchetti RPM di seguito:

https://calcolo.infn.it/xen/kernel-xen-2.6.18-xen_3.1.0_CNAF.SL50.x86_64.rpm
https://calcolo.infn.it/xen/xen-3.1.0-CNAF.SL50.x86_64.rpm
https://calcolo.infn.it/xen/xen-devel-3.1.0-CNAF.SL50.x86_64.rpm

Queste RPM sono state costruite compilandole su Scientific Linux 5.0 x86_64.
Prima di installare bisogna essere sicuri che non ci siano altri pacchetti Xen relativi alla distribuzione di default 3.0.3 presente su Scientific Linux. E' necessario prima di tutto rimuovere le RPM di Xen di default (versione 3.0.3) nel caso fossero presenti sul sistema.
A questo punto è possibile installare le nuove RPM di Xen 3.1.0

```
rpm –ivh kernel-xen-2.6.18-xen_3.1.0_CNAF.SL50.x86_64.rpm
rpm –ivh xen-3.1.0-CNAF.SL50.x86_64.rpm
rpm –ivh xen-devel-3.1.0-CNAF.SL50.x86_64.rpm
```

Verranno installate le librerie di Xen, gli Xen tools e il kernel Xen.

Per partire con il nuovo kernel Xen occorre aggiungere una entry in /boot/grub/menu.lst

```
title Scientific Linux SL (2.6.18-xen-3.1.0)
        root (hd0,0)
        kernel /xen-3.1.0.gz
        module /vmlinuz-2.6.18-xen ro root=/dev/VolGroup00/LogVol01
        module /initrd-2.6.18-xen.img
```

E' importante specificare il device corretto nella entry

```
root= ...
```

in questo caso è una partizione LVM */dev/VolGroup00/LogVol01*.
Bisogna poi abilitare i servizi *xend* e *xendomains* al boot, riavviare e partire con il nuovo kernel 2.6.18-xen-3.1.0 dal menù di grub. Se tutto va bene settare il kernel Xen come default al boot .

2.3 Utilizzo di Xen 3.1.0

Un grande vantaggio nell'utilizzo di Xen 3.1 è la possibilità di consentire a più macchine virtuali miste a 32 e 64 bit di potere coesistere, cosa che non era possibile con Xen 3.0.

In figura 1 possiamo vedere due macchine virtuali Scientific Linux 5, una a 32 bit

migrata da un ambiente paravirtualizzato a Xen 3.1.0 con virtualizzazione completa, e un'altra a 64 bit. Entrambe sono in esecuzione contemporaneamente.

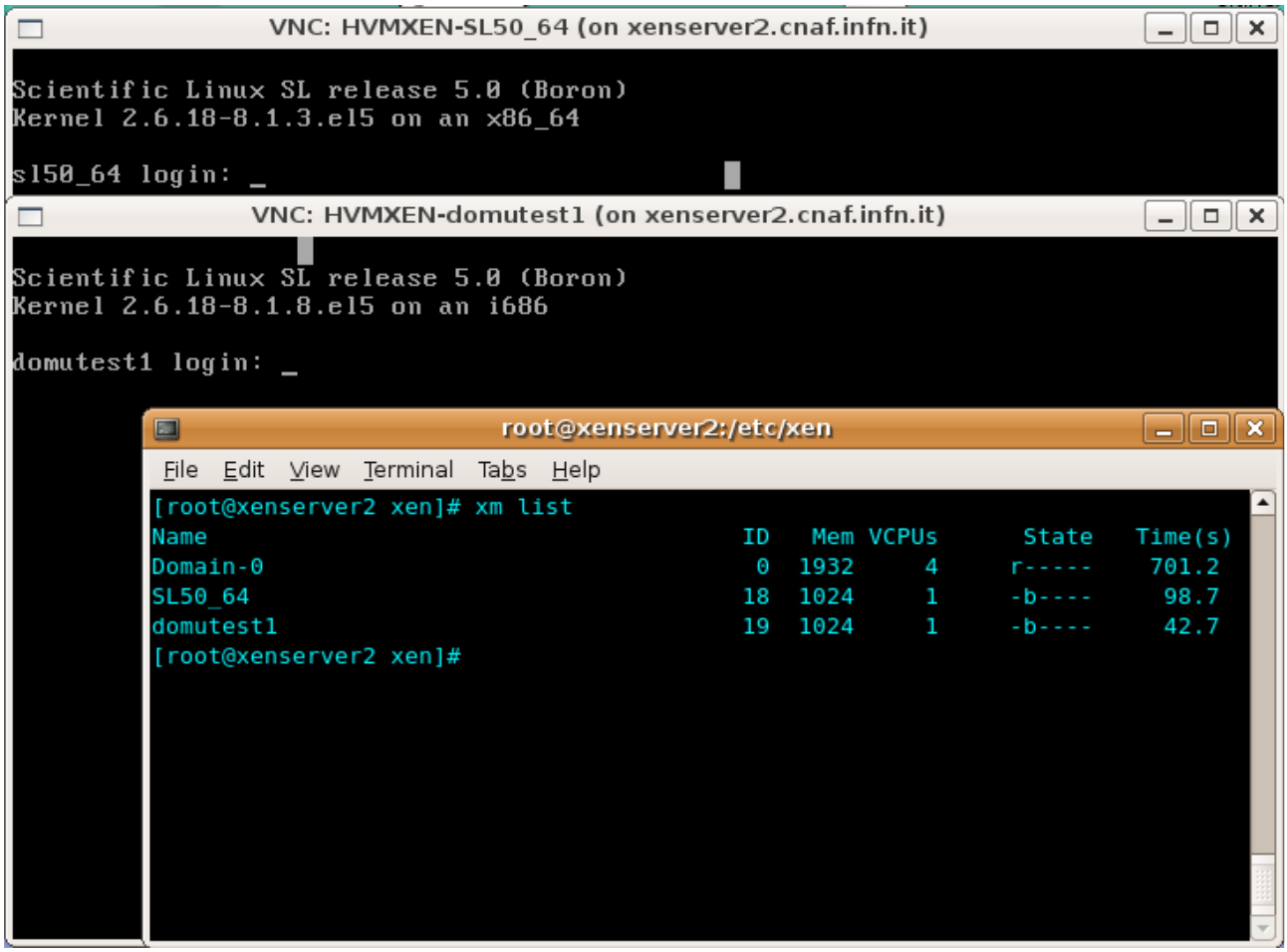


Figura 1

Analizziamo ora alcune situazioni comuni che si possono presentare nell'utilizzo di Xen come hypervisor per l'implementazione di macchine virtuali.

2.3.1 Installazione completa di una macchina virtuale a 64 bit

Trattiamo ora dell'installazione di una macchina virtuale su un dom0 Xen 3.1.0.

Stiamo parlando di un hardware di base con processori moderni che supportano la virtualizzazione in hardware come citato nei **prerequisiti**.

Innanzitutto è necessario creare un profilo per la nuova macchina virtuale da installare. Il profilo è un file che va creato in */etc/xen/*. Lo chiameremo **SL50_64** per indicare che il sistema guest è una Scientific Linux 5.0 a 64 bit:

```
name = "SL50_64"
builder = "hvm"
memory = "1024"
disk=['file:/xen/domU/boot.iso,hdc:cdrom,r','file:/xen/domU/SL50_64,hda,w', ]
boot = "d"
vif = [ 'mac=00:16:3e:3f:a9:06, bridge=xenbr0', ]
```

```
vfb = ["type=vnc,vncunused=1"]
uuid = "ffc68565-354a-32da-7302-72a1bd5ff1bf"
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
on_shutdown = 'destroy'
```

analizziamo la sintassi del file di configurazione:

- **name**: nome del profilo/macchina virtuale
- **builder**: si specifica, in questo caso la virtualizzazione completa
- **memory**: la macchina virtuale avrà 1Gb di RAM
- **disk**: si specificano 2 device di storage; un file boot ISO come cdrom e un file che funge da contenitore per il sistema operativo che stiamo installando. Il sistema una volta installato risiederà nel file
- **boot**: sintassi quemu; specifica quale è il dispositivo di boot, **d** sta per cdrom
- **vif**: specifica il mac address e il bridge utilizzato da Xen. in questo caso il bridge utilizza eth0 del dom0. Ogni macchina virtuale deve avere un proprio mac address assegnato arbitrariamente. Normalmente ogni guest ha un mac address che deve essere del tipo **00:16:3e:x:y:z** con x,y e z a piacere, basta che sia unico per ogni macchina sulla rete locale.
- **uuid**: è un identificativo della macchina virtuale, si può anche omettere, deve essere unico per ogni macchina. Non è un parametro necessario
- **device_model**: specifica che i device sono emulati con qemu
- **kernel**: necessario per la virtualizzazione hardware; un certo numero di CPU verrà assegnato esclusivamente a questa macchina virtuale
- **vcpus**: numero di CPU assegnate alla macchina virtuale
- **on_reboot** / **on_crash** / **on_shutdown**: come l'hypervisor deve trattare la macchina virtuale in queste tre situazioni

Una volta configurato il profilo di base, bisogna creare il file che conterrà l'immagine della macchina virtuale dopo l'installazione. Ammettiamo che le immagini delle nostre macchine virtuali risiedano in */xen/domU*

```
cd /xen/domU
dd if=/dev/zero of=SL50_64 bs=1k seek=20479k count=1
```

Così viene quindi creato un file *sparse* di 20GB. In realtà lo spazio occupato da questo file non è 20GB ma è uguale allo spazio effettivamente utilizzato dai file contenuti nell'immagine ad installazione ultimata. In questo modo si possono creare file d'immagine che virtualmente hanno una certa dimensione, consentendo quindi al sistema di poter "crescere" dinamicamente durante la sua vita, ma in realtà la dimensione effettiva è data dallo spazio realmente utilizzato e non allocato. Per esempio, se si crea una file *sparse* da 50GB per una machina virtuale e il sistema in esame occupa soltanto 3GB, lo spazio effettivamente occupato sul dom0 da questa macchina virtuale sarà di 3GB e non di 50GB.
Un file *sparse* contiene dei "buchi", sequenze di byte nulli che non occupano nessun blocco fisico sul disco.

Il passo successivo è scaricare un'immagine di boot della Scientific Linux 5.0 x86_64
Basta scaricare la ISO image minimale che contiene i tool d'installazione di SL da ftp://ftp.scientificlinux.org/linux/scientific/50/x86_64/images/boot.iso.

Siamo pronti per lanciare la macchina virtuale ed iniziare con l'installazione:

```
xm create /etc/xen/SL50_64
```

Utilizzando i tool di gestione di Xen vediamo che la macchina virtuale parte ed è attiva:

```
xm list

Name                                        ID   Mem VCPUs      State   Time(s)
  Domain-0                                   0  4096     1     r-----    227.0
  SL50_64                                   12  1024     1     r-----      1.7
```

a questo punto collegandoci con vnc possiamo direttamente accedere alla console della macchina virtuale che abbiamo appena creato (figura 2):

```
vncviewer localhost
```

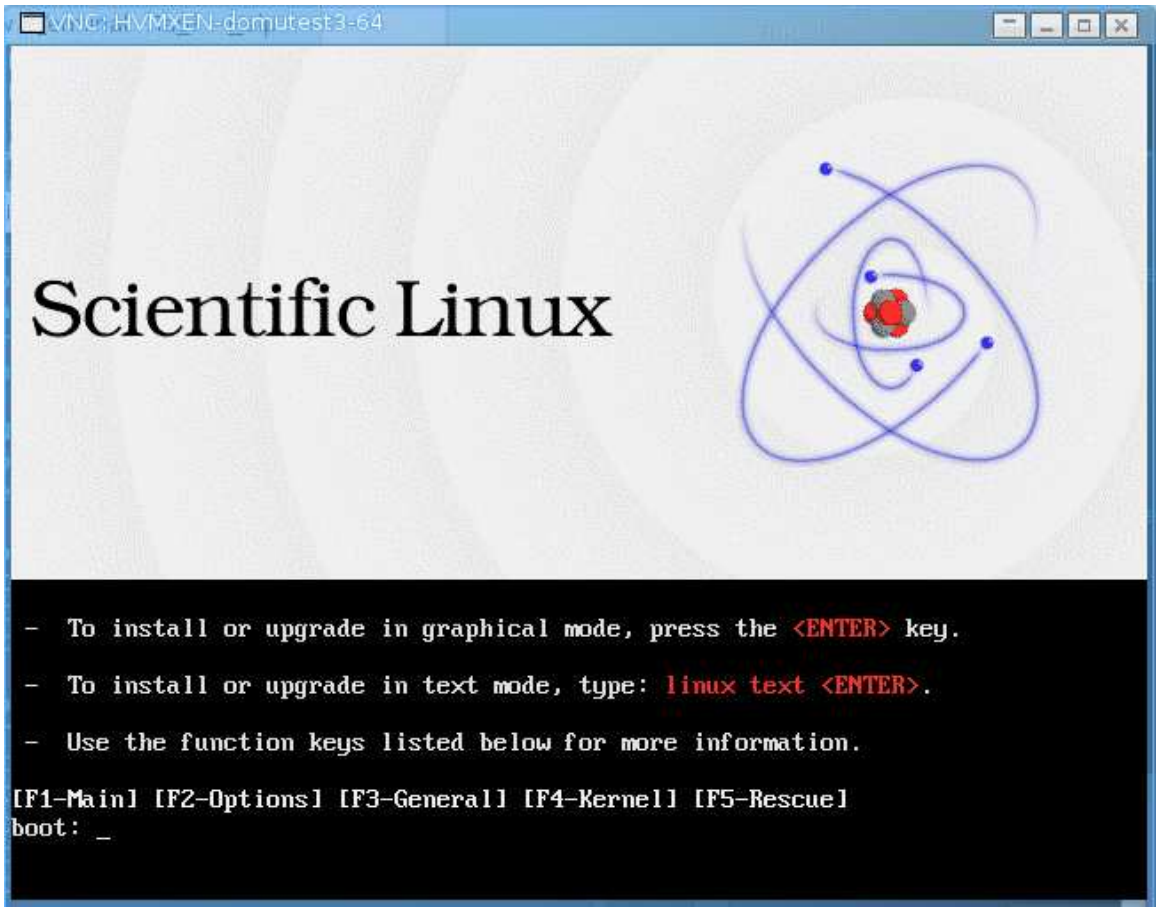


Figura 2

da qui si può iniziare la procedura d'installazione di un guest a 64 bit come si fa per le macchine reali. L'installazione può avvenire via rete utilizzando un qualsiasi repository mirror ufficiale.
Ad installazione ultimata occorre fare lo shutdown della macchina virtuale dalla console ed eventualmente fermare sul dom0 la macchina se tenta di fare un reboot con il comando

```
xm shutdown ID
```

Dove ID è il numero identificatore che Xen assegna alla macchina guest in esecuzione (vedi `xm list`).
A questo punto per fare partire la macchina virtuale appena installata occorre effettuare una modifica al file di configurazione *SL50_64* in questo modo:

```
name = "SL50_64"
builder = "hvm"
memory = "1024"
disk = [ 'file:/xen/domU/SL50_64,hda,w', ]
vif = [ 'mac=00:16:3e:3f:a9:06, bridge=xenbr0', ]
vfb = ["type=vnc,vncunused=1"]
uuid = "ffc68565-354a-32da-7302-72a1bd5ff1bf"
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
on_shutdown = 'destroy'
```

cioè rimuoviamo la entry relativa al dispositivo virtuale CDROM e forziamo il boot dal file di immagine del sistema operativo */xen/domU/SL50_64*.
Per riavviare la macchina virtuale:

```
xm create /etc/xen/SL50_64
```

Al primo boot è opportuno disabilitare SELinux e rimuovere i servizi di cui non si ha bisogno assieme gli indirizzi IP assegnati in fase di installazione e l'hostname.
A questo punto effettuato lo shutdown della macchina virtuale avremo un file di immagine */xen/domU/SL50_64* contenente la distribuzione Scientific Linux 5.0 che potrà essere spostato su altri hypervisor Xen e personalizzato per servizi specifici. Questo file è diventato quindi la base per distribuire una macchina virtuale perfettamente funzionante senza dovere re-installarla ogni volta. Il file di configurazione */etc/xen/SL50_64* è il profilo necessario per potere lanciare la macchina virtuale in modalità completamente virtualizzata su un qualsiasi hypervisor Xen.

2.3.2 Porting di una macchina virtuale paravirtualizzata su Xen hvm

Analizziamo ora come potere trasferire una macchina virtuale che gira su Xen in modalità paravirtualizzata su un dom0 Xen che supporta la virtualizzazione in hardware. Questo è uno scenario piuttosto comune via via che si migrano i servizi implementati su macchine virtuali da hardware obsoleto a nuovo hardware che ha CPU di ultima generazione

con il supporto per le estensioni VT (virtualizzazione in hardware). Il file di immagine della macchina virtuale di cui vogliamo fare il porting va modificato assieme al file di profilo. Vanno pertanto inserite opportune modifiche al sistema che gira come macchina virtuale e nel profilo Xen che lo descrive, in particolare:

- installare un kernel senza supporto Xen sulla macchina virtuale che è in esecuzione sul vecchio hardware
- modificare il file di profilo di Xen della macchina virtuale
- configurare Xen dom0 in modo analogo al vecchio hypervisor relativamente ai bridge delle interfacce di rete (file di configurazione */etc/xen/xend-config.sxp*) ed eventualmente copiare eventuali script personalizzati in */etc/xen/scripts*
- spostare il file contenente l'immagine della macchina virtuale e il suo profilo sulla nuova macchina con Xen 3.1.0
- fare lo startup della macchina virtuale e selezionare il kernel NON xen al boot nel menù di GRUB

ESEMPIO:

Spostiamo la macchina virtuale *lists.infn.it* da un hardware obsoleto ad un nuovo hardware VT capable:

- innanzi tutto va installato un kernel standard, senza le xen hypercall sulla machina virtuale che è in esecuzione sul vecchio hardware:

  ```
  yum install kernel-2.6.18-8.1.8.el5
  ```

- si effettua lo shutdown della macchina virtuale
- sul dom0 si comprime il file del sistema operativo che risiede in */xen/domU/lists.img* tenendo conto che il file di immagine è uno *sparse* file. Per fare questo:

  ```
  tar -cvzSf lists.img.tar.gz /xen/domU/lists.img
  ```

- si copia il file /xen/domU/lists.img e il profilo /etc/xen/lists sulla nuova macchina con Xen 3.1.0
- si modifica opportunamente /etc/xen/xend-config.sxp in modo da replicare la configurazione di Xen che era presente nel precedente Xen 3.0.3 sul vecchio hardware (bridging delle interfacce di rete, script personalizzati ecc.). Nel caso specifico non vi sono modifiche sostanziali da apportare
- si fa ripartire Xen

  ```
  service xend restart
  ```

- si modifica radicalmente il file del profilo /etc/xen/lists. Di seguito viene riportato il file come era su Xen 3.0.3 e il file modificato per potere essere utilizzato su Xen 3.1.0 con la virtualizzazione hardware. In rosso sono evidenziate le parti del file di configurazione che differiscono:

```
name = "lists"
memory = "1024"
disk = [ 'tap:aio:/xen/domU/lists.img,xvda,w', ]
vif = [ 'mac=00:16:3e:3f:a9:f5, bridge=xenbr1', ]
vfb = ["type=vnc,vncunused=1"]
uuid = "ffc68565-354a-32da-7302-72a1bd5ff1af"
bootloader="/usr/bin/pygrub"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
```

profilo per la macchina virtuale lists su Xen 3.0.3 hypervisor con paravirtualizzazione

```
name = "lists"
builder = "hvm"
memory = "1024"
disk = [ 'file:/xen/domU/list.img,hda,w', ]
vif = [ 'mac=00:16:3e:3f:a9:f5, bridge=xenbr1', ]
vfb = ["type=vnc,vncunused=1"]
uuid = "ffc68565-354a-32da-7302-72a1bd5ff1af"
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
```

File profilo per la macchina virtuale lists su Xen 3.1.0 hypervisor fully virtualized

Come è possibile notare i due file di configurazione sono sostanzialmente diversi. In particolare cambia il nome del dispositivo fisico che emula il disco, da *xvda* a *hda*, e viene poi utilizzato *hvmloader* che è una sorta di mini-kernel che sta a un livello inferiore del kernel della macchina virtuale in esecuzione

- Bisogna scompattare l'immagine del sistema copiata sul nuovo dom0 con Xen 3.1.0. Ammettiamo che il file sia stato copiato in */xen/domU/*

```
cd
tar -xvzSf lists.tar.gz
```

A questo punto la macchina virtuale lists può partire in modalità fully virtualized, basta lanciare il comando

```
xm create /etc/xen/lists
```

e selezionare nel menù di grub il kernel NON Xen. Bisogna poi ricordarsi di impostare il kernel NON Xen come default una volta che il sistema è partito andando a modificare

```
/boot/grub/menu.lst
```

Come sempre per collegarsi alla console della macchina virtuale basta utilizzare vnc:

```
vncviewer localhost
```

3 PERSONALIZZAZIONE DI XEN DOM0

E' possibile configurare Xen dom0 in modo personalizzato per supportare esigenze specifiche. Si possono effettuare diverse customizzazioni della configurazione di Xen relativamente all'utilizzo delle interfacce di rete e degli altri dispositivi come i dischi. Analizziamo alcune situazioni classiche.

3.1 Modifiche relative alle interfacce di rete

Normalmente quando Xen dom0 è installato ha una configurazione di default che prevede il bridging dell'interfaccia di rete delle macchine virtuali con l'interfaccia eth0 del dom0. Prendiamo in esame alcune configurazioni diverse dal default.

3.1.1 Utilizzo di entrambe le interfacce di rete del dom0

Normalmente si hanno 2 interfacce di rete fisiche sul dom0: eth0 ed eth1. Configurando Xen opportunamente si può fare in modo che la macchina virtuale in esecuzione sul dom0 utilizzi la scheda eth1 del dom0, e per la macchina guest questa scheda sarà la propria eth0. In pratica si modifica la configurazione di default di Xen in modo che venga creato un bridge sulla eth1 del dom0 piuttosto che sulla default eth0. In questo modo la eth0 può essere utilizzata per il management del dom0 e la eth1 (su una sottorete IP differente) sarà l'interfaccia utilizzata da tutte le macchine virtuali. Ma la situazione più flessibile è di configurare due bridge separati, uno per la eth0 e l'altro per la eth1 in modo da potere avere a disposizione entrambe le interfacce di rete del dom0 come interfacce che possono essere virtualmente utilizzate dai sistemi guest.

Per fare questo occorre creare uno script personalizzato in */etc/xen/scripts* che chiamiamo *my-network-script* con le seguenti righe:

```
#!/bin/sh
dir=$(dirname "$0")
"$dir/network-bridge" "$@" vifnum=0
"$dir/network-bridge" "$@" vifnum=1
```

Inoltre occorre modificare il file di configurazione */etc/xen xend-config.sxp* così da settare il bridge in modo opportuno. In particolare va sostituta la riga

```
(network-script network-bridge)
```

con

```
(network-script my-network-script)
```

A questo punto quando Xen parte sul dom0 vengono creati due bridge, uno sulla eth0 e l'altro sulla eth1. Le macchine virtuali in esecuzione sul dom0 possono così utilizzare il bridge legato alla eth1 anzichè il default sulla eth0. Di seguito è riportato come esempio un dom0 che esegue tre macchine virtuali:

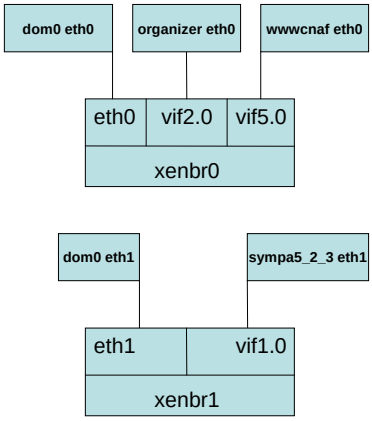
```
Name                                        ID Mem(MiB) VCPUs State   Time(s)
     Domain-0                                0     1427     1 r-----   6185.8
     organizer                               5      511     1 -b----    573.3
     sympa5_3_2                              1     1023     1 -b----   3891.3
     wwwcnaf                                 2     1023     1 -b----  23681.2
```

sympa5_3_2 ha come scheda di rete un'interfaccia virtuale *vif1* collegata alla eth1 del dom0 tramite un opportuno bridge *xenbr1*.
organizer e *wwwcnaf* hanno come scheda di rete un'interfaccia virtuale, rispettivamente *vif5.0* e *vif2.0*, collegate alla eth0 tramite il bridge *xenbr0*:

```
brctl show

bridge name     bridge id               STP enabled     interfaces
xenbr0          8000.feffffffffff       no              vif5.0
                                                        vif2.0
                                                        peth0
                                                        vif0.0
xenbr1          8000.feffffffffff       no              vif1.0
                                                        peth1
                                                        vif0.1
```

Di seguito è riportata una rappresentazione schematica di come sono organizzate le interfacce di rete delle macchine guest relativamente alle interfacce del dom0.



Ovviamente per fare questo bisogna configurare anche il profilo della macchina virtuale in modo opportuno. Va specificato a quale bridge appartiene l'interfaccia di rete della macchina virtuale che si vuole creare. Ad esempio se si vuole utilizzare la seconda interfaccia del dom0 (eth1) al posto di eth0:

```
vif = [ 'mac=00:16:3e:5f:77:fd, bridge=xenbr1', ]
```

oppure se si vuole rendere visibili entrambe le interfacce eth0 ed eth1 alla macchina virtuale si modifica il profilo in questo modo:

```
vif = [ 'mac=00:16:3e:5f:77:fd, bridge=xenbr0', 'mac=00:16:3e:5f:77:fc, \
bridge=xenbr1', ]
```

Di seguito è riportata una rappresentazione schematica di come sono

organizzate le interfacce di rete delle macchine guest relativamente alle interfacce del dom0

3.2 Utilizzo di una partizione fisica

E' possibile configurare Xen in modo che una macchina virtuale utilizzi una partizione fisica del disco invece che un'immagine su file. Questo consente di aumentare enormemente le prestazioni di I/O su disco. Basta modificare nel profilo della macchina virtuale la entry *disk*. Ad esempio se vogliamo dedicare la partizione sdb2 sul dom0 per essere utilizzata in modo diretto da una macchina virtuale dovremo modificare una entry nel file del profilo:

```
disk = ['file:/xen/domU/lists.img,hda,w','phy:/dev/sdb2,hdb,w']
```

In questo caso la macchina virtuale lists utilizzerà il file */xen/domU/lists.img* ma avrà anche accesso diretto alla partizione */dev/sdb2* del dom0, e all'interno della macchina virtuale *lists* la partizione fisica sarà vista come disco *hdb*.

3.3 Startup macchine virtuali

Si può configurare il dom0 in modo tale che allo shutdown vengano prima spente le macchine virtuali in esecuzione. Questo si configura in */etc/sysconfig/xendomains*. Generalmente questa opzione è settata come default.

Per far sì che le macchine virtuali sul dom0 vengano create allo startup di Xen dom0, occorre creare un link simbolico al file del profilo delle macchine virtuali in */etc/xen/auto/* per ogni macchina virtuale che si vuole in modalità di auto-startup.

4 XEN 3.1.0 E WINDOWS

E' possibile utilizzare Xen come hypervisor sul quale girano macchine virtuali Windows. Questo permette di consolidare servizi quali Domain Controller.

4.1 Windows 2003 Server

Ad esempio installiamo un Domain Controller Windows Server 2003. Prima di tutto occorre creare un file *sparse* che conterrà l'immagine del sistema operativo a installazione ultimata:

```
cd /xen/domU
dd if=/dev/zero of=winsrc2003.img bs=1k seek=20479k count=1
```

Poi creiamo un file di profilo per l'installazione della macchina virtuale:

```
name = "winsrv2003"
builder = "hvm"
memory = "1024"
disk = [ 'file:/xen/domU/winsrv2003.iso,hdc:cdrom,r', \
'file:/xen/domU/winsrv2003.img,hda,w', ]
boot = "d"
vif = [ 'mac=00:16:3e:3f:a9:16, bridge=xenbr0', ]
vfb = ["type=vnc,vncunused=1"]
```

```
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
```

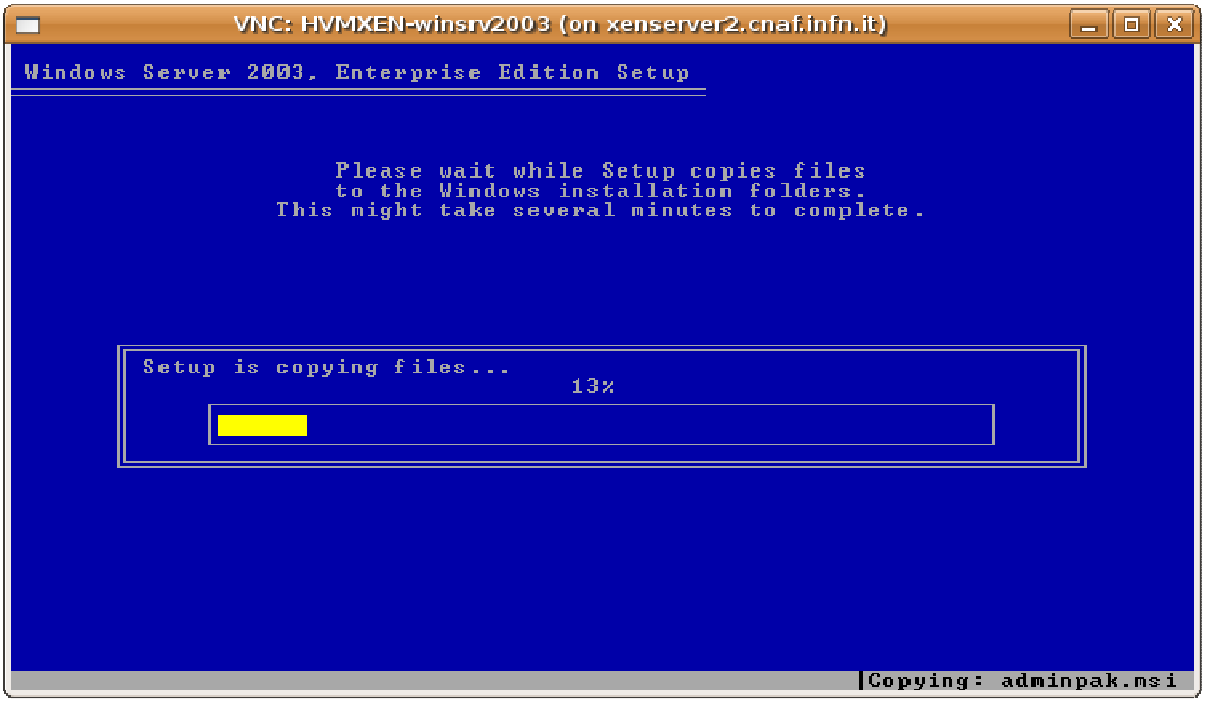


Figura 3

Specifichiamo che come disco virtuale si utilizzi l'immagine iso di Windows Server 2003 come supporto di boot e che venga emulato come CDROM, e il file di immagine appena creato sarà il disco di sistema. Lanciamo la macchina virtuale:

```
xm create /etc/xen/winsrv2003
```

Come si può vedere in figura 3 attraverso vnc siamo collegati alla console e stiamo installando Windows 2003. Ad installazione ultimata si può eliminare la entry relativa al dispositivo virtuale CDROM e all'immagine iso di installazione dal file di profilo della macchina e si fa ripartire il guest Windows 2003. Come mostrato in figura 4 abbiamo un sistema perfettamente funzionante.

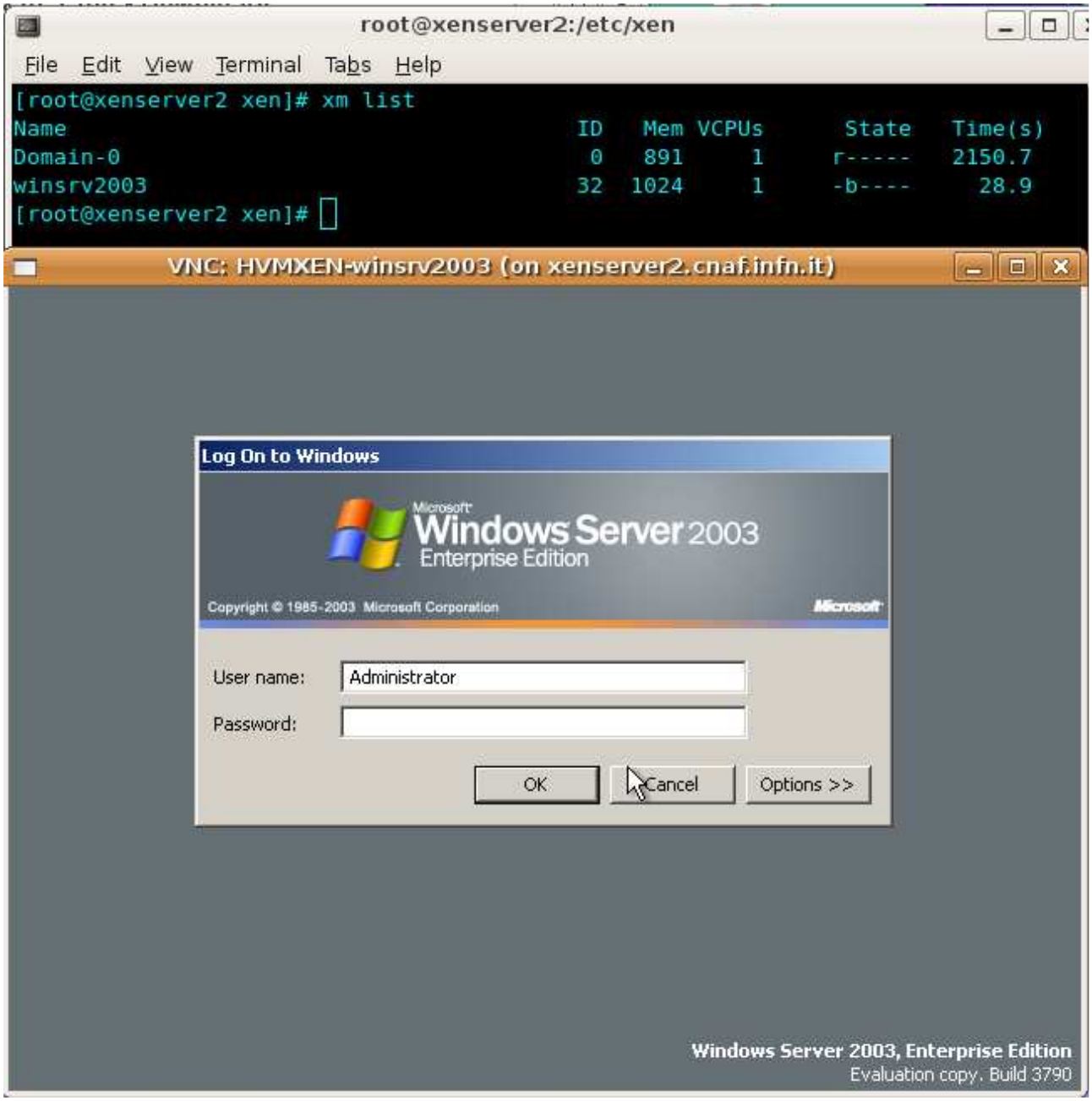


Figura 4

4.2 Windows Vista

Per installare Windows Vista procediamo in maniera analoga a Windows 2003 Server.

```
cd /xen/domU
dd if=/dev/zero of= vista-enterprise.img bs=1k seek=102399k count=1
```

Creiamo un file di profilo per l'installazione della macchina virtuale Vista:

```
name = "vista-enterprise"
```

```
builder = "hvm"
memory = "2048"
disk = [ 'file:/xen/domU/vista-
enterprise.iso,hdc:cdrom,r','file:/xen/domU/vista-enterprise.img,hda,w', ]
boot = "d"
vif = [ 'mac=00:16:3e:4f:b9:17, bridge=xenbr1', ]
vfb = ["type=vnc,vncunused=1"]
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
```

Il sistema si installa senza problemi, e come si vede in figura 5 vi accediamo tramite connessione vnc alla console. Ad installazione ultimata basta rimuovere dal file di profilo la entry del CDROM virtuale.

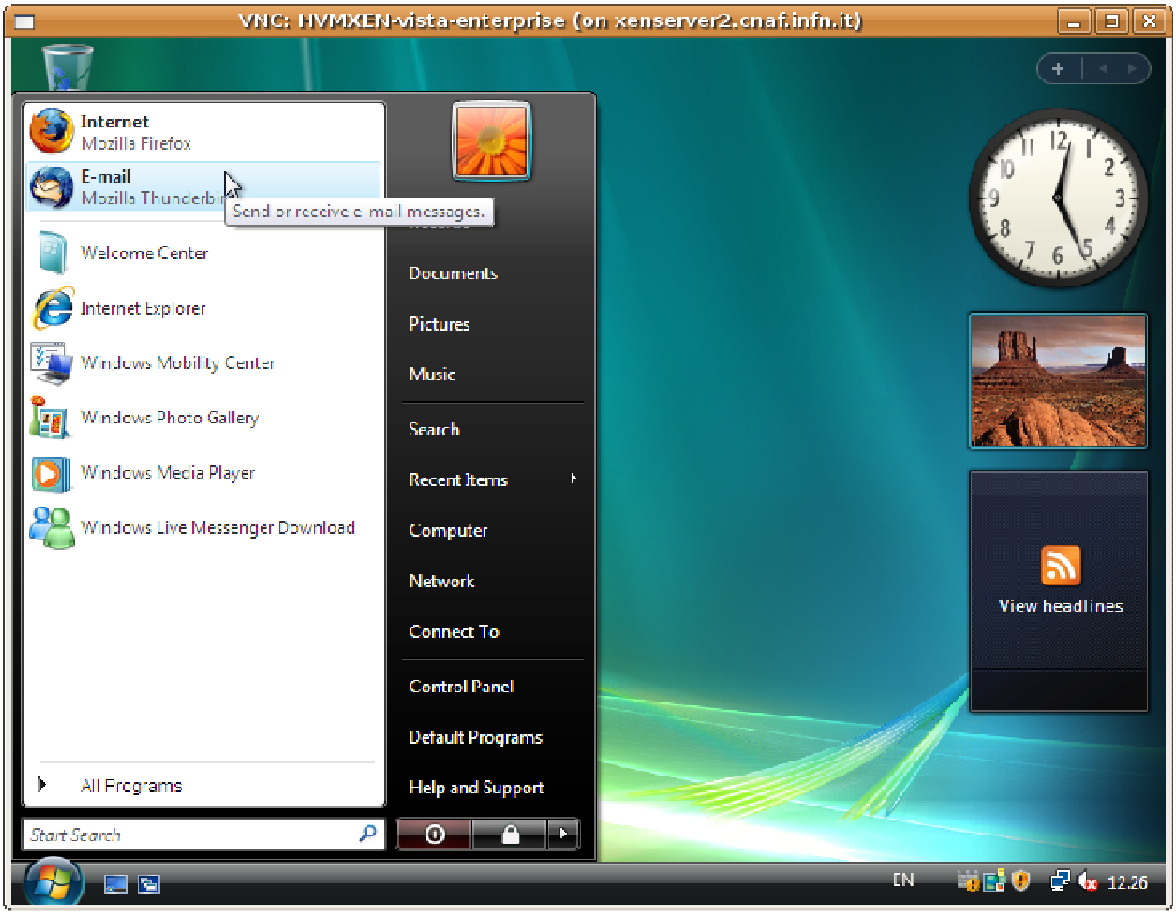


Figura 4

Ci si accorge subito però che il dispositivo di rete non funziona. Il problema è che la scheda di rete virtuale emulata RLT8139 non funziona correttamente in Vista versione 32 bit. Con la versione a 64 bit non ci sono problemi.

Per risolvere questo baco si può procedere in questo modo:

Si forza Xen ad utilizzare il driver NE2000 per l'emulazione dell'interfaccia di rete con quemu. Il problema è che in questo caso la scheda funzionerà a 10Mbit, ma la velocità è

comunque accettabile se non si necessita di una banda eccessiva.

Per abilitare l'emulazione NE2000 procediamo come segue.

Modifichiamo il profilo della macchina guest che chiamiamo **vista-enterprise**. Utilizziamo un'immagine iso che contiene i driver NE2000 e la rendiamo visibile alla macchina guest Vista come un CDROM virtuale:

```
name = "vista-enterprise"
builder = "hvm"
memory = "2048"
disk = [ 'file:/xen/domU/drivercd.iso,hdc:cdrom,r', 'file:/xen/domU/vista-
\ enterprise.img,hda,w', ]
vif = [ 'mac=00:16:3e:4f:a9:17, bridge=xenbr1, model=ne2k_pci', ]
vfb = ["type=vnc,vncunused=1"]
device_model = "/usr/lib64/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
```

Come si vede dal profilo *model=ne2k_pci* abilita l'emulazione dell'interfaccia di rete con il driver NE2000 di qemu per Xen, e drivercd.iso è il nostro CDROM virtuale che contiene i driver NE2000 per Windows 2000. Questo driver installato su Vista funziona ugualmente. Windows Vista non contiene più i driver per supportare questo dispositivo di rete. Se accendiamo la macchina virtuale con *xm create /etc/xen/vista-enterprise*, e accediamo al CDROM virtuale possiamo installare i driver della nuova scheda. Si riavvia il sistema operativo eliminando *'file:/xen/domU/drivercd.iso,hdc:cdrom,r'* dal profilo e ora la scheda di rete che vede Windows Vista è funzionante.

5 CONCLUSIONI

Xen consente l'implementazione di macchine virtuali in modo molto efficiente. La macchina guest grazie alla virtualizzazzione hvm ha pieno possesso della CPU a lei riservata. E' possibile riservare alla macchina virtuale in esecuzione su Xen hypervisor dispositivi fisici in modo esclusivo in modo da incrementare ad esempio le prestazioni di I/O su disco. Dove le prestazioni dell'hardware non sono un requirement così stringente, il sistema operativo della macchina virtuale può risiedere in un unico file con benefici riguardo alla possibilità di backup e ripristino in caso di disaster recovery, oppure parte del sistema può risiedere su un file e parte su una partizione fisica del file system.

La versione di Xen 3.1.0 trattata in particolare in questo documento ha caratteristiche importanti che mancano a Xen 3.0 quali il fatto di potere girare macchine virtuali miste a 32 o 64 bit. Questo la rende molto utile per il deployment di servizi su nuovo hardware migrati da hardware obsoleto.

Xen 3.1.0 poi ha caratteristiche che lo rendono ideale come base per servizi di alta affidabilità per macchine virtuali hvm. E' possibile migrare una macchina virtuale in esecuzione tra due Xen hypervisor, quindi tra due macchine fisiche distinte. Queste caratteristiche di Xen non sono però trattate in questo documento.