Universita degli Studi di Padova
Facolta di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

“Industrializzazione di un sistema di paravirtualizzazione per l'erogazione di servizi web”

Anno Accademico 2010-2011

Laureando
Marani Gian Piero

Relatore
Prof. Sergio Congiu

Indice generale

1. Introduzione

Questo elaborato presenta il lavoro di tesi e stage svolto presso la sede di Polesine Innovazione di Rovigo nel periodo febbraio – aprile 2010.

1.1 Azienda

Polesine Innovazione è un'azienda speciale della Camera di Commercio di Rovigo che opera dalla fine del 1986 a favore dello sviluppo delle imprese della provincia con un'attività di servizi di terziario avanzato e di ricerca applicata.

L'azienda opera nell'ambito della formazione, si occupa di tutela ambientale e di sicurezza del lavoro, è attiva sul fronte dell'innovazione tecnologica e dei servizi telematici.

Nel Consiglio di Amministrazione di Polesine Innovazione sono presenti le Associazioni di categoria Artigiane, Industriali, Agricole e Commerciali oltre a due Istituti di credito e all'Amministrazione provinciale.

1.2 Obiettivi

L'obiettivo dello studio è la predisposizione e l'utilizzo della piattaforma XEN di virtualizzazione al fine di verificarne l'effettiva efficacia nell'erogazione di servizi web.

Nello specifico l'attenzione si è concentrata sulla verifica dei requisiti di alta affidabilità, efficienza e sicurezza, indispensabili per l'erogazione dei servizi all'utenza.

1.3 Struttura della tesi

La tesi è strutturata in due “sezioni” principali ed in un insieme di sottosezioni.

Le sezioni strutturano l'attività che è stata svolta;

la prima accoglie e sviluppa lo studio della tecnica e degli strumenti di virtualizzazione, mentre la seconda dettaglia il lavoro svolto, gli strumenti utilizzati fornendo evidenza dei risultati ottenuti.

Al termine dello studio, una raccolta degli script più importanti utilizzati nell'attività, ed i riferimenti bibliografici.

2. Virtualizzazione

2.1 Definizione

Per virtualizzazione si intende l'astrazione di una risorsa fisica attraverso un'interfaccia logica che ne nasconda i dettagli implementativi.

Nel caso specifico della virtualizzazione di sistemi informatici, consiste nella predisposizione ed utilizzo di intere macchine virtuali, differenti per sistema operativo ed applicazioni installate, sul medesimo sistema hardware.

La predisposizione di un tale sistema viene realizzata mediante utilizzo di uno strato software intermedio detto VMM (*Virtual Machine Monitor*) che ha l'effetto di controllare e “regolamentare” (utilizzare anche altra terminologia) l'accesso delle macchine virtuali alle risorse fisiche della macchina.

2.2 Virtual Machine

Per spiegare al meglio come si svolge tale processo si è deciso di introdurre alcuni concetti di Macchina Virtuale e VMM.

La prima definizione di VM risale al 1974 quando Gerald J. Popek e Robert P. Goldberg scrissero un articolo in cui gettaronoasi teoriche della virtualizzazione.

Benché si riferissero a sistemi di terza generazione, costruiti con i primi circuiti integrati, l'analisi che fecero può essere considerata valida anche per i moderni sistemi di quarta generazione, dotati di microprocessori.

La definizione che fu data di macchina virtuale è la seguente:

“*Un duplicato software di un computer reale nel quale un sottoinsieme statisticamente dominante di istruzioni del processore virtuale viene eseguito nativamente sul processore fisico”*

2.3 Virtual Machine Monitor

Virtual Machine Monitor o VMM è un software che risiede e viene eseguito su una macchina reale e che ha il completo controllo delle risorse hardware.
Esso definisce regola e controlla degli ambienti di esecuzione virtuali (VM) che forniscono agli utenti un accesso diretto ed esclusivo alle risorse della macchina fisica.

Le caratteristiche principali del VMM sono:

- Equivalenza: Gli effetti dell'esecuzione di un programma attraverso il VMM devono essere identici a quelli dello stesso programma eseguito direttamente sulla macchina originale (ad eccezione del maggior tempo d'esecuzione dovuto all'overhead del VMM e alla ridotta disponibilità di risorse);

- Efficienza: La maggior parte delle istruzioni eseguite all'interno di tali ambienti deve essere eseguita direttamente sul processore fisico senza l'intervento del VMM (vedi definizione di Popek-Goldberg), ottenendo di conseguenza un degrado minimo delle prestazioni;

- Controllo delle risorse: il controllo delle risorse deve essere di esclusiva competenza del VMM.

- Il VMM è composto da diversi moduli:

- Dispatcher: intercetta le istruzioni sensibili eseguite dalle VM (vedi capitolo seguente “Requisiti di Popek-Goldberg”) e lascia il controllo ai moduli predisposti a gestire la situazione;

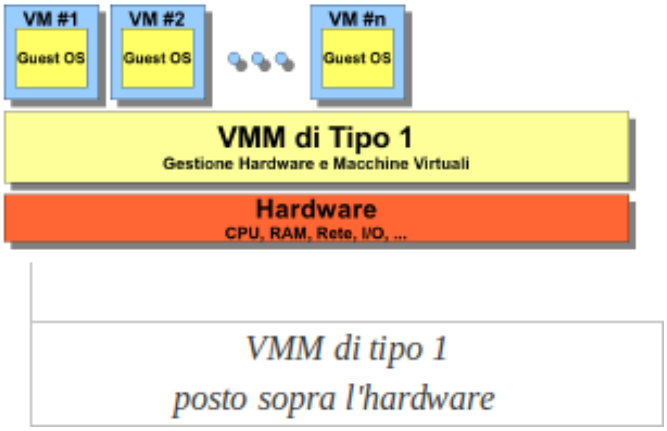
- Allocator: fornisce alle macchine virtuali le risorse necessarie evitando i conflitti (gestione delle risorse hardware);

- Interpreter: simula le istruzioni che fanno riferimento alle risorse in modi da riflettere la loro esecuzione nell'ambiente delle macchine virtuali (indispensabile in quanto le VM non hanno accesso diretto alle risorse fisiche).

• Lo scopo della virtualizzazione dei sistemi e quello di condividere in modo trasparente le risorse hardware di una macchina fisica tra i sistemi operativi di più macchine virtuali, di far eseguire loro piu istruzioni possibili direttamente sul processore senza interazione del VMM e di risolvere potenziali cause di malfunzionamenti dovute alle eventuali mancanze dell'architettura nei confronti dei requisiti di Popek-Goldberg.

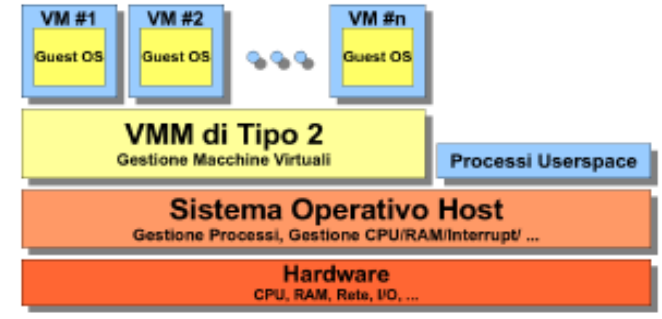
Si possono individuare due tipologie di VMM, in base alla collocazione nell'ambiente della macchina fisica:

• Il VMM di tipo 1 è posto immediatamente sopra l'hardware ed dispone di tutti i meccanismi di un normale kernel o sistema operativo in quanto a gestione delle memoria, delle periferiche e del processore; in più implementa i meccanismi di gestione delle macchine virtuali. Le macchine virtuali eseguite al di sopra del VMM dispongono di un proprio sistema operativo e sono dette macchine “guest".



VMM di tipo 1
posto sopra l'hardware

• Il VMM di tipo 2 è un normale processo in esecuzione nell'ambito di un sistema operativo detto “host". Gestisce direttamente le macchine virtuali, che sono suoi sottoprocessi, mentre la gestione dell'hardware e demandata al sistema ospitante.



VMM di tipo 2
ospitato nel sistema operativo

2.4 Requisiti di Popek-Goldberg

La maggior parte dei processori moderni prevedono il funzionamento in almeno due modalità operative: *user-mode* e *supervisor-mode.* Il set di istruzioni del processore in use-mode è limitato a quelle che non presentano problemi di sicurezza, vale a dire le istruzioni che non modificano lo stato delle risorse di sistema (es. non modificano lo stato dei registri della CPU); viceversa in supervisor-mode può essere eseguita qualunque istruzione.
Lo stato del processore dipende dal processo che lo sta utilizzando: tipicamente solo il kernel può utilizzarlo in supervisor-mode.

Se il processore e in stato di user-mode ma tenta di eseguire un'istruzione che puo essere eseguita solo in stato di supervisor-mode, viene generato un interrupt che deve essere gestito dal sistema operativo.

Ai fini della virtualizzazione il set di istruzioni può essere suddiviso in tre sottoinsiemi, non necessariamente disgiunti:

- Privilegiate: istruzioni eseguibili solo in supervisor-mode;
- *Sensibili:* istruzioni che modificano lo stato delle risorse del sistema;

- *User:* tutte le altre istruzioni.

Data questa definizione, Popek e Goldberg formularono una condizione sufficiente per la virtualizzazione di un sistema:

“La costruzione di un VMM è sempre possibile se il set di istruzioni sensibili del calcolatore è un sottoinsieme delle sue istruzioni privilegiate”
In pratica un VMM può essere costruito su qualunque computer a patto che le istruzioni che modificano o dipendono dallo stato della macchina reale causino un interrupt se eseguite all'interno di una VM (che opera in user-mode).
In questo modo l'interrupt può essere intercettato dal dispatcher e gestito opportunamente.
Allo stesso modo le istruzioni del set user-mode di una VM verranno eseguite direttamente sul processore fisico, senza la mediazione del VMM.
Se i requisiti di Popek-Goldberg sono rispettati, il VMM e in grado di soddisfare i tre requisiti di equivalenza, ecienza e controllo delle risorse sopra esposti.

Problemi relativi all'architettura x86

L'architettura x86, creata da Intel alla fine degli anni '70, è attualmente la più diffusa nei sistemi che si utilizzano per la virtualizzazione (desktop, laptop, server di medio-piccole dimensioni).

Il grande limite di questa architettura è che non rispetta le condizioni di Popek-Goldberg: nel set di istruzioni del processore sono presenti istruzioni sensibili non privilegiate.
Se una VM esegue un'istruzione sensibile non privilegiata, essa sfuggirà al controllo del VMM in quanto non verrebbe generato nessun interrupt: questo si traduce nel mancato rispetto dell'essenziale requisito del controllo esclusivo delle risorse da parte del VMM.

Per sopperire a questa mancanza sono state sviluppate tecniche che consentono in qualche modo di intercettare anche queste istruzioni potenzialmente dannose e che permettono quindi di virtualizzare anche architetture x86, con impatti piu o meno negativi sulle performance.
Alcune di queste tecniche verranno discusse nel paragrafo seguente.

2.5 Tipologie di Virtualizzazione

Esistono molteplici tipologie di virtualizzazione di sistemi quali:

- Virtualizzazione completa
- Paravirtualizzazione
- Virtualizzazione a livello kernel
- Virtualizzazione hardware
- Emulazione

2.5.1 Full-virtualization e Para-virtualization

La virtualizzazione completa (Full-virtualization) consiste nella completa simulazione software delle risorse hardware. Per far questo ogni istruzione eseguita dalle VM deve essere intercettata e, se è sensibile, tradotta a runtime per mezzo di meccanismi software detti “binary-translation". Il VMM è tipicamente di tipo 2, ovvero un processo eseguito all'interno dell'ambiente del sistema operativo host. Il controllo di tutte le istruzioni permette di aggirare i limiti delle architetture x86 ma provoca inevitabilmente un degrado consistente delle prestazioni.

La tecnologia di Paravirtualizzazione (Para-virtualization) offre prestazioni decisamente migliori rispetto alla virtualizzazione completa, appena al di sotto degli stessi sistemi non virtualizzati, grazie all'utilizzo di un VMM di tipo 1 e all'utilizzo di speciali chiamate di sistema, dette *hypercalls*.

Il VMM di tipo 1 utilizzato, detto *hypervisor*, gestisce direttamente le risorse hardware e fornisce alle VM un'interfaccia software simile a quella di un normale sistema operativo; la comunicazione tra VM e hypervisor avviene tramite le hypercalls che sostituiscono le *supercalls* utilizzate nei sistemi operativi dalle applicazioni che richiedono l'accesso all'hardware.

Con questo meccanismo è possibile implementare questa tecnologia anche sulle architetture x86, in quanto le hypercalls consentono di intercettare qualunque

istruzione sensibile eseguita dalle VM, permettendo contemporaneamente alle stesse l'esecuzione di istruzioni non sensibili (user) direttamente sul processore senza l'intervento del VMM.
L'unico svantaggio di questa tecnologia è che i sistemi guest devono essere opportunamente modificati per poter utilizzare le hypercalls, il che riduce il numero e il tipo di sistemi operativi effettivamente utilizzabili (i software proprietari non sono utilizzabili in quanto è necessario avere accesso al codice sorgente per effettuare le modiche).

2.5.2 Altre tipologie

2.5.2.1 Virtualizzazione a livello kernel

In questa tipologia le funzionalità per la virtualizzazione sono offerte direttamente dal normale kernel del sistema operativo. Le macchine virtuali così create possono avere ognuna il proprio file-system, ma condividono il kernel del sistema operativo host.

Lo svantaggio di questo approccio è che possono essere avviate macchine virtuali con un solo tipo di sistema operativo e che un eventuale falla di sicurezza nel kernel si ripercuoterebbe su tutte le VM.

2.5.2.2 Virtualizzazione hardware

Dal 2005 Intel e AMD hanno cominciato a dotare alcuni dei processori da loro prodotti di estensioni specifiche per gestire alcuni aspetti della virtualizzazione direttamente a livello hardware.
Queste tecnologie, la *VT* di Intel e la *Pacifica* di AMD, implementano parzialmente nel processore complessi meccanismi per la gestione delle istruzioni sensibili non privilegiate o per la traduzione degli indirizzi di memoria delle VM in indirizzi fisici, funzioni che solitamente spettano al VMM.

Piuttosto che essere considerato una tipologia a sè stante, si può dire che il supporto hardware alla virtualizzazione estende le tecnologie già esistenti; in particolare, su macchine dotate di questi processori, si possono, teoricamente,

para-virtualizzare anche sistemi operativi non modicati.
Questo apre la porta della virtualizzazione con buone performance anche ai sistemi a codice sorgente chiuso e quindi non modificabili, come quelli della famiglia Windows.

2.5.2.3 Emulazione

L'emulazione non è una tipologia di virtualizzazione, ma poiché i due termini a volte vengono usati indifferentemente in quanto anche l'emulazione permette di eseguire diverse macchine virtuali su un sistema operativo host, fornendo un'interfaccia software verso le risorse fisiche.

A differenza della virtualizzazione, l'emulazione non ha lo scopo di eseguire piu operazioni possibili nativamente sul processore fisico; al contrario è certo che tutte le istruzioni saranno controllate ed eventualmente tradotte prima di essere eseguite. Questo permette di eseguire anche ambienti compilati per architetture diverse da quelle della macchina host (ad esempio di eseguire sistemi operativi e applicazioni compilati per architetture PowerPc su processori x86 e viceversa); in pratica un'istruzione di un architettura può essere convertita in una o più istruzioni equivalenti di un'altra architettura e poi eseguita.

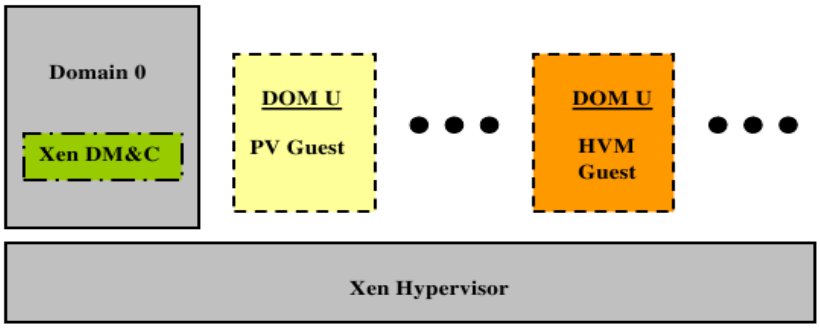
La virtualizzazione invece non prevede questa eventualità, ma i sistemi operativi guest devono essere compilati per la stessa architettura della macchina fisica ospitante. Tra i prodotti per l'emulazione più noti si possono ricordare QEMU, in ambiente Linux, e Microsoft Virtual PC, in ambiente Windows.

2.6 Xen

2.6.1 Architettura

Xen è un sistema di virtualizzazione opensource composto da vari elementi:

- Hypervisor;
- Domain 0 (Dom0);
- Domain U (DomU);
- Domain Managment and Control (DM&C).



Schema che raffigura la struttura degli elementi sopracitati

Hypervisor

L'hypervisor di xen è stato realizzato seguendo alcuni dei principi del VMM di tipo 1 enunciati da Popek-Golberg: esso è posto direttamente sopra l'hardware la sua funzione primaria è la gestione delle VM, che possono essere fermate o riavviate come fossero processi di un sistema operativo; oltre a questo si occupa dello scheduling della cpu e della suddivisione della memoria centrale tra le Virtual Machine in esecuzione, che posso accedervi solo tramite i servizi offerti dell'hypervisor.

La differenza con il modello teorico di VMM sta nel fatto che esso non si occupa dei dispositivi di storage, dei dispositivi di rete e dei dispositivi di I/O, la cui gestione è demandata al Dom0.

Domain 0

Il Domain 0 è una VM in esecuzione sulla macchina su cui è installato un sistema operativo modificato opportunamente per dialogare con l'hypervisor, tipicamente unix-like (linux NetBSD e solaris). La sua funzione è fornire un'interfaccia di collegamento tra i vari DomU e i dispositivi hardware di rete, storage e I/O. Questo permette di semplificare enormemente il lavoro dell'hypervisor e al tempo stesso di sfruttare l'enorme quantità di driver del sistema operativo installato.

Per permettere ai DomU l'accesso all'hardware il Dom0 utilizza due particolari driver di backend, Network Backend Driver e Block Backend Driver, che virtualizzano rispettivamente le periferiche di rete e i dispositivi a blocchi.

Domain U

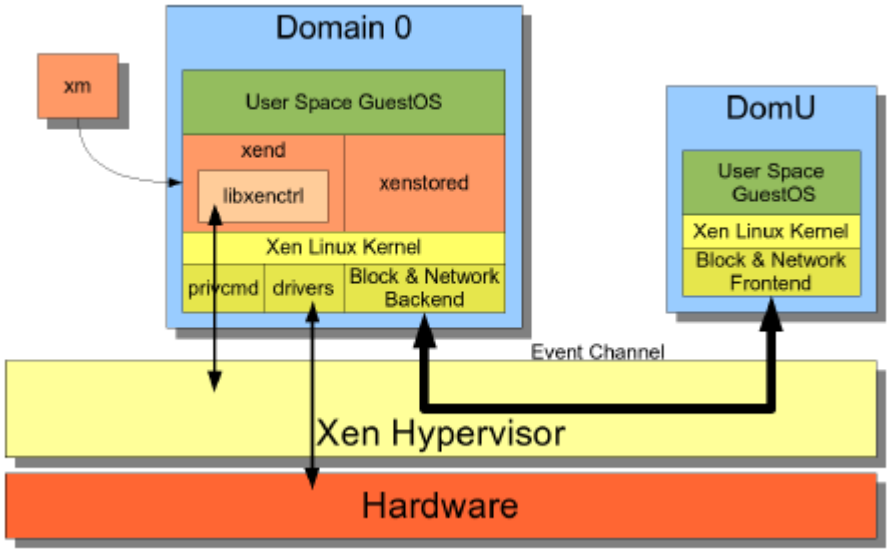
Con il termine DomU vengono indicate tutte le VM presenti sulla macchina fisica e controllati dall'hypervisor (ad eccezione del Dom0). Sono presenti due tipi di DomU, a seconda del tipo di virtualizzazione: DomU PV Guest e DomU HVM Guest, rispettivamente con para-virtualizzazione e virtualizzazione completa; queste due tipologie di DomU differiscono tra loro dal modo con cui comunicano con il Dom0 e l'hypervisor.

I DomU PV guest utilizzano i driver di frontend, controparte dei driver di backend, per accedere alle risorse tramite il Dom0; i DomU HVM guest invece non avendo a disposizione i driver di frontend, sfruttano un particolare *daemon process* del Dom0 (ne viene inizializzato una per ciascuna VM), Qemu-dm, che si occupa di intercettare le richieste di accesso all'hardware. Inoltre i DomU HVM guest per poter essere inizializzati come fossero vere macchine fisiche necessitano di software supplementare rispetto alle PV guest: lo Xen virtual firmware, che è un BIOS virtuale che permette alle virtual machine di effettuare le operazioni di post e boot.

Domain Management and Control

I DM&C sono una serie di demoni (daemon processes) tramite cui viene gestito l'ambiente virtuale. Tra di esse il principale è il demone *xend*. Xend ha il compito di fungere da tramite tra l'amministratore del sistema e l'hypervisor che ne gestisce l'allocazione delle risorse a basso livello. Esso fa uso della libreria *libxenctrl* per dialogare con lo speciale driver *privcmd*, che rappresenta l'interfaccia di comunicazione attraverso il quale il Dom0 comunica all'hypervisor le richieste dell'amministratore. Queste richieste vengono effettuate attraverso l'applicazione a riga di comando *xm*, che consente di creare, spegnere, riavviare, distruggere e monitorare le macchine virtuali.

Accanto xend vi e il demone *xenstored* che gestisce parte dello scambio di informazioni tra il Dom0 e i DomU e mantiene delle strutture dati necessarie per effettuare lo scambio da e verso i dispositivi di rete e a blocchi per conto dei DomU. La gestione dei dispositivi infatti avviene con la mediazione di quest'ultimo demone, responsabile della corretta gestione dei canali di comunicazione, detti *event channel*, che come detto sostituiscono gli interrupt con un meccanismo ad eventi. Lo scambio di dati vero e proprio da e verso i dispositivi avviene in modo asincrono e per mezzo di aree di memoria condivisa sotto l'arbitrio dell'hypervisor, che assicura l'isolamento degli spazi di memoria delle VM.



Schema dell'architettura Xen con DomU PV Guest

3. Piano di attività

In questo capitolo verrà trattato il lavoro svolto, a partire dalle motivazioni che stanno alla base del lavoro svolto, alla strumentazione utilizzata fino alle problematiche affrontate e ai test eseguiti.

3.1 Vantaggi della virtualizzazione

I vantaggi dell'utilizzo di una piattaforma di virtualizzazione sono molteplici ad esempio nel campo di consolidamento server, dove si deve considerare innanzitutto la riduzione dei costi di

- Acquisto di elaboratori: potendo virtualizzare le macchine fisiche il numero effettivo di elaboratori necessari all'attività di un'azienda si riduce enormemente; ad esempio se prima venivano utilizzati sei server per l'erogazione di servizi, dopo aver virtualizzato per mantenere invariato il carico di lavoro ne sarebbero sufficienti uno o due più eventualmente uno aggiuntivo di backup;

- Consumi: riducendo il numero di elaboratori viene ridotto notevolmente il quantitativo di energia elettrica utilizzata per il loro funzionamento e per l'impianto di raffreddamento, senza contare inoltre il recupero di volume (spazio rack);

- Manutenzione: vi è una effettiva riduzione del tempo necessario per svolgere le operazioni sistemistiche più comuni, quali installazione, configurazione, replica e backup, in quanto il fatto di utilizzare un sistema di virtualizzazione implica che quello che le VM identificano come il proprio dispositivo di massa, altro non è che un file di una certa dimensione (alcuni Gigabyte) cui è associato un file di configurazione proprio della macchina virtuale stessa; quindi interagendo con questo file, che funge da hard disk virtuale, è possibile svolgere le funzioni sopracitate con sforzo e tempi ridotti al minimo.

Un ulteriore vantaggio è l'aumento della disponibilità dei servizi erogati, in quanto si ottiene

- Maggiore tolleranza ai guasti: riducendo il numero di elaboratori la probabilità di guasti viene automaticamente ridotta;

- Riduzione dei tempi di downtime;

- Disaster recovery: le procedure di disaster recovery vengono notevolmente semplificare e velocizzate grazie all'utilizzo delle tecniche e degli strumenti di live migration;

Altro fattore importante è il bilanciamento del carico (*load balancing*), vale a dire la possibilità di spostare l'esecuzione di una VM da un server, a pieno carico, ad un altro, in tempo reale e senza interruzione di servizi tramite la live migration.

Inoltre la possibilità di associare le risorse hardware alle VM adattandole in base alle esigenze (dimensionamento delle risorse) ne consente un utilizzo ottimale.

Un'ulteriore applicazione della virtualizzazione consiste nella possibilità, molto apprezzata da sistemisti e programmatori, di testare nuove soluzioni, prima di metterle in produzione, in tutta sicurezza e su una moltitudine di ambienti diversi.

3.2 Hardware e Software utilizzati

Nell'ambito del lavoro svolto sono state utilizzate differenti macchine fisiche allo scopo di verificare le modalità operative e le performance del sistema Xen.

Workstation IBM xseries 206

- 2.8 Pentium 4 with 1 MB L2 cache and 800 Mhz front side bus
- 1 GB of PC2700 CL2.5 ECC DDR SDRAM UDIMM system memory;
- two 40 GB hard disks 7200 RPM
- Bootable 48x-20x IDE CD-ROM
- 340-watt, voltage-sensing power supply and two cooling fans
- I/O ports:
  - Four USB
  - Integrated Gigabit Ethernet with RJ-45
  - Two high-speed NS16550A software-compatible serial ports
  - One high-speed parallel port supporting devices
- Integrated ATI 7000M and 16 MB video memory

Server Dell Poweredge R710

- Intel Xeon E5506, 4C, 2.13GHz, 4M Cache, 4.80GT/s, 80W TDP, DDR3-800MHz
- 16 GB Memory , DDR3, 800 MHz
- three 146GB, SAS 3Gbps, 3.5-in, 15K RPM Hard Drive (Hot Plug)
- 16X DVD+/-RW ROM Drive SATA
- High Output Power Supply, Redundant (2 PSU), 870W, Performance BIOS Setting
- I/O ports:
  - Four USB
  - Embedded Gigabit Ethernet NIC with 4P TOE
  - Broadcom NetXtreme II 5709 Dual Port 1GbE NIC with TOE, PCIe-4

Per quanto riguarda il software è stata selezionato un sistema open quale CentOS, versione 5.3, distribuzione derivata da Red Hat Enterprise Linux e quindi concepita per fornire una piattaforma di classe enterprise per chiunque intenda utilizzare GNU/Linux per usi professionali; nel caso specifico per server virtualization.

Il motivo di questa scelta è dovuto al fatto che questa distribuzione permette di scegliere, già nella fase di installazione, se utilizzare o meno un sistema di virtualizzazione basato su Xen; scelta non consetita da altri sistemi operativi.

CentOS infatti installa e configura automaticamente l'hypervisor (versione 3.2) e svolge la funzione di Dom0 grazie al kernel modificato *vmlinuz-2.6.18-8.1.4.el5xen.* Per la workstation ed il server sono state usate rispettivamente le versioni a 32 e 64 bit.

Per quanto riguarda i DomU PV guest è stata utilizzata ancora la distribuzione CentOS, mentre per i DomU HVM guest si è ricorso al sistema operativo Windows XP SP2 a 32 bit.

3.3 Test

3.3.1 Prestazioni

Sebbene non esistano benchmark universalmente accettati per la valutazione delle prestazioni delle tecnologie di virtualizzazione, si può affermare che è necessario tenere in considerazione due aspetti fondamentali che sono l'impatto minimo e la scalabilità del sistema.

L'impatto minimo riguarda il tempo di esecuzione di un programma di test in una macchina virtuale che *deve* avvicinarsi quanto più possibile al tempo che lo stesso programma impiega in un sistema non virtualizzato

La scalabilità in quanto parametro di qualità del sistema è tanto migliore quanto minore è la misura del degrado di performance.
Nel caso specifico tale degrado deve mantenersi in proporzione lineare rispetto all'incremento del numero di macchine virtuali.

Ciò è sinonimo di un'equa distribuzione delle risorse tra le macchine virtuali e un overhead praticamente trascurabile.

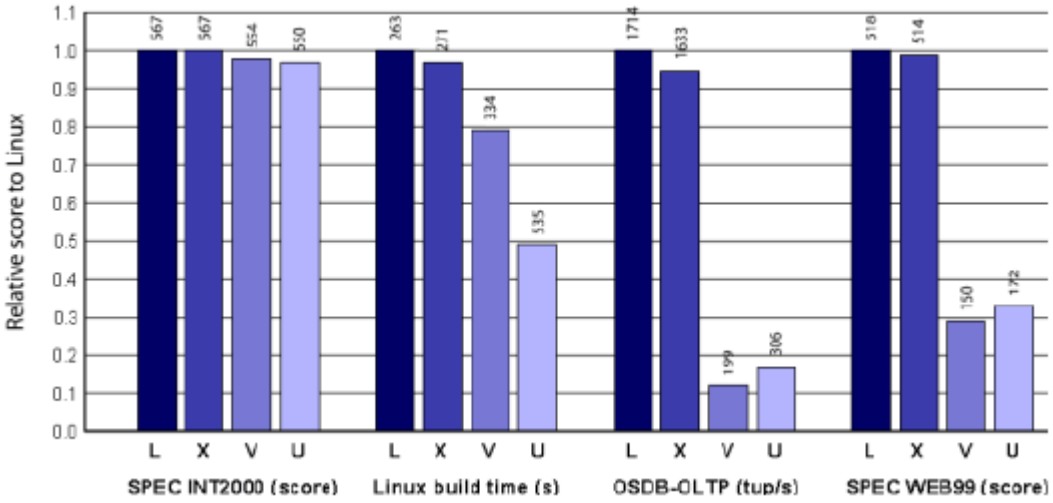
Il gruppo di ricerca di Xen ha effettuato alcuni test per valutare le prestazioni di un ambiente GNU/Linux virtualizzato tramite Xen in paragone con lo stesso ambiente non virtualizzato e con altre tecnologie di virtualizzazione.

In quel contesto sono state prese in considerazione situazioni ad alto tasso di utilizzo di CPU e RAM, altre con frequenti operazioni di I/O (tipicamente accessi ripetuti ad un database) e altre che contemplano entrambi i casi (ad esempio la compilazione del kernel, che utilizza intensamente la CPU ma esegue anche molte operazioni di lettura e scrittura su file).
I risultati evidenziano che in situazioni di utilizzo intensivo della CPU la differenza tra applicazione nativa e virtualizzata è quasi impercettibile, mentre per le operazioni di I/O il programma virtualizzato è penalizzato in misura inferiore al 5%. Inoltre le prestazioni di Xen sono notevolmente superiori a quelle di altri software di virtualizzazione.

Il secondo aspetto, la scalabilità, è stato tenuto in gran considerazione dai

progettisti di XEN.

I test effettuati a questo riguardo hanno rilevato che il tempo di calcolo dei programmi di test in molti casi cresce linearmente all'aumentare del numero di macchine virtuali: è il comportamento ottimale.



Risultati di alcuni test di benchmark effettuati dall'Università di Cambrige.

Vengono comparati un sistema Linux non virtualizzato (L), un sistema virtualizzato con Xen (X) e sistemi virtualizzati con altri due software molto popolari, VMware Workstation (V) e User Mode Linux (U). Il test SPEC INT2000 utilizza intensivamente la CPU, con rare operazioni di I/O: il valore riportato è il punteggio assegnato dal benchmark.

Un altro test ha riguardato il tempo di compilazione del kernel di Linux.
Il terzo test, OSDB-OLTP, è relativo alla valutazione delle performance di database, ovvero operazioni orientate all'I/O: il risultato è espresso in tuple accedute o modificate al secondo.

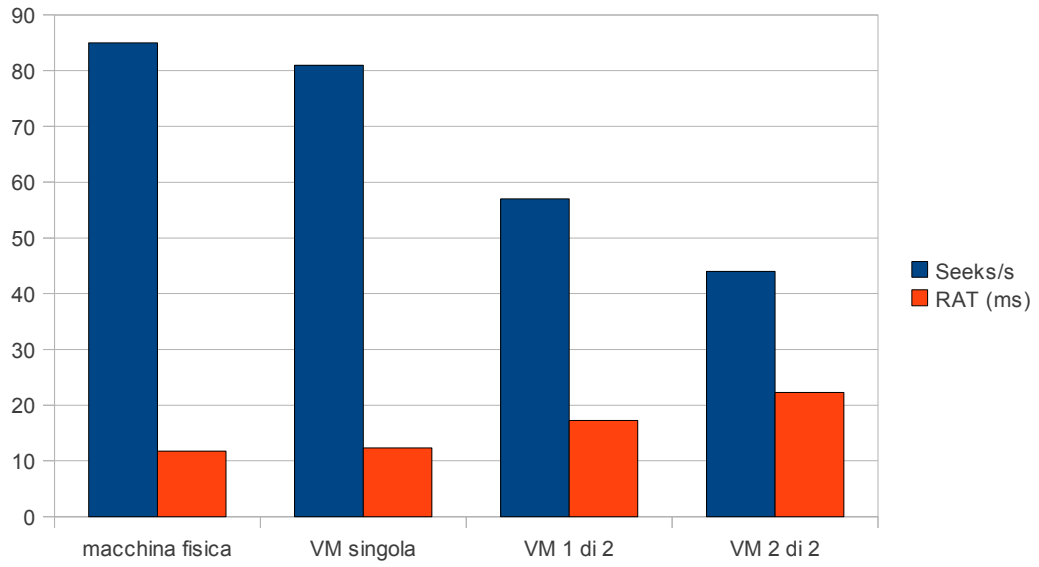
L'ultimo test, SPEC WEB99, riguarda la prontezza del sistema nel fornire servizi di web server: il valore indicato è il massimo numero di connessioni simultanee supportate.

Per confermare quanto dichiarato dai benchmark sopracitati è stato utilizzato un

piccolo script *seeker* che effettua accessi casuali al disco riportando il risultato in

- Seeks/s: accessi al secondo
- Rat: random access time, tempo di accesso casuale (in millisecondi)

Il primo test è stato eseguito direttamente sulla macchina fisica, il secondo su una singola VM in esecuzione e il terzo (VM 1 di 2 e VM 2 di 2) su due VM contemporaneamente.



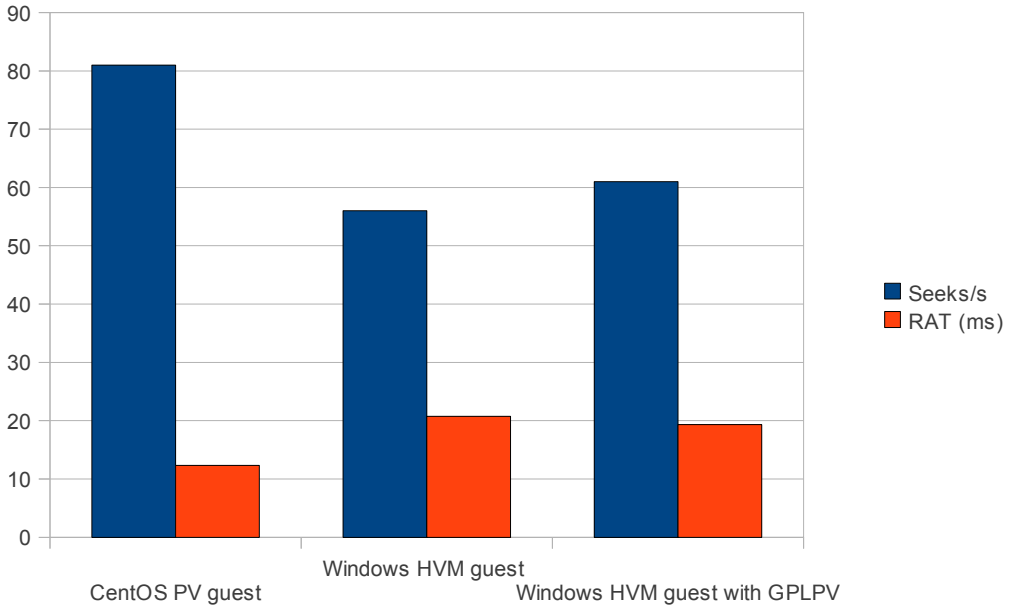
Il risultato corrisponde completamente alle aspettative: al crescere delle VM le prestazioni calano linearmente (tutte le vm testate sono PV guest).

Oltre ai DomU PV guest sono state testate le performance dei DomU HVM guest, su cui era stato installato Windows XP SP2.

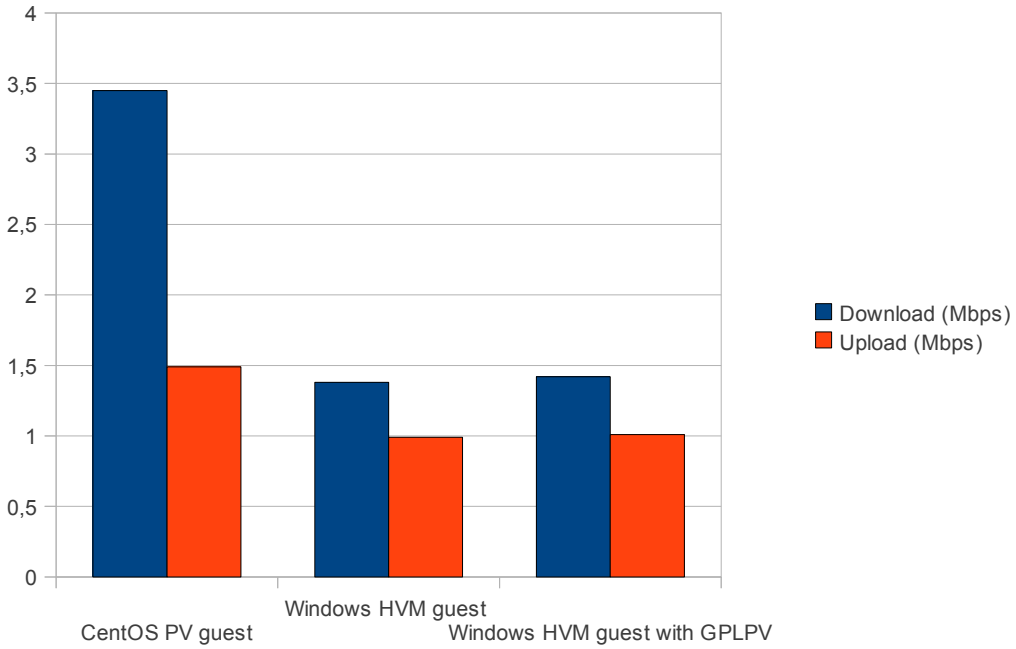
Le prestazioni dei DomU HVM Guest con installato Windows XP SP2 sono state, come da spettativa, decisamente inferiori rispetto ai DomU paravirtualizzati, quindi per cercare di migliorarne le performance si è25/02/2011 provato a testare i *GPLPV drivers*.

I GPLPV drivers sono dei driver per Windows che teoricamente permetterebbero ai DomU HVM guest di dialogare con il driver di backend del Dom0 come fossero PV guest, ofrendo quindi un sostanziale miglioramento prestazionale.

I risultati non hanno evidenziato differenze prima e dopo l'uso dei suddetti drivers (v. grafici seguenti).



Test eseguito utilizzando il programma Seeker



Test eseguito tramite il sito www.speedtest.net, utilizzando il server di Vicenza

3.3.2 Xen versione 4.0

Nell'aprile 2010 è stata rilasciata la quarta versione dell'hypervisor di Xen, che ha introdotto molti miglioramenti e nuove funzionalità, come un aumento delle proformance e della scalabilità del sistema di virtualizzazione.

Purtroppo il nuovo hypervisor non è supportato da CentOS a causa di alcuni driver mancanti alla distribuzione; per poterlo testare si è cercato di utilizzare altre distribuzioni, quali Ubuntu e Fedora, che teoricamente avrebbero consentito di virtualizzare la macchina fisica utilizzata, in tempi brevi e in modo relativamente semplice.

Per eseguire l'upgrade dell'hypervisor occorre:

- scaricare i sorgenti dell''hypervisor e del kernel aggiornato da utilizzare nel Dom0 (2.6.32, prim veniva utilizzato il 2.6.18);
- compilare l'hypervisor;
- configurare e compilare il kernel;
- aggiornare il boot loader (grub) in modo che all'avvio venga utilizzato il nuovo software.

La compilazione dell'hypervisor non ha dato problemi ed è stata effettuata in tempi piuttosto brevi (circa 40 minuti).

Il vero problema è stato configurare correttamente il kernel: nonostante i ripetuti tentativi non si è infatti riusciti a installare un kernel funzionante (anche a causa del limitato tempo a disposizione).

3.4 *Strumenti di gestione*

Al fine di utilizzare Xen per la virtualizzazione su vasta scala è necessario poter usufruire di alcuni strumenti che permettano di gestire le operazioni più comuni in modo semplice o addirittura automatico.

Purtroppo Xen non dispone di tali strumenti ma solo di servizi basilari messi a disposizione dai demoni DM&C e utilizzabili da riga di comando; questo fa sì che il compito di un amministratore di sistema diventi piuttosto oneroso, specialmente quando il numero delle VM da gestire è elevato.

Per ovviare a questo problema si è scelto di realizzare alcuni script che si occupino delle operazioni di backup, clonazione e ripristino delle virtual machine (dumpXenStore è uno script che visualizza lo stato dei domini attivi e le loro caratteristiche, scaricabile dal sito del progetto xen):

- xenBackup.sh
- dumpXenstore.sh
- restoreVM.sh
- cloneVM.sh

Questi script sono strutturati in modo tale da interagire con dei particolari file associati alle VM che svolgono la funzione di hard disk virtuale: i file *IMG* che si presentano nel formato *nomeVM.img*; questi file non si limitano a svolgere la funzione appena descritta ma contengono al loro interno le informazioni essenziali delle VM associate:

- Nome;
- Uuid (universally unique identifier);
- Quantità di ram associata (in MB);
- Numero di cpu virtuali associate;
- Immagine del kernel del DomU modificato;
- Dispositivi a blocchi virtuali (il file img stesso più eventuali altri aggiuntivi);

- Dispositivo virtuale contente il root filesystem;
- Mac address;

3.4.1 Backup

Xenbackup è uno script rilasciato sotto licenza GNU/Gpl scritto da John D. Quinn, che è stato successivamente modificato per adattarlo alle esigenze del sistema in esame.

Questo script si basa sul programma *rdiff-backup*, scritto in python, che permette di effettuare un backup di tipo incrementale di files e cartelle anche attraverso una rete (funzionalità indispensabile nel caso venga usato ad esempio un NAS).

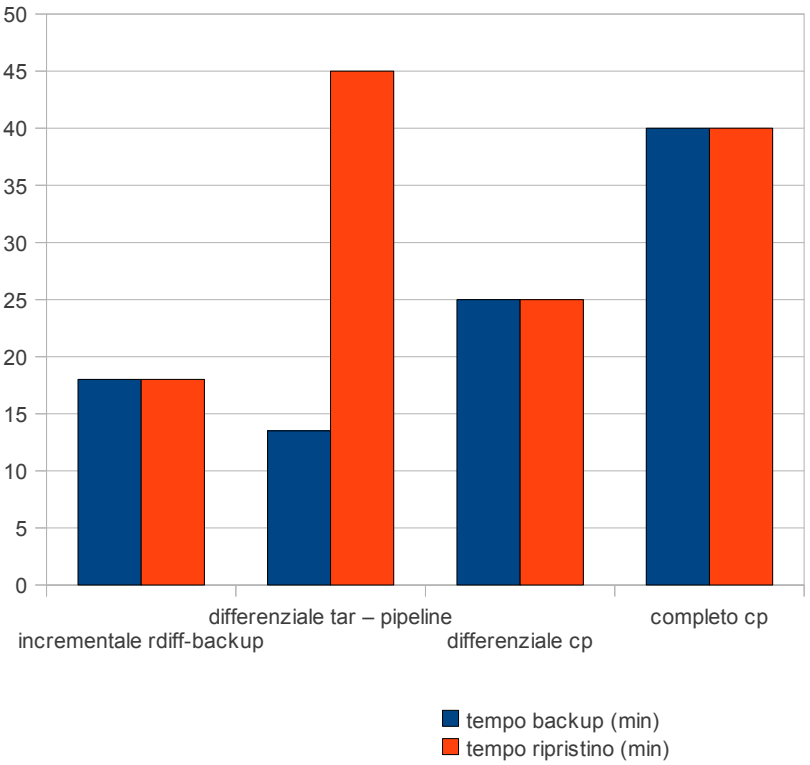
Si è scelto questa soluzione a seguito della valutazione dei test effettuati sulle diverse soluzioni di backup. I tipi di backup presi in considerazione sono stati:

- backup incrementale con rdiff-backup;
- backup completo con il comando *cp* messo a disposizione del sistema operativo del Dom0;
- backup differenziale del contenuto dei file IMG utilizzando sia il comando *cp* sia utilizzando il metodo *tar – pipeline*

I test sono stati effettuati sui file IMG di 5 VM per un totale di circa 40 GB di spazio disco:

Dai test è emerso che il tipo di backup più performante e anche il più semplice da implementare è appunto il backup incrementale dell'intera cartella contente i file IMG.

Al fine di rendere automatico il backup delle virtual machine è stato adoperato il processo di sistema *cron*, che permette di eseguire programmi e script periodicamente.
L'implementazione di questa funzionalità è molto semplice: innanzitutto è necessario modificare il file *crontab* (cartella */etc/*) inserendo data e ora in cui il backup deve entrare in funzione.



Ad esempio inserendo la riga

```
30 20 * * * sh xenBackup.sh
```

è possibile eseguire lo script ogni sera alle 20:30.

Infine basta spostare lo script xenBackup.sh all'interno di una delle tre cartelle associate a cron:

- /etc/cron.hourly/
- /etc/cron.daily/
- /etc/cron.weekly

Queste cartelle permettono di eseguire programmi e script su base oraria, giornaliera e settimanale rispettivamente.

3.4.2 Clonazione

CloneVm è un altro script molto utile il cui scopo è la clonazione (replica) di una VM esistente. Questo script si affida a tre utility:

la prima è *virsh*, un programma che, sfruttando la libreria *libvirt* (che funge da interfaccia con i servizi offerti dai demoni DM&C di xen), permette di gestire le virtual machine da riga di comando;
a seguire il programma *uuidgen*, che genera un nuovo identificatore uuid, quindi lo script m*acGen*, scritto in python, che si occupa di generare un nuovo indirizzo mac.

La clonazione di una VM avviene in modo piuttosto semplice:
il primo passo consiste nel creare una copia del file IMG con il nuovo nome (ricevuto in ingresso da riga di comando al momento dell'esecuzione dello script);
successivamente si passa all'estrazione, tramite virsh, del file XML di configurazione dal file IMG della macchina da clonare.

Nel passo successivo vengono modificati i valori all'interno del file XML, quali il nome della VM e il percorso dei dispositivi a blocchi virtuali con i parametri ricevuti in ingresso dalla riga di comando;
infine macGen e uuidgen si occupano di generare gli ultimi due parametri richiesti.

L'ultimo passo consiste nell'utilizzare il nuovo file di configurazione per creare e inizializzare il nuovo DomU, sempre tramite virsh.

Il tempo necessario per la clonazione dipende dalle caratteristiche della macchina fisica su cui viene eseguito lo script.

Nei test sono stati registrati tempi di circa 20 minuti per quanto riguarda l'esecuzione sulla workstation e tempi inferiori ai 10 minuti per l'esecuzione su server.

3.4.3 Ripristino

RestoreVM è un semplice script che ripristina il funzionamento di un DomU a seguito di un crash della virtual machine dovuto ad esempio ad un problema software.

Il presupposto per il corretto funzionamento dello stesso è quindi il file IMG integro, non corrotto. Si basa anch'esso sul programma virsh.

4. Conclusioni

Xen è un sistema di virtualizzazione molto performante grazie alla paravirtualizzazione ed è inoltre estremamente stabile e sicuro; grazie alle tecniche e agli strumenti di live migration e agli script realizzati durante lo studio è molto semplice gestire un elevato numero di macchine fisiche virtualizzate e le corrispondenti VM.

Tuttavia a meno di utilizzare distribuzioni Unix-based che lo implementano di default, il processo di installazione da codice sorgente risulta assai complicato e oneroso in termini di tempo.

Da considerare inoltre il fatto che la distribuzione Red Hat Linux Enterprise (una delle più utilizzate in ambito server), da cui deriva CentOS, non supporta più questo software di default, lasciando l'onere della gestione degli aggiornamenti all'amministratore di sistema o al tecnico addetto alla manutenzione.

In definitiva tenendo conto dei vantaggi e degli svantaggi del sistema di paravirtualizzazione Xen, nel caso non siano assolutamente necessarie prestazioni elevate, può essere sufficiente ricorrere ad altri software di virtualizzazione, decisamente meno performanti ma più semplici da implementare e mantenere.

5. Bibliografia

1. Riccardo Sbirrazzuoli. Virtualizzazione, concetti teorici. *Sviluppo di un sistema di management di ambienti di esecuzione virtualizzati per sistemi batch.* 2008

2. Xen *Architecture Overview*. 2008. http://www.xen.org/support/documentation.html

3. Aravind Menon, Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman, Willy Zwaenepoel. Performance overheads in Xen. *Diagnosing Performance Overheads in the Xen Virtual Machine Environment.* 2005

4. Davide Grossato. Vantaggi della virtualizzazione. *La Virtualizzazione*. 2010. http://www.slideshare.net/

5. Paolo Campegiani. Sistemi di virtualizzazione. *Virtualizzazione.* 2006. www.scuolaiad.it/

6. Andrew Levy . *Virtualization with Xen* 2009 www.csc.com

7. Red Hat. Xen DM&C e Api. docs.redhat.com

8. Riccardo Veraldi . *Xen e i benefici della virtualizzazione HVM*. 2007 http://www.infn.it/

6. Allegati

6.1 xenBackup.sh

```
#!/bin/bash
#
#   Copyright John Quinn, 2008
#
#   This program is free software: you can redistribute it
and/or modify
#   it under the terms of the GNU General Public License as
published by
#   the Free Software Foundation, either version 3 of the
License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be
useful,
#   but WITHOUT ANY WARRANTY; without even the implied
warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General
Public License
#   along with this program.  If not, see
<http://www.gnu.org/licenses/>.

#
# xenBackup - Backup Xen Domains
#
#             Version:    1.0:     Created:  John D Quinn,
http://www.johnandcailin.com/john
#             Version:    1.1:     Modified: Gian Piero
Marani

# initialize our variables
targetLocation="/var/backup"                    # the
default backup target directory
mountPoint="/mnt/xen"                    # the mount point
to use to mount disk areas
xenDiskArea="/var/lib/xen/images"        # the LVM volume
```

```
group to use
deviceMapper="/dev/mapper"       # the mount point of loop
devices
quiet="false"                              # keep the
chatter down
rdiffbackupExe=/usr/bin/rdiff-backup     # rdiff-backup
executable
xmExe=/usr/sbin/xm                       # xm executable
globalBackupResult=0                     # success status of
overall job

# settings for logging (syslog)
loggerArgs=""                            # what extra
arguments to the logger to use
loggerTag="xenBackup"                    # the tag for our
log statements
loggerFacility="local3"                  # the syslog
facility to log to



# function to print a usage message and bail
usageAndBail()
{
   cat << EOT
Usage: xenBackup -t [cartellaDidestinazione]

Backup xen domains to a target area

   -t      target LOCATION for the backup e.g. /tmp or
root@www.example.com:/tmp
           (not used for tar engine)
   -q      run in quiet mode, output still goes to syslog

Example 1
   Backup all imgfiles to the /tmp directgory
   $ xenBackup -t /tmp

Example 2
   Backup all imgfiles to directory /var/xenImages on
machine backupServer,
   $ xenBackup -t root@backupServer:/var/xenImages



EOT
```

```
   exit 1;
}

# parse the command line arguments
while getopts q:t:h o
do     case "$o" in
        q)     quiet="true";;
        t)     targetLocation="$OPTARG";;
        h)     usageAndBail;;
        [?])   usageAndBail
       esac
done

   if test $targetLocation == "null"
   then
      usageAndBail
   fi
# if quiet don't output logging to standard error
if test ${quiet} = "false"
then
   loggerArgs="-s"
fi

# setup logging subsystem. using syslog via logger
logCritical="logger -t ${loggerTag} ${loggerArgs} -p $
{loggerFacility}.crit"
logWarning="logger -t ${loggerTag} ${loggerArgs} -p $
{loggerFacility}.warning"
logDebug="logger -t ${loggerTag} ${loggerArgs} -p $
{loggerFacility}.debug"

# make sure only root can run our script
test $(id -u) = 0 || { ${logCritical} "This script must be
run as root"; exit 1; }

# make sure that the guest manager is available
test -x ${xmExe} || { ${logCritical} "xen guest manager ($
{xmExe}) not found"; exit 1; }

# make sure that the rdiff-backup program is available
test -x ${rdiffbackupExe} || { ${logCritical} "rdiff-backup
executable (${rdiffbackupExe}) not found"; exit 1; }
```

```
# set verbosity
   if test ${quiet} = "false"
   then
      verbosity="3"
   else
      verbosity="0"
   fi


#cartella di destinazione
targetSubDir=${targetLocation}/rdiff-backup-imgFiles.mirror

# make the targetSubDir if it doesn't already exist
mkdir ${targetSubDir} > /dev/null 2>&1


# backup using rdiff-backup to the target directory
${logDebug} "backing up ImgFile of all VMs to $
{targetSubDir} using rdiff-backup"
${rdiffbackupExe} --verbosity ${verbosity} ${xenDiskArea}/ $
{targetSubDir}

# make sure that the backup was successful
if test $? -ne 0
then
   ${logCritical} "FAILURE: error backing up Imgfile of all
VMs"
   globalBackupResult=1
else
   ${logDebug} "SUCCESS: Imgfile of all VMs backed up"
fi


if test ${globalBackupResult} -eq 0
then
   ${logDebug} "SUCCESS: backup of all domains completed
successfully"
else
   ${logCritical} "FAILURE: backup completed with some
failures"
fi
exit ${globalBackupResult}
```

6.2 cloneVM.sh

```
#!/bin/bash

# function to print a usage message and bail

usageAndBail()

{

   cat << EOT

Usage: cloneVm [percorsoVM] [nomeVM] [destinazione]
[nomenuovaVM]

Example

   clone the selected vm

   $ cloneVm  -t /percorsoVMdaClonare -n nomeVMdaClonare
-d /destinazione -m VMNuova

EOT

  exit 1;

}

# parse the command line arguments

while getopts t:n:d:m:h o

do     case "$o" in

        m)     nuovoNome="$OPTARG";;

        d)     destinazione="$OPTARG";;

        n)     nomeVm="$OPTARG";;

        t)     percorsoVm="$OPTARG";;

        h)     usageAndBail;;

        [?])   usageAndBail

       esac

done


# make sure only root can run this script

test $(id -u) = 0 || { echo "This script must be run as
root"; exit 1; }


# make sure that the guest manager is available

test -x ${xmExe} || { echo "xen guest manager (${xmExe}) not
found"; exit 1; }
```

```
test -f ${percorsoVm}/${nomeVm}.img || { echo "Vm (${nomeVm}) not found"; exit 1; }


#duplicazione file img che funge da hard disk
dd if=${percorsoVm}/${nomeVm}.img of=${destinazione}/${nuovoNome}.img bs=4k


#estrazione di un file formato xml contenente i dati della VM
virsh dumpxml ${nomeVm} > ${percorsoVm}/${nuovoNome}.xml


#modifica del file xml per permettere la creazione della nuova VM


#modifica campo <nome> e campo >percorso>
sed -i "s/${nomeVm}/${nuovoNome}/g" ${percorsoVm}/${nuovoNome}.xml
#cambio i valori dei campi campi uuid e mac address utilizzando l'utility "uuidgen" e lo script python "macGen.py"
macAddress=`python macGen.py`
uuid=`uuidgen`
sed -i "s/.*uuid.*/<uuid>$uuid<\/uuid>/g" ${percorsoVm}/${nuovoNome}.xml
sed -i "s/.*address.*/<mac address=\'$macAddress\'\/>/g" ${percorsoVm}/${nuovoNome}.xml


#definisco la nuova VM con le informazioni contenute nel file xml corrispondente
virsh define ${percorsoVm}/${nuovoNome}.xml
```

6.3 macGen.py

```
#! /usr/bin/python
# macGen.py script generates a MAC address for Xen guests
#
import random
mac = [ 0x00, 0x16, 0x3e,
random.randint(0x00, 0x7f),
random.randint(0x00, 0xff),
random.randint(0x00, 0xff) ]
print ':'.join(map(lambda x: "%02x" % x, mac))
```

6.4 restoreVm.sh

```
#!/bin/bash
#
# restoreVM
#             Version:    1.0:     Created:  Gian Piero
Marani

# initialize our variables
xenDiskArea="/var/lib/xen/images"        # the LVM volume
group to use
xmExe=/usr/sbin/xm                       # xm executable

# settings for logging (syslog)
loggerArgs=""                            # what extra
arguments to the logger to use
loggerTag="restoreVM"                  # the tag for our log
statements
loggerFacility="local3"                  # the syslog
facility to log to

# function to print a usage message and bail
usageAndBail()
{
   cat << EOT
Usage: xenBackup [OPTION]...
Backup xen domains to a target area. different backup
engines may be specified to
produce a tarfile, an exact mirror of the disk area or a
mirror with incremental backup.

   -n      restore the specified DOMAIN

   -h      this help

Example
   ripristino della nomeVM con file img situato in
/var/lib/xen/images
   $ restoreVm -n nomeVM


EOT
```

```
   exit 1;
}

# parse the command line arguments
while getopts n:h o
do     case "$o" in
        n)     nomeVM="$OPTARG";;
        h)     usageAndBail;;
        [?])   usageAndBail
       esac
done


# setup logging subsystem. using syslog via logger
logCritical="logger -t ${loggerTag} ${loggerArgs} -p $
{loggerFacility}.crit"
logWarning="logger -t ${loggerTag} ${loggerArgs} -p $
{loggerFacility}.warning"
logDebug="logger -t ${loggerTag} ${loggerArgs} -p $
{loggerFacility}.debug"

# make sure only root can run our script
test $(id -u) = 0 || { ${logCritical} "This script must be
run as root"; exit 1; }

# make sure that the guest manager is available
test -x ${xmExe} || { ${logCritical} "xen guest manager ($
{xmExe}) not found"; exit 1; }


   # make sure we've got the name of the VM
   if test $nomeVM == "null"
   then
      usageAndBail
   fi

   # unmount mount point if already mounted
   umount ${mountPoint} > /dev/null 2>&1

   # check if the imgfile exist
   xenDisk=${xenDiskArea}/${nomeVM}.img
    test -r ${xenDisk} || { ${logCritical} "xen disk area
```

```
not readable. are you sure that the domain \"${nomeVM}\"
exists?"; exit 1; }


   #estrazione di un file formato xml contenente i dati
della VM
   virsh dumpxml ${nomeVm} > ${xenDiskArea}/${nomeVM}.xml
   #ripristino la vm
   virsh define ${xenDiskArea}/${nomeVM}.xml
```

6.5 dumpXenstore.sh

```
#!/bin/sh
#script che visualizza le informazioni di tutti i domains
function dumpkey() {
   local param=${1}
   local key
   local result
   result=$(xenstore-list ${param})
   if [ "${result}" != "" ] ; then
      for key in ${result} ; do dumpkey ${param}/${key} ;
done
     else
      echo -n ${param}'='
      xenstore-read ${param}
   fi
}

for key in /vm /local/domain /tool ; do dumpkey ${key} ;
done
```