UNIVERSITA' DEGLI STUDI DI CAMERINO

Scuola di Scienze e Tecnologie

Corso di laurea in Informatica

La virtualizzazione e Proxmox

Laureando:

DENNY SCAGNETTI

Relatore:

FAUSTO MARCANTONI

Correlatore:

FRANCESCO MARIA COMPAGNUCCI

Anno Accademico 2015/2016

QUESTA TESI LA DEDICO A TUTTA LA MIA FAMIGLIA E A COLORO CHE MI HANNO SUPPORTATO FINO AD ORA

Indice

Introduzione

Il significato del termine italiano virtuale indica un evento che esiste potenzialmente ma non si è ancora verificato, ad esempio in una gara si ha un vincitore virtuale; lo stesso termine nell'ambito dell'ottica indica qualcosa di fittizio, non reale, immagine virtuale; nella fisica rappresenta qualcosa che si potrebbe effettuare, lavoro virtuale; nell'ambito burocratico invece indica la condizione in cui il pagamento di un tributo o simile avvenga direttamente e non tramite una marca da bollo, imposta virtuale e così via nei vari contesti. Nell'ambito puramente informatico il termine virtuale dalla definizione sta ad indicare "la creazione di una versione virtuale di una risorsa normalmente fornita fisicamente". In pratica, definisce un livello di astrazione che si va a sovrapporre all'hardware fisico di un elaboratore, permettendo in questo modo di poter installare diversi sistemi operativi sulla stessa macchina. Ciò viene reso possibile grazie alla creazione di *virtual machine,* che sono ambienti appositamente creati dal sistema di virtualizzazione e dispongono di un set di risorse hardware virtuali (RAM, CPU, HD, NIC, etc.). Su di esse è possibile installare un sistema operativo ed eseguire delle applicazioni in modo indipendente, rispetto all'hardware fisico sottostante, aumentando in tal modo l'utilizzo e la flessibilità delle risorse hardware disponibili.

La virtualizzazione comunque è un concetto già presente nell'architettura dei sistemi operativi, infatti esiste la memoria virtuale che consiste nel simulare uno

spazio di memoria centrale maggiore di quello fisicamente presente, grazie all'utilizzo di spazi di memoria secondaria su altri dispositivi, oppure esiste la condizione in cui ogni processo caricato nel sistema, virtualmente, utilizza tutta la potenza di calcolo quando invece, tale potenza di calcolo è ripartita tra tutti i processi secondo gli algoritmi di scheduling del sistema operativo in uso.

In questa tesi ci occuperemo della virtualizzazione intesa come livello di astrazione dell'hardware e analizzeremo i vantaggi che una tale soluzione comporta.

Il principale vantaggio che la virtualizzazione comporta è che le virtual machine con essa create, possono essere implementate in modo tale da garantire che siano isolate dalla macchina fisica; andando così a creare un box in cui sarebbe possibile fare test di debugging o ad esempio testare la sicurezza di un sistema, riproducendone virtualmente le caratteristiche. Viene poi fornita una sorta di incapsulamento, in quanto tutta la virtual machine sarà contenuta in pochi file facili da spostare e da copiare semplificando in tal modo le procedure di backup e *disaster recovery*, in quanto avendo una copia giornaliera delle immagini virtualizzate delle risorse[1], sarà possibile un ripristino rapidissimo in caso di malfunzionamenti dovuti ai più disparati motivi. Inoltre con la virtualizzazione si ottiene il partizionamento delle risorse disponibili (un po' come succede per gli hard disk), realizzando una sorta di mascheramento di tali risorse agli utilizzatori, con l'obiettivo di evitare all'utente di dover comprendere e gestire dettagli complessi delle risorse e di aumentare, nel contempo, la condivisione e l'utilizzo delle risorse, garantendo la possibilità di una successiva espansione.

Esistono poi sistemi di virtualizzazione che permettono la migrazione a caldo di una virtual machine detta anche *live migration*, con la gestione dei *failover* del tutto trasparente all'utente. In pratica, nell'eventualità di un guasto hardware, la virtual machine sarà automaticamente migrata su un altro server in servizio senza alcuna interruzione operativa.

La virtualizzazione di una risorsa, hardware o software, consente di realizzare una copia identica per funzionalità alla risorsa reale. Unità ottiche, spazio di

archiviazione, sistemi operativi, divisione di un hard disk in unità logiche, costituiscono tutti esempi di risorse virtualizzabili.

Il concetto di virtual machine non è comunque una novità, infatti questa esigenza di poter sfruttare una macchina potente era già utilizzato nel 1967, quando ci fu la prima implementazione del CP-40 su IBM System 360, un sistema operativo basato su memoria virtuale e *time sharing*, che consentiva di eseguire tante istanze software del Sistema Operativo sullo stesso hardware con lo scopo di partizionare la potenzialità dei grandi mainframe.

Nel corso degli anni l'introduzione di Minicomputer e Personal Computer, fornì una via più accessibile ed efficiente per distribuire la potenza di calcolo, che portò intorno agli anni '80 all'accantonamento del concetto di virtualizzazione.

Negli anni '90 invece si assiste alla reviviscenza di questa tecnologia, poiché risultò chiaro che sarebbe potuta essere stata d'aiuto per la risoluzione di tutte quelle problematiche relative alla proliferazione di hardware meno costosi tipo: il sottoutilizzo delle risorse, i costi di gestione e la manutenzione elevati e la vulnerabilità sempre più alta.

Struttura della tesi

Premettiamo che quando possibile, nei capitoli seguenti, si è cercato di utilizzare le voci di Wikipedia come un riferimento primario per la maggior parte dei termini e delle tecnologie trattate, perché in molti casi fornisce una valida descrizione, per lo più imparziale, che non promuove alcuna soluzione tecnica unica per una data tecnologia.

Nel capitolo 1 - **Un po' di teoria** - vengono citati tutti i riferimenti teorici alla base della virtualizzazione, partendo dai teoremi fondamentali di Popek e Goldberg, per poi passare alle definizioni di Hypervisor, Virtual Machine ed illustrare i tipi di virtualizzazione possibili. Inoltre vengono illustrate le tipologie di storage utilizzabili nell'ambito della virtualizzazione. Infine vengono analizzati i pro e i contro che una soluzione virtualizzata comporta.

Nel capitolo 2 – **Proxmox VE** – viene visualizzato in tutte le sue sfaccettature questo sistema di virtualizzazione, mostrando e spiegando abbastanza approfonditamente le sue caratteristiche, le sue peculiarità e l'interfaccia grafica.

Nel capitolo 3 – **La pratica** - viene presentato il nostro progetto, illustrandone la struttura ed il funzionamento, per poi passare alla presentazione dell'idea che dovrebbe permettere, grazie all'utilizzo di appositi script e della virtualizzazione, un risparmio energetico ed economico, illustrando nei vari paragrafi una serie di prodotti e di step che hanno fatto parte della realizzazione del progetto finale.

Nel capitolo 4 - **Conclusione** - si conclude il lavoro riassumendo generalmente le fasi del progetto, e si fa una valutazione sui risultati ottenuti e citando sia gli aspetti positivi che negativi.

Capitolo 1

Un po' di teoria

1.1 Popek e Goldberg

I primi, che riuscirono a formulare delle tesi importanti sul concetto di virtualizzazione, furono Popek e Golberg [1], riuscendo a definire le linee guida per la virtualizzazione ed impostando le condizioni necessarie e sufficienti che una piattaforma deve soddisfare affinché possa essere correttamente virtualizzata.
Tali parametri sono stati introdotti nel 1974 appunto da Gerald J. Popek e Robert P. Goldberg[1] e ancora oggi rimangono validi rappresentando le direttive che i programmatori devono seguire durante la progettazione di architetture in grado di supportare la virtualizzazione in maniera efficiente.
In definitiva, seguendo i loro presupposti, possiamo parlare di virtualizzazione se e solo se vengono rispettate le seguenti tre caratteristiche fondamentali:

[1] Gerald J. Popek ricercatore dell'University of California, Los Angeles e Robert P. Goldberg Honeywell ricercatore dell'Harvard University

- **Equivalenza**: Il sistema che viene eseguito nell'ambiente virtuale si deve comportare esattamente come se fosse eseguito su una macchina fisica equivalente;

- **Controllo delle risorse**: L'ambiente di virtualizzazione deve avere il completo controllo delle risorse virtualizzate;

- **Efficienza**: Molte delle istruzioni della piattaforma virtualizzata devono essere eseguibili senza l'intervento dell'ambiente di virtualizzazione.

La realizzazione a livello software della virtualizzazione delle risorse è affidata ad una categoria di applicazioni che va sotto il nome di Virtual Machine Monitor (VMM), che rappresenta uno strato di software che si interpone tra l'hardware fisico e le virtual machine rispettando le tre caratteristiche fondamentali viste prima.
Per definire i teoremi di Popek e Goldberg viene introdotta una classificazione in tre gruppi delle *instruction set architecture* (ISA) del processore:

- **Istruzioni privilegiate**: L'esecuzione delle istruzioni privilegiate da parte di una virtual machine, causano un'eccezione che viene catturata (*trap*) dalla VMM che emula, ricorrendo ai servizi del sistema operativo *host*;

- **Istruzioni sensibili al controllo**: Queste istruzioni tentano di modificare la configurazione delle risorse del sistema;

- **Istruzioni sensibili al comportamento**: sono istruzioni il cui risultato dipende dalla configurazione delle risorse (modalità di funzionamento del processore o contenuti dei registri)

Enunciata tale distinzione tra le categorie di istruzioni del processore, risulta possibile ora enunciare i due teoremi.

> **Teorema 1**: «*Per ogni computer convenzionale, di terza generazione*[2]*, può essere realizzato un VMM in grado di emularlo se i set di istruzioni sensibili al controllo e al comportamento sono sottoinsiemi delle istruzioni privilegiate*.»

In altri termini, tutte le istruzioni privilegiate, sono catturate perché eseguite dalla VMM in un contesto non privilegiato mentre le istruzioni non privilegiate possono essere eseguite direttamente senza l'intervento della VMM, coerentemente al principio dell'efficienza. Questa caratteristica consente di realizzare una virtualizzazione ricorsiva, utile a definire le condizioni in cui un VMM può essere eseguito in una copia di se stesso, come meglio specificato nel secondo teorema:

> **Teorema 2**: «*Un sistema di terza generazione è virtualizzabile ricorsivamente se:*
>
> 1. *è virtualizzabile*;
> 2. *è possibile costruire per essa una macchina virtuale che non abbia dipendenze di temporizzazione*.»

In sostanza, se una virtual machine può eseguire una copia del VMM, allora la macchina originale è ricorsivamente virtualizzabile e l'unico limite al numero di VMM nidificate è rappresentata solo dalla quantità di memoria disponibile nel sistema.

Esistono poche architetture di terza generazione che risultano essere pienamente

[2] è una macchina che ha un processore convenzionale con due modalità di funzionamento: supervisor e user. In modalità supervisor è disponibile l'intero set di istruzioni, mentre in modalità user no.

virtualizzabili, quindi per far si che vengano comprese anche quelle architetture che non rispettano pienamente le caratteristiche viste prima, vengono ampliate quelle definizioni in modo da ottenere una forma più generale anche se meno efficiente. Parliamo di *Hybrid Virtual Machine* (HVM), la sua struttura è molto simile al sistema del VMM visto prima, ma in questo caso ci sono più istruzioni interpretate di quelle eseguite direttamente, questo comporta un'efficienza minore. Da questa nuova definizione deriva un ulteriore teorema, che altro non è che una estensione del teorema 1:

> **Teorema 3:** «*Un monitor HVM può essere costruito per qualsiasi macchina convenzionale di terza generazione, in cui l'insieme di istruzioni sensibili sono un sottoinsieme del set di istruzioni privilegiate*.»

1.2 Hypervisor

Dalla definizione di Popek e Goldberg una virtual machine è rappresentata da:
"un efficiente, ed isolato duplicato di una macchina reale", definizione che tiene fede ancora oggi alla condizione che ogni utente di una virtual machine, può installare un diverso sistema operativo ed avere l'impressione di essere l'unico utilizzatore dell'intero sistema. Tale tecnica utilizzata per permettere la condivisione tra più utenti di un'unica macchina, viene realizzata grazie ad un software chiamato *hypervisor* o *virtual machine monitor*.
Un hypervisor [2] realizza un livello di astrazione tra l'hardware reale dell'elaboratore, su cui funziona, e l'ambiente virtuale, in modo da far credere alle varie virtual machine in esecuzione, di aver a disposizione tutte le risorse

hardware, controllando e gestendo questi accessi da parte delle virtual machine in maniera celata alle stesse.

L'hypervisor realizza queste operazioni in diversi modi a seconda del sistema di virtualizzazione e si occupa di risolvere le eventuali eccezioni che avvengono quando una virtual machine prova ad eseguire un'istruzione privilegiata, mantenendo una discreta leggerezza di esecuzione, per non gravare sull'efficienza delle virtual machine. Possiamo quindi definire una virtual machine come un ambiente creato dall'hypervisor. Esso inoltre svolge attività di controllo al di sopra di ogni sistema permettendone lo sfruttamento delle attività dei sistemi operativi e delle applicazioni in modo da scoprire eventuali malfunzionamenti e dare la possibilità di intervenire celermente.
Esistono fondamentalmente due tipi di hypervisor:

- **Tipo 1**: detti nativi (o *bare metal*) (vedi fig.1.1), vengono eseguiti direttamente
  utilizzando le risorse hardware della macchina host e non si appoggiano a software sottostante. Soluzione adottata da Xen e da Proxmox;

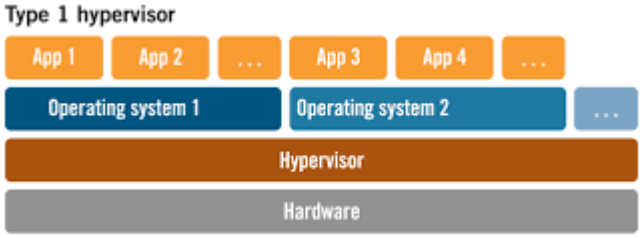

Figure 1.1 Type 1 or bare-metal hypervisor sits directly on the host hardware.

- **Tipo 2**: o *hosted* (vedi fig.1.2), sono applicazioni eseguite sempre sulla macchina host, ma in modo tale che software di virtualizzazione, hypervisor e sistema operativo host, siano eseguiti su tre livelli diversi. Questo tipo di hypervisor è quello implementato fra l'altro dalle soluzioni adottate da: VMWare Workstation, Microsoft Virtual PC e VirtualBox.

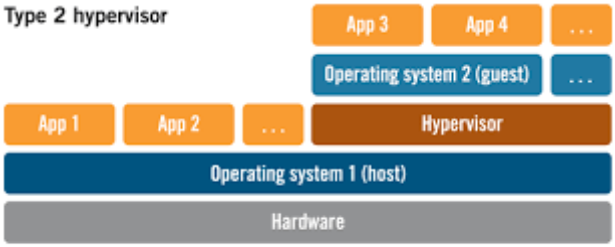

Figure 1.2 Type 2 hypervisor runs as an application on a host operating system.

1.2.1 Hypervisor Tipo 1 e 2 a confronto

L'hypervisor di tipo 1 [3] fornisce migliori prestazioni e maggiore flessibilità poiché opera come un sottile strato atto ad esporre alla virtual machine le risorse hardware, riducendo le richieste non necessarie per eseguire l'hypervisor stesso.

I server che eseguono il Tipo 1 sono spesso server *single-purpose* che non offrono nessun'altra funzione. Essi diventano parte dell'insieme delle risorse e sono designati specificatamente per supportare l'operazione di più applicazioni in varie virtual machine.

Tipicamente, il tipo 1 è più efficiente del tipo 2 sotto molti aspetti, pur fornendo quasi completamente le stesse funzionalità con la stessa virtual machine.

Per sua natura, l'hypervisor di tipo 1 esegue virtual machine che offrono servizi agli utenti, il tipo 2 invece offre una serie di servizi che raramente vengono utilizzati da un provider.

In genere, un hypervisor di tipo 1 come per esempio VMware ESX è dislocato nei server posti nei *datacenter*. Viene usato, per esempio, quando un'azienda per creare numerose applicazioni aziendali client-server o servizi IT nello stesso server fisico. Questo permetterebbe di gestire le risorse in maniera ottimale senza usare un server per ogni applicazione o servizio.

D'altro canto, un hypervisor di tipo 2 [4] è definito "hosted" poiché viene

installato ed eseguito al di sopra del sistema operativo host esistente, ed ogni istanza *guest* o virtual machine viene eseguita sotto l'hypervisor stesso. Sebbene l'aggiunta di uno strato di sistema operativo host può potenzialmente limitare le performance e far emergere possibili problemi di sicurezza, la convenienza di un sistema operativo conosciuto può facilitare la configurazione e la gestione dei task che possono essere particolarmente utili per gli utenti finali.

L' hypervisor di tipo 2 è perciò utilizzato dai dispositivi degli utenti finali come ad esempio laptop, workstation o PC. Questi sistemi, di solito, non richiedono istanze di virtual machine molto performanti, anzi ne vengono create un numero esiguo per verifiche di compatibilità o per test funzionali. Per esempio, uno sviluppatore software potrebbe creare una virtual machine Linux che poggia su un hypervisor di tipo 2 nel suo PC Windows per testare il proprio software su una struttura basata su Linux.

1.3 Approfondimento sulle VM

Iniziamo con il dividere le virtual machine in due categorie: le system virtual machine, che forniscono una virtualizzazione completa di una piattaforma e supportano l'esecuzione di un sistema operativo e le *process virtual machine*, realizzate per eseguire un solo programma, ovvero che supportano l'esecuzione di un singolo processo. Una caratteristica essenziale delle virtual machine è che il software in esecuzione su di esse è limitato a sfruttare le risorse virtuali che gli

vengono fornite.

Le system virtual machine sono a volte anche chiamate hardware virtual machine e consentono la condivisione delle risorse fisiche del sistema tra diverse virtual machine, ciascuna con il proprio sistema operativo.

I principali vantaggi che si ottengono con queste macchine sono: la possibilità di poter eseguire diversi sistemi operativi su un'unica macchina, garantendo comunque l'isolamento tra i diversi sistemi, la possibilità di eseguire un instruction set architecture diverso da quello che fornisce l'hardware reale ed infine semplificano la distribuzione di applicazioni, la manutenzione, il *disaster recovery* e l'*high availability*.

L'esecuzione di numerose virtual machine sullo stesso server, ciascuna con il proprio sistema operativo, è una pratica spesso usata nel *server consolidation* che ha proprio come obiettivo quello di sfruttare al meglio le risorse dei server che si hanno a disposizione, riducendo al minimo il numero delle macchine fisiche, combattendo il problema del *server sprawl*, che si presenta quando si hanno numerosi server che occupano molto spazio e consumano molte più risorse di quanto dovrebbero rispetto al carico di lavoro cui sono sottoposti.

La maggior parte di questa tipologia di apparati subisce un carico di lavoro

del 15-20% della loro capacità, il che evidentemente costituisce un paradosso dal punto di vista economico. Servizi che solitamente sono eseguiti su server diversi, in quest'ottica sono eseguiti su VM diverse ma sullo stesso server, pratica questa che è solitamente definita come *quality-of-service isolation* (*QoS isolation*). La pratica di passaggio da numerosi piccoli server ad un solo server che virtualizza l'esecuzione degli altri è detta *Physical to virtual* o *P2V transformation*. Questa tecnica viene anche utilizzata nell'analisi forense e consente la creazione di una copia esatta e funzionante di un calcolatore, comprese le partizioni nascoste o criptate, e il tutto avviene senza alterare in alcun modo i dati.

Le process virtual machine sono virtual machine eseguite come se fossero normali applicazioni all'interno di un sistema operativo. Esse vengono create quando il processo è avviato e distrutte quando viene terminato. Il loro scopo è quello di

fornire un ambiente di programmazione che crei un livello di astrazione dalle risorse hardware a disposizione del sistema operativo e che consenta pertanto di eseguire il programma su qualsiasi piattaforma allo stesso modo.

Una process virtual machine deve fornire un livello di astrazione concettualmente simile a quello dei linguaggi di programmazione ad alto livello, non deve invece preoccuparsi di come sono implementate a livello hardware le sue istruzioni, a differenza delle system virtual machine, i cui set di istruzioni sono a basso livello.

Ogni process virtual machine si appoggia su un interprete e sulla compilazione *just in time* per la traduzione delle istruzioni.

Un esempio molto diffuso di questo tipo di macchine è rappresentato dalla Java Virtual Machine o JVM (vedi fig.1.3) , la quale esegue i programmi scritti in *bytecode*[3] e presenta la caratteristica di disporre di implementazioni della virtual machine Java per diversi ambienti operativi, che rappresenta la chiave della portabilità di Java, proclamata nello slogan "*write once, run everywhere*" ("scrivi una volta, esegui dappertutto"). La virtual machine realizza infatti un ambiente di esecuzione omogeneo, che nasconde al software Java (e quindi al programmatore) qualsiasi specificità del sistema operativo sottostante.

Un altro esempio è quello del .NET Framework di Microsoft, la cui virtual machine si chiama *Common Language Runtime*, altri tipi di process virtual machine sono quelle che consentono di astrarre dai meccanismi di comunicazione dei computer *cluster*[4], tali meccanismi di comunicazione che di conseguenza spesso risultano essere eterogenei. In tal caso esiste una virtual machine per ciascun processo consentendo al programmatore di occuparsi degli algoritmi di condivisione delle risorse e distribuzione più che sui meccanismi di comunicazione.

[3] risultato della compilazione di codici sorgenti scritti in linguaggio java

[4] gruppo di computer collegati che lavorano insieme, condividendo capacità di calcolo ed altre risorse

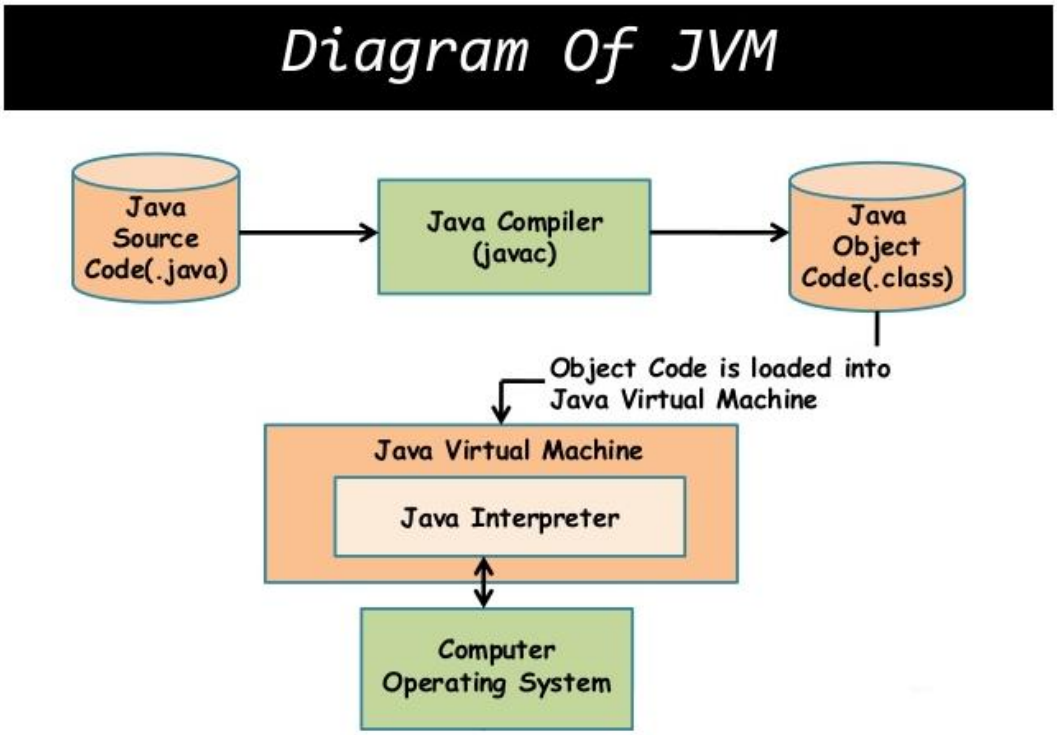
Diagram Of JVM
Java Source Code(.java)
Java Compiler (javac)
Java Object Code(.class)
Object Code is loaded into Java Virtual Machine
Java Virtual Machine
Java Interpreter
Computer Operating System

fig 1.3

1.4 Storage per macchine virtuali

In un sistema virtualizzato non solo a livello “workstation”, ma a livello server (come ad esempio Proxmox o Xen), la componente fondamentale dopo i server fisici che permettono alle VM di girare è lo storage.
Con il termine storage si indicano i dispositivi hardware per l’immagazzinamento in modo non volatile di dati elettronici.
In particolare, nell’ambito della virtualizzazione e delle macchine virtuali, si parla sempre più spesso di network storage, ovvero qualunque tipo di storage che coinvolge l’accesso alle informazioni tramite una rete di computer. Ciò dà la possibilità a multipli nodi di condividere lo storage nello stesso tempo. Condividendo lo storage, la gestione delle informazioni è centralizzata, riducendone la duplicazione ed evitando quindi ridondanze inutili e spesso dannose.

1.4.1 Storage unico condiviso

E’ un modello di network storage dove tutti i dati sono mantenuti in uno stesso posto. Tale nodo del sistema deve quindi essere affidabile, scalabile, veloce e sicuro. Sarà quindi composto da array RAID, dispositivi di backup ed interfacce veloci e ridondanti.
I due sottomodelli più utilizzati in sistemi a storage unico condiviso sono sicuramente:

- **SAN** (Storage Area Network): sistema estremamente veloce, ma che

sfrutta hardware e tecnologie costose;

- **NAS** (Network Attached Storage): sistema più lento, ma più economico e semplice da mettere in opera, gestire ed integrare; è basato su standard universalmente riconosciuti e tecnologie/hardware non costosi.

1.4.1.1 Storage Area Network

Una Storage Area Network (SAN) [5], mostrata in fig. 1.4 è una rete ad alta velocità composta da dispositivi di memorizzazione di massa condivisi. I protocolli attualmente più diffusi per l'utilizzo degli storage via rete sono FC (Fibre Channel) ed iSCSI (Internet SCSI).
Esiste una definizione formale di rete SAN sul dizionario tecnico pubblicato dalla "SNIA" (Storage Networking Industry Association) che la definisce nei seguenti termini:
"Una rete il cui scopo principale è il trasferimento di dati tra sistemi di computer ed elementi di storage e tra elementi di storage. Una rete SAN consiste in un'infrastruttura di comunicazione, che fornisce connessioni fisiche e in un livello di gestione che organizza connessioni, elementi di storage e sistemi di computer in modo da garantire un trasferimento di dati sicuro e robusto."

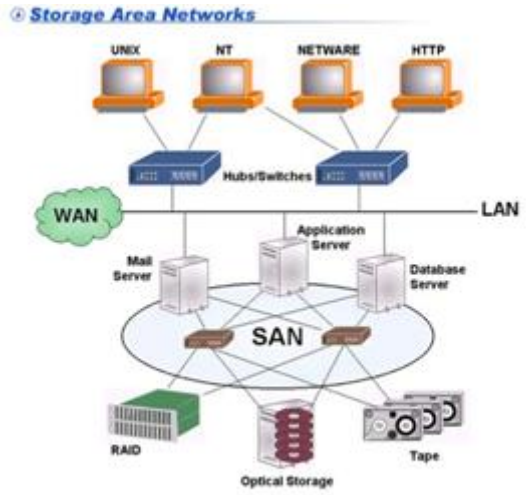

fig. 1.4

Un'architettura SAN lavora in modo che tutti i dispositivi di memorizzazione siano disponibili a qualsiasi server della rete LAN o MAN di cui la SAN fa parte; una SAN può essere anche condivisa fra più reti interconnesse, anche di natura diversa: in tal caso uno dei server locali agisce come "bridge" fra i dati memorizzati e gli utenti finali. Il vantaggio di un'architettura di questo tipo è che tutta la potenza di calcolo dei server è utilizzata per le applicazioni, in quanto i dati non risiedono direttamente su alcuno di tali server.

Vantaggi

Le aziende devono poter accedere ai dati in modo rapido e sicuro e quindi la filosofia dell'architettura SAN è quella di poter integrare tutte le caratteristiche dei tradizionali sistemi di memorizzazione:

- Alte prestazioni
- Alta disponibilità
- Scalabilità
- Facilità di gestione

Tutto ciò integrato con le caratteristiche di connettività ed accesso distribuito del network computing, attraverso un'architettura di rete dedicata alla gestione ed

archiviazione dei dati, in grado di non sovraccaricare i server nelle operazioni di scrittura e lettura degli stessi, da e verso lo storage. Le reti SAN forniscono una serie di indubbi vantaggi rispetto ai dispositivi di storage connessi direttamente ai server (detti "*Direct Attached Storage*"). Offrono una connettività *any-to-any* tra server e dispositivi di storage, aprendo in tal modo la strada al trasferimento diretto di dati tra periferiche di memorizzazione, con conseguenti miglioramenti dell'efficienza nello spostamento di dati e processi, quali il backup o la replica dei dati.

Applicazioni

L'impiego delle attuali tecnologie di networking proposte per le reti SAN consente di:

- Raggiungere distanza di connettività superiori e prestazioni migliori rispetto a quanto non sia possibile con l'attuale tecnologia SCSI;
- Facilitare il compito di centralizzare la gestione dello storage;
- Consolidare lo storage ed il clustering dei sistemi presenti;
- Condividere i dati tra piattaforme eterogenee.

Garantendo alte prestazioni ed un accesso diretto ai dischi, le SAN faciliteranno lo spiegamento di database centralizzati e di applicazioni enterprise, facilitandone il backup ed eventuali disaster recovery.

1.4.1.2 Network Attached Storage

Un Network Attached Storage (NAS) [6], mostrato in fig. 1.5, è un dispositivo collegato ad una rete di computer la cui funzione è quella di condividere tra gli utilizzatori della rete un'area di storage. I NAS, generalmente, sono dei semplici

computer con a bordo il minimo necessario per poter utilizzare le funzionalità di networking.

Attualmente i più diffusi sono macchine con sistema operativo GNU/Linux (trasparente all'utente finale) contenenti numerosi hard disk per l'immagazzinamento dati. Tale architettura ha il vantaggio di rendere disponibili i file contemporaneamente su piattaforme eterogenee, in quanto il sistema operativo implementa i server di rete per tutti i protocolli più diffusi (FTP, NFS, SMB, ecc ecc.). Questi apparati non vanno confusi con le SAN, soluzioni di storage ben differenti: come visto nella sezione precedente, tali sistemi comprendono una rete e sono assoggettati a protocolli spesso proprietari e tecnologie molto più costose. Piuttosto, un sistema NAS può essere utilizzato come nodo di una SAN, data la scalabilità di tale architettura.

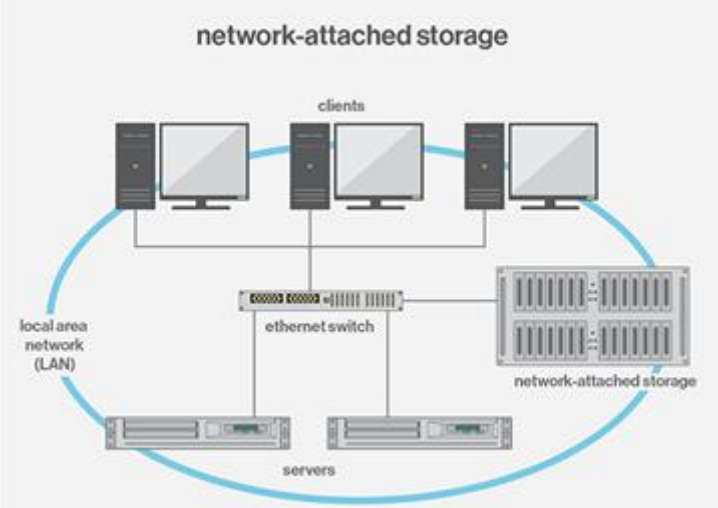

fig. 1.5

Vantaggi e Svantaggi

Il vantaggio dei NAS, oltre a centralizzare l'immagazzinamento dei dati in un solo posto invece che spargerli su diversi sistemi di una rete, è che sono unità altamente specializzate dal punto di vista prestazionale e della sicurezza dei dati:

gran parte di tali sistemi può implementare array RAID al fine di garantire la ridondanza dei dati e permettendo inoltre l'aggiunta e la rimozione di dischi rigidi in runtime senza bloccare il funzionamento dell'intero sistema ("*hot swap*").
Nell'ambito di questa architettura, lo svantaggio maggiore è costituito principalmente dall'enorme quantità di dati che transita sulla rete e dai limiti prestazionali e di stabilità dei file system di rete disponibili sul mercato.

1.4.2 Storage Distribuito

Un *filesystem* distribuito (o più semplicemente DFS) è una tipologia di "network filesystem" che permette la memorizzazione di file e risorse in dispositivi di archiviazione distribuiti in una rete informatica. A differenza di un comune file system locale, i dati non vengono letti o archiviati su di un dispositivo locale, ma, attraverso un meccanismo client-server, su dispositivi remoti, collegati in maniera trasparente alla propria gerarchia di file. Ogni nodo del sistema ha accesso diretto solo ad una parte dell'intero filesystem, a differenza delle soluzioni "storage unico condiviso" in cui tutti i nodi hanno accesso diretto all'intero sistema di storage.
Generalmente i DFS includono strutture per garantire la replicazione trasparente dei dati e meccanismi di tolleranza ai guasti. Il che significa che se un numero limitato di nodi va offline, il sistema continua a lavorare senza perdita di dati.
Un *distributed data store* è una rete in cui l'utente archivia i propri dati su un insieme di nodi della rete. Normalmente l'utente stesso rende disponibile il suo computer per archiviare i dati di altri utenti agendo da nodo attivo a sua volta.
Il tipico esempio di una rete di questo tipo è la rete "BitTorrent", in cui il nodo origine può andare offline senza interrompere la distribuzione dei dati condivisi. Questo però è il caso semplice in cui i file sono un numero limitato.
In una rete come "Freenet" tutti i nodi rendono disponibili tutti i file.

1.5 Tipi di virtualizzazione

Prima di approfondire il discorso relativo ai tipi di virtualizzazione bisogna fare una piccola premessa, l'architettura IA-32, più comunemente conosciuta come x86, non è nativamente compatibile con quelli che sono i criteri di virtualizzazione di Popek e Goldberg visti precedentemente e per ovviare a queste mancanze, esistono sia soluzione software (che affronteremo subito), che soluzioni hardware che saranno esaminate successivamente.

Il problema della virtualizzazione è stato affrontato in diversi modi e dal lato software può essere realizzato secondo quattro metodologie differenti: l'*emulation*, la *full virtualization*, la *paravirtualization* e la *operating system level virtualization*.

Tutte queste metodologie danno l'impressione all'utilizzatore di disporre di un sistema operativo stand alone, non virtualizzato.

1.5.1 Emulation

Con l'emulazione [7] (fig. 1.6), l'hypervisor simula l'intero hardware set, presentando un ambiente non necessariamente uguale alle risorse disponibili. Ciò permette al sistema operativo guest di essere eseguito ed utilizzato senza alcun tipo di modifica.

I limiti, in termini di presentazioni e di funzionalità di questo tipo, sono facilmente immaginabili. Infatti gli emulatori presentano al sistema operativo guest un'architettura hardware standard, precludendo quelle che potrebbero essere le funzionalità alle quali siamo abituati, ad esempio quelle implementate in hardware. Inoltre deve interfacciare la CPU, la memoria e l'I/O tra l'host e il guest. Per quanto riguarda la memoria, viene generalmente assegnate alla macchina virtuale una sequenza di indirizzi di memoria che utilizza in modo esclusivo e l'emulatore somma all'indirizzo della cella richiesta una costante. Un modo di gestire la CPU invece, consiste nell'interpretare il flusso di istruzioni provenienti dal sistema guest ed eseguire sul processore *host* una serie di istruzioni semanticamente equivalenti. Questo può essere resi possibile con l'assegnazione di una variabile per ogni registro e *flag* della CPU simulata.

Questo carico di lavoro dell'emulatore rende difficoltoso l'uso di questa tecnica di virtualizzazione quando bisogna emulare sistemi guest che richiedono processori di velocità equivalente al processore dell'host.

Esempi di software che utilizzano l' Emulation sono: Microsoft Virtual PC, QEMU

fig. 1.6

1.5.2 Full Virtualization

La Full Virtualization (fig. 1.7) detta anche Native Virtualization, è molto simile alla Emulation. Come nell'emulazione, i sistemi operativi guest girano senza alcuna modifica in macchine virtuali ospitate sul sistema host. La differenza sostanziale rispetto all'emulazione sta nel fatto che i sistemi operativi guest devono essere compatibili con l'architettura hardware della macchina fisica. In questo modo, molte istruzioni possono essere eseguite direttamente sull'hardware senza bisogno di un software di traduzione garantendo prestazioni superiori rispetto all'emulazione.

Esempi di software che utilizzano la Full Virtualization sono: VMware, Parallel Desktop, Virtual Box, Proxmox e Xen, limitatamente ai sistemi operativi proprietari non modificabili.

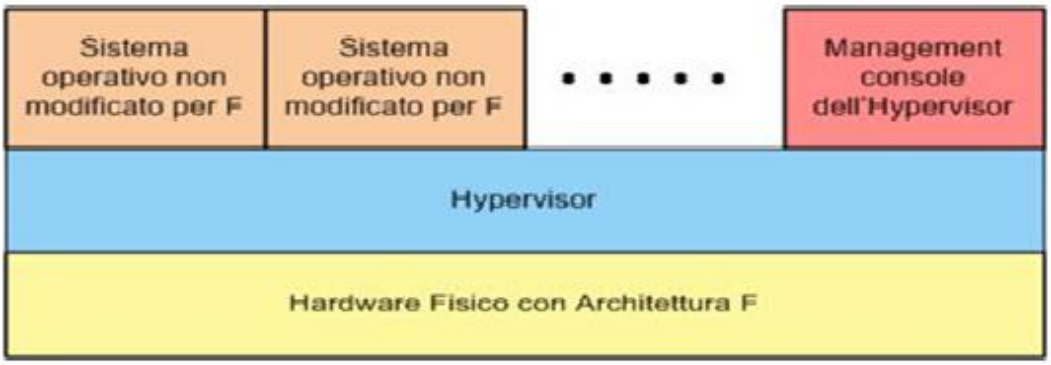

fig. 1.7

1.5.3 Paravirtualization

Un'altra tecnica di virtualizzazione è la paravirtualizzazione (fig. 1.8). L'hypervisor presenta alle macchine virtuali una versione modificata dell'hardware sottostante, mantenendone tuttavia la medesima architettura.
Il sistema operativo in esecuzione sulle virtual machine è invece modificato per evitare alcune particolari chiamate di sistema o istruzioni privilegiate.
Non viene invece modificata l'ABI(Application Binary Interface) e perciò le applicazioni possono essere eseguite senza modifiche. Questa tecnica permette di ottenere un decadimento delle prestazioni minimo rispetto al sistema operativo non virtualizzato, dato che le istruzioni provenienti dalle macchine virtuali vengono eseguite quasi tutte direttamente sul processore senza che intervengono modifiche e solo una parte minore utilizza il sistema trap.
Esempi di paravirtualizzazione sono Xen e User-mode Linux, Proxmox.

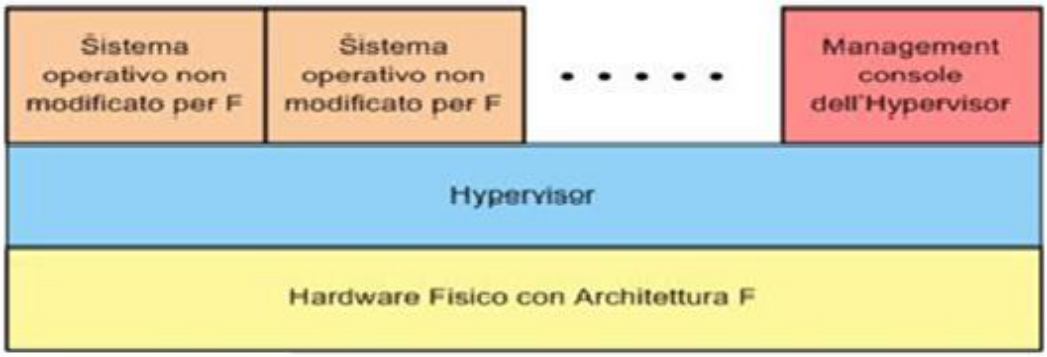

fig. 1.8

1.5.4 Operating System level Virtualization

Una quarta tecnica di virtualizzazione è la cosiddetta Operating System level Virtualization (fig. 1.9), nella quale non si utilizza un hypervisor, ma la virtualizzazione viene creata utilizzando copie del sistema operativo installato sull'host.

I sistemi guest creati saranno, a tutti gli effetti, istanze del sistema operativo host con un proprio file system, una propria configurazione di rete e delle proprie applicazioni.

Il vantaggio principale di questa tecnica è il miglior utilizzo delle risorse grazie alla condivisione di spazi di memoria. Essendo i sistemi operativo delle macchine host, le istanze guest non richiederanno un kernel privato ma utilizzeranno lo stesso con un con un conseguente utilizzo di minor memoria fisica. Questo è tuttavia il principale svantaggio di questa tecnica, che la rende inutilizzabile da coloro che vogliono eseguire sistemi operativi diversi sullo stesso host. Un altro

punto debole è il limitato isolamento tra le macchine guest. Un problema di stabilità o di performance di un solo guest, può influire negativamente sugli altri. Esempi di Operating System level Virtualization sono: Virtuozzo, Linux Vservers, Solaris Container, Proxmox.

fig. 1.9

1.6 Perché ricorrere alla virtualizzazione?

La tecnologia di virtualizzazione introduce un cambiamento radicale per quanto riguarda flessibilità delle operazioni IT, velocità di distribuzione, prestazioni delle applicazioni e disponibilità. La virtualizzazione permette di offrire un numero maggiore di funzionalità e, al tempo stesso, la possibilità di acquisire il controllo sui costi. Inoltre, permette di sviluppare ogni giorno nuove ed interessanti modalità di impiego per gli ambienti virtualizzati. Con la virtualizzazione, le risorse di dati e di elaborazione, server e storage, vengono raggruppate in maniera logica in un unico pool di risorse.
I motivi che rendono la virtualizzazione un buon investimento sono numerosi, i principali sono descritti di seguito:

- "**consolidamento ed ottimizzazione**": quando le tecnologie di

virtualizzazione non erano ancora mature si è assistito al proliferare di sistemi nelle server farm, al fine di garantire adeguati livelli di servizio ed al contempo il corretto isolamento tra contesti differenti. Ad oggi è possibile consolidare i sistemi, mantenendo gli stessi livelli di sicurezza e performance.
E' possibile consolidare numerose macchine virtuali su un numero limitato di sistemi e successivamente ripartizionare a piacimento le stesse in funzione del carico (manualmente o in modo automatico) senza interrompere l'operatività delle singole VM. Strumenti specializzati di gestione possono accendere, spegnere e spostare le VM in base ad eventi configurabili.
In tal modo è possibile ridurre la quantità dell'hardware ed ottenere un risparmio tangibile;

• "**disaccoppiamento tra hardware e servizi**": ottenuto introducendo tra la vista logica e la vista fisica dell'hardware un livello di indirezione (VMM), la cui realizzazione dipende dal tipo di virtualizzazione che si intende adottare.
Grazie a questa proprietà è possibile sostituire l'hardware o migrare le applicazioni con grande facilità ed immediatezza;

• "**flessibilità**": esiste la possibilità di convertire una macchina fisica in virtuale e viceversa, clonare macchine per differenti fini, spostare una macchina virtuale da un host ad un altro senza interromperne l'esecuzione, clusterizzare lo stesso strato di virtualizzazione per permettere la migrazione automatica delle macchine virtuali in caso di failure dell'hardware host;

• "**backup, disaster recovery, archiviazione**": possibilità di eseguire tali azioni manualmente o in modalità schedulata senza interruzione del

servizio. Qualsiasi sistema è tipicamente in continua evoluzione e spesso nasce l'esigenza di mantenere multiple versioni di tale sistema, tramite le quali rendere possibile un roll-back o eseguire uno o più fork per differenti finalità.
Il disaster recovery risulta notevolmente semplificato sfruttando la virtualizzazione: effettuare il restore di una VM è immediato e sicuro, rendendo possibile il ripristino dei servizi in un tempo molto limitato.
Anche le strategie di backup traggono dei benefici: effettuare il backup di una VM consiste nel copiare un numero estremamente limitato di file (il/i file che compone/compongono il disco virtuale ed i file di configurazione), il che permette un sistema di backup semplificato (parlando di backup della VM, non a livello di filesystem);

• "**isolamento**": ogni VM può raggiungere alti livelli di isolamento, a seconda che si tratti di virtualizzazione secondo il paradigma delle macchine virtuali o paravirtualizzazione. Sfruttando le macchine virtuali, si ottiene un isolamento completo e trasparente per i sistemi guest, come se ognuno di essi girasse su dell'hardware dedicato;

• "**supporto al networking**": tramite i software di virtualizzazione disponibili attualmente è possibile emulare differenti segmenti di reti, switching e routing. Per tale motivo, anche complesse architetture composte da segmenti di frontend, backend, DMZ e management possono essere virtualizzate ed integrate nella VLAN reali dando potenzialmente la possibilità di virtualizzare interi datacenter;

• "**clustering**": virtualizzando i server che faranno parte di un cluster, non solo sarà possibile effettuare un bilanciamento del carico a livello di VM, ma ci sarà la possibilità di bilanciare il carico dei processi che girano all'interno di ogni singola VM (dal momento che il cluster è costruito sulle

VM e non sulla macchina host.

• "**sviluppo e testing**": in alcuni contesti è comune la necessità di adoperare numerosi sistemi per brevi periodi di tempo, al fine di effettuare porting e test di compatibilità. La virtualizzazione, anche in questo contesto, è un'ottima soluzione sia tecnica che economica che permette di avere a disposizione ambienti di test operativi in poco tempo (ad esempio clonando una VM esistente) su cui effettuare tutte le prove necessarie in sicurezza;

• "**virtual appliance**": un altro aspetto da non sottovalutare è il possibile mercato di VM preconfigurate (su media e/o web), vendute con soluzioni ad hoc o secondo modelli standard in pronta consegna.

1.7 Svantaggi della virtualizzazione

In questo paragrafo, dopo aver visto i vantaggi che un sistema virtualizzato comporta, verranno analizzati anche i fattori negativi che la accompagnano. La virtualizzazione infatti, sta rivoluzionando i data center in larga parte positivamente, ma nessuna tecnologia è senza lati negativi e in questo caso, ci sono molti problemi potenziali di gestione, sicurezza, consumi energetici e ritorni d'investimento che possono pregiudicare un progetto di virtualizzazione se questo non è stato accuratamente pianificato.
Partiamo dall'idea che con un equivalente sistema virtualizzato si riduca il lavoro di gestione per l'IT. Non è del tutto vero, infatti la semplicità di poter creare una nuova virtual machine per una nuova funzionalità, può portare all'incontrollata prolificazione di virtual machine (sprawl), con la conseguenza di dover gestire un

numero considerevole di macchine sia fisiche che virtuali. Insomma, non è detto che il tempo da dedicare alla gestione si riduca, anche l'infrastruttura virtuale va gestita.

Altra cosa da valutare è che, per le applicazioni che girano sulle virtual machine molti fornitori software non offrono lo stesso supporto disponibile per quelle che girano nei server 'veri'. Il calcolo degli esborsi per le licenze può essere anche più complicato del solito in un data center virtuale: occorre studiarsi bene i termini e condizioni nei contratti di licenza dei vari fornitori.

Una volta consolidati i server può venire da pensare di aver finalmente ridotto le bollette dell'energia. Ma anche qui non si può dare niente per scontato. E' vero che i server sono in numero minore, ma ciascuno ha un tasso più alto di utilizzo della CPU, e richiede più energia. Bisogna quindi tenere in considerazione anche questo fattore quando si calcolano i benefici in un eventuale passaggio ad una soluzione virtualizzata. Un altro problema, direttamente collegato al risparmio energetico, è il dissipamento del calore, quando si eliminano molti server, il data center deve essere riconfigurato per evitare di sprecare aria condizionata su spazi vuoti.

Anche se sotto alcuni punti di vista la virtualizzazione aumenta la sicurezza, infatti la possibilità di "clonare" le virtual machine e trasferirle da una macchina all'altra, facilita di molto le strategie di disaster recovery, e quindi la riduzione dei rischi di perdite di dati e indisponibilità, d'altra parte la virtualizzazione, se non gestita in modo appropriato, comporta nuove minacce per la sicurezza dei dati e la continuità dei sistemi. Infatti, concentrare più servizi su un'unica macchina fisica, vuol dire ridurre in un unico punto tutte le vulnerabilità. Inoltre gli hypervisor non si occupano di criptatura, per cui aprono in un certo senso la strada ad attacchi cosiddetti "man-in-the-middle", in cui eventuali malintenzionati possono intercettare dati non criptati mentre le virtual machine sono trasferite da un server fisico all'altro. Come si è visto, un progetto di virtualizzazione, deve essere ben pianificato per non incorrere in una molteplicità di problematiche implementative, le quali rischiano di oscurare i benefici che se ne otterrebbero. In buona sostanza,

ogni innovazione ha i suoi pro e i suoi contro, ma un buono studio iniziale ed una progettazione accurata, fanno si che da ogni tecnologia si possano massimizzare solo i vantaggi, limitando al minimo l'incidenza degli aspetti negativi.

Capitolo 2

Proxmox VE

Proxmox VE [8] è un progetto open source sviluppato e mantenuto dall'austriaca Proxmox Server Solution GmbH con il patrocinio della Internet foundation Austria (IPA) rilasciato sotto licenza GNU GPL 3. Proxmox VE è una piattaforma di virtualizzazione, ovvero un hypervisor di tipo bare metal (tipo 2), basata su Linux Debian 6 Squeeze a 64bit , che integra in un'unica soluzione diverse tecnologie di virtualizzazione come KVM e LXC.
L'ambiente è pronto in pochi minuti e permette di eseguire facilmente macchine e appliance virtuali, il tutto gestibile tramite una semplice interfaccia web. Utilizzando più nodi hardware è possibile configurare un'intera infrastruttura virtuale, costruendo un cluster di risorse ridondato per l'alta affidabilità, rendendolo Proxmox Ve un prodotto idoneo sia alle PMI ma anche al mondo Enterprise.

Ecco le principali caratteristiche:

- Basato su Debian a 64bit.
- Linux e Windows virtual machine.
- Sistemi operativi 32 e 64 bit.
- Supporto agli ultimi chipset Intel e AMD.
- Ottimizzazione per il bare-metal per i reali carichi di lavoro.
- Layer di gestione con tutte le funzionalità necessarie per creare e gestire una infrastruttura virtuale.

- Gestione tramite interfaccia web senza la necessità di utilizzare un qualsiasi software client.

2.1 Tipologie di virtualizzazione utilizzate

Linux Containers (LXC)

LXC è un ambiente di virtualizzazione a livello di sistema operativo per l'esecuzione di più sistemi Linux isolati su un singolo host di controllo Linux. LXC funziona come un'interfaccia nello spazio utente per le funzionalità del kernel di Linux di contenimento. Gli utenti Linux possono facilmente creare e gestire sistema o delle applicazioni contenitori con un potente API e strumenti semplici.

Kernel-based Virtual Machine KVM

Kernel-based Virtual Machine è una tecnologia di virtualizzazione composta da un modulo (kvm.ko) integrato nel kernel linux (dalla release 2.6.20) che permette di sfruttare le estensioni per la virtualizzazione dei processori (Intel VT e AMD-V).
KVM di per se non esegue nessuna emulazione , ma consente la gestione della virtualizzazione dell'hardware, esponendo le sue capacità ad un secondo componente Qemu che si occupa dell'emulazione delle macchine virtuali. KVM fa quindi da "tramite" tra le richieste fatte da Qemu ed il kernel.

2.2 Funzionalità e Caratteristiche

Live Migration and Cluster HA

Proxmox VE può essere eseguito come Cluster HA (High Available), ovvero in una modalità con una configurazione che si basa su un nodo primario, detto master, e gli altri nodi che sono detti slave.
Questi nodi sono connessi insieme in modo da formare , da un punto di vista logico , un unico grande calcolatore.
Con l'implementazione di un cluster, si è in grado di bilanciare il carico di lavoro su diversi host, aumentando la disponibilità delle macchine virtuali. Inoltre si è in grado di eseguire migrazione in tempo reale (live migration) delle macchine virtuali, anche se non si dispone di storage condiviso.
In caso di manutenzione hardware, è possibile spostare "al volo" le macchine virtuali su un altro nodo, senza tempi di inattività o downtime limitato. In parole semplici, se una macchina virtuale o contenitore (VM o CT) è configurato come HA e un host fisico va in failure la VM viene riavviata automaticamente su uno dei nodi che compone il cluster.
Con la funzionalità Live Migration possiamo quindi spostare i server virtuali in esecuzione da un host fisico ad un altro senza tempi di inattività.
Nel caso di KVM, il passaggio di una macchina virtuale, in esecuzione da un host fisico ad un altro, avviene senza alcuna interruzione. Per poter utilizzare la migrazione in tempo reale, tutti i dischi virtuali devono risiedere su storage condiviso, tra gli host, come una SAN o NAS.

Snapshot

Con lo Snapshot è possibile conservare il KVM stato della macchina virtuale. Un'istantanea include il contenuto della memoria della macchina virtuale, impostazioni della macchina virtuale, e lo stato di tutti i dischi virtuali. Quando si rollback di una fotografia, si ripristina la memoria, dischi virtuali e tutte le impostazioni della macchina virtuale allo stato in cui erano quando si ha l'istantanea.
È possibile scattare una foto e continuare a utilizzare la macchina virtuale da quel punto, prendere un'altra istantanea in un momento successivo, e così via. È possibile eseguire il rollback alla snapshot di una precedente nota condizione di funzionamento del progetto se i cambiamenti non funzionano come previsto. Se si esegue il test del software, si potrebbe desiderare di salvare più istantanee come rami da una singola linea di base in un albero processo. Ad esempio, è possibile scattare una foto prima di installare diverse versioni di un'applicazione per fare in modo che ogni installazione ha inizio da una linea di base identica.

Backup & Restore

Lo strumento di backup integrato (vzdump) crea istantanee (snapshot) di guest virtuali LXC e KVM. In pratica viene creato un archivio tar dei dati VM o CT che comprende i dischi virtuali e i dati di configurazione.
Caratteristiche principali:

- Completa integrazioni GUI, ma funziona anche via CLI.
- Live Backups via LVM snapshot.
- Schedulazione di job di backup.
- “Backup Now” tramite GUI.

- Restore via GUI.
- Tutti i job possono essere monitorati tramite GUI.

Networking

Proxmox VE utilizza un modello di rete detto “bridge”e tutte le macchine virtuali possono condividere un bridge, come se i cavi di rete virtuali di tutte le guest machine siano collegati allo stesso switch.
Per il collegamento di macchine virtuali con il mondo esterno, i bridge sono collegati alle schede di rete fisiche dei server host a cui sono assegnate configurazioni di rete.
Per una maggiore flessibilità, sono supportate VLAN (IEEE 802.1Q) funzionalità di bonding e network aggregations.
In questo modo è possibile costruire complesse reti virtuali flessibili per i gli hosts, sfruttando tutta la potenza dello stack di rete Linux.

Storage

Proxmox VE utilizza un modello di storage molto flessibile.
Le immagini delle macchine virtuali possono essere memorizzati nello storage locale oppure su storage condiviso come NFS e SAN (ad esempio utilizzando

iSCSI o FC) e infine è supportato l'utilizzo di DRBD per le vm KVM.
Tecnologie di storage supportate:

- Archiviazione locale (obbligatorio)
- iSCSI
- FC
- NFS

In pratica possono essere utilizzate tutte le tecnologie di storage disponibili e supportate da Debian Linux.

Management

Proxmox VE è semplice da usare e non vi è alcuna necessità di installare uno strumento separato per la gestione, o di altri nodi supplementari di management né di database esterni adottando la filosofia NO SPOF (Single Point of Failure). Se si utilizza già un cluster, è possibile collegarsi a qualsiasi nodo che lo compone, per gestire l'intero cluster.
La gestione avviene tramite una console Web, basata su framework javascript, e consente all'amministratore di controllare tutte le funzionalità.

- Interfaccia di ricerca rapida, in grado di gestire migliaia di VM
- Console VNC sicura, supporto SSL
- Procedura guidata per la creazione di server virtuali e contenitori
- Perfetta integrazione e gestione con cluster VE Proxmox 2.0
- Gestione delle support subscription
- Gestione dei permessi sugli tutti gli oggetti (VM, CT, Storage, etc)
- Supporto multiplo alle fonti di autenticazione (locali, AD MS, LDAP)

- Tecnologie AJAX per gli aggiornamenti dinamici delle risorse
- Basato sul framework Ext JS 4.x JavaScript.

Requisiti di sistema

Raccomandati:

- Dual or Quad Socket Server (Quad/Six/Hexa Core CPUs´)
- CPU: 64bit (Intel EMT64 or AMD64)
- Intel VT/AMD-V capable CPU/Mainboard (per poter effettuare la virtualizzazione con l'hypervisor KVM)
- 8 GB RAM va bene
- Hardware RAID with batteries protected write cache (BBU) or flash protection
- Dischi veloci, le migliori performance si ottengono con dischi a 15k rpm SAS in RAID10
- Almeno due schede di rete con indirizzi IP e nomi host configurati sul DNS (NB: il numero di schede di rete sale a 3 se si utilizza uno storage esterno con connessione IP, quale NFS, iSCSI, FCoE)

Requisiti minimi:

- CPU: 64bit (Intel EMT64 or AMD64)
- Intel VT/AMD-V CPU/Mainboard (per poter testare la virtualizzazione con KVM)
- 1 GB RAM
- Disco rigido
- Una scheda di rete con indirizzo IP e nome host configurato sul DNS

Grafical User Interface

Come mostra la fig. 2.1, l'interfaccia [9] proposta da Proxmox è molto semplice da comprendere e da gestire. E' accessibile da qualsiasi browser attraverso l'url https://youripaddress:8006 con le credenziali di accesso di login e password (youripaddress, login e password sono dati che dovranno essere impostati inizialmente durante l'installazione di Proxmox).

Questa interfaccia utente è composta da quattro settori principali:

Header

In cima. Mostra le informazioni di stato e contiene i pulsanti per le azioni più importanti, come il logout, creare nuove virtual machine o containers, cercare oggetti (VM o containers) nel datacenter.

Resource Tree

Sul lato sinistro. Mostra un albero di navigazione dove è possibile selezionare oggetti specifici, per verificare poi, attraverso il Content Panel, il proprio stato.

Log Panel

Nella parte inferiore. Visualizza le voci del registro per le attività recenti. Grazie ad esso è possibile verificare se le operazioni eseguite dalla macchina sono andate a buon fine o meno, e se qualcun altro sta lavorando su un altro nodo del cluster, il tutto, in tempo reale.

Content Panel

Al Centro. Visualizza le opzioni di configurazione e lo stato degli oggetti selezionati.

Per esempio: i permessi, i backup e quindi gli snapshot, le generalità di una VM (RAM, HD, SO ecc…) , la console interattiva di una VM e molto altro.

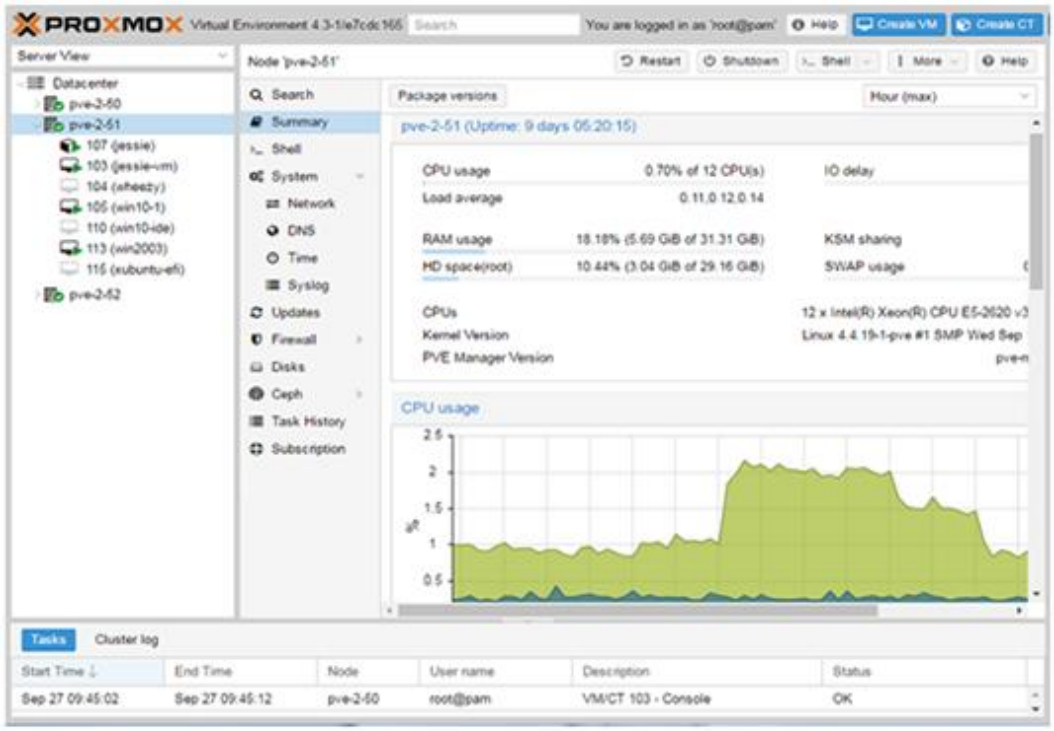

fig. 2.1

Capitolo 3

La pratica

Lo scopo principale è quello di creare un sistema che gestisca l'archiviazione delle e-mail di un istituto d'istruzione superiore. La scelta è ricaduta su di un server con al suo interno installato Linux Debian. Successivamente, si è installato Proxmox VE secondo la configurazione guidata, impostando l'ip e le credenziali del login. Fatto ciò, abbiamo optato per la creazione di due macchine virtuali, ognuna con un obiettivo ben definito.

Una macchina virtuale avrà il compito di archiviare le e-mail, mentre l'altra sarà il Domain Controller, ovvero quel server che gestisce le richieste di autenticazione per la sicurezza (login, controllo dei permessi, ecc.) e organizza la struttura del dominio in termini di utenti, gruppi e risorse di rete fornendo dunque un servizio di directory service.

La macchina virtuale, destinata all'archiviazione e-mail, contiene come sistema operativo Windows 10. Durante la creazione della macchina virtuale, oltre a scegliere tutti i valori che la coinvolgono (quantità di RAM, di ROM ecc…) è possibile selezionare la periferica di avvio del sistema (*boot order*), grazie a questa opzione, è stato semplice installare il sistema operativo da CD.

Per il Domain Controller, la situazione cambia notevolmente.

Quest'altra macchina virtuale è stata creata da un altro server attraverso un processo chiamato "migrazione di un server in esecuzione da fisico a virtuale o migrazione P2V".

Questa operazione consiste nella migrazione del sistema operativo (nel nostro

caso si tratta di Windows Server 2012), dei programmi applicativi, e di tutti i dati presenti nell'hard disk del server in esecuzione verso una macchina virtuale, tutto questo per rendere la macchina virtuale il meno differente possibile dal server che si vuole virtualizzare.

3.1 Migrazione P2V

Poiché il meccanismo di attivazione e l'architettura di sistema sono saldamente consolidate con l'attuale hardware, sono necessarie alcune azioni preliminari prima della migrazione di sistemi operativi Windows esistenti.

Quindi bisogna verificare che non ci siano *device* nascosti presenti nel server fisico in esecuzione. Per effettuare tale controllo basta digitare `set devmgr_show_nonpresent_devices = 1` e successivamente `start devmgmt.msc`, fatto ciò si aprirà il device management che mostrerà tutti i dispositivi (compresi quelli nascosti), così da poter rendere visibili gli eventuali dispositivi nascosti, presenti nel server.

Fatta questa verifica bisogna scaricare ed eseguire un utility chiamata MergeIDE. Il sistema Windows scrive alcune informazioni di registro sul controller IDE o SATA fisico nel Registro di sistema, MergeIDE modifica il registro Windows e bypassa i riscontri del controller, permettendo quindi lo "spostamento" dell'installazione di Windows su un altro ambiente hardware. Dopo aver eseguito MergeIDE, abbiamo creato una nuova macchina virtuale su Proxmox con la stessa dimensione del disco che andremo a virtualizzare. Successivamente abbiamo scaricato Selfimage. Esso è un programma di utility open source di imaging del disco rigido in grado di creare, comprimere e ripristinare l'immagine di un disco rigido o partizione. E' utile per il backup di

un disco rigido, nonché il ripristino a uno stato precedente. Questa utility è composta da due sezioni: la sezione di input, dove si può selezionare intere unità o partizioni che verranno poi estratte e copiate nella zona di destinazione specificata nella sezione di output. Questa zona di destinazione può essere o un file, o un hard disk oppure un NBD (*Network Block Device*). A noi perciò è interessato l'NBD, ovvero uno "spostamento via rete" del nostro device. Per eseguire questa transazione si è dovuta aprire una connessione su una porta, nel nostro caso la porta 1024. Per aprire questa connessione abbiamo scaricato Putty (console SSH), ci siamo connessi all'ip in cui Proxmox si trova e, dopo aver inserito l'username e la password, abbiamo digitato questo comando :

```
qemu-nbd -t /dev/dm-9 -p 1024 –f raw.
```

Perciò questo comando apre la connessione e il transito del NBD sulla porta 1024, identificando come destinazione il device della nostra macchina virtuale appena creata così da sovrascriverlo, e specificando il formato del file come .raw. Eseguito questo comando, abbiamo aperto SelfImage e inserito nella sezione di input l'hard disk del nostro server fisico, mentre nella sezione di output abbiamo specificato, nella sezione dedicata al Network Block Device, l'ip di Proxmox e la porta 1024 di comunicazione. Concluso il lavoro di SelfImage, abbiamo interrotto la connessione sulla porta 1024 e abbiamo eseguito con successo la nuova macchina virtuale, concludendo così la migrazione P2V.

Tuttavia questo metodo non ha successo se si vuole virtualizzare solo una partizione e non l'intero hard disk. Perciò, per ottimizzare al meglio lo spazio, abbiamo deciso di intervenire, riducendo al minimo lo spazio occupato dalla nostra macchina virtuale attraverso l'utilizzo di *SystemRescueCd*. SystemRescueCd è un vero e proprio sistema operativo avviabile da cd, spesso utilizzato per recuperare dati da un disco danneggiato, che agisce sulle partizioni presenti nel disco, consentendo una qualunque modifica, una copia bit a bit e molto altro. Dopo averlo avviato nella nostra macchina virtuale, abbiamo individuato la nostra partizione ed abbiamo selezionato quel settore in

cui era presente il nostro sistema operativo, tralasciando il restante spazio vuoto della partizione e lo abbiamo copiato in una nuova partizione, risparmiando, nel nostro caso, 290 GB di spazio poiché l'intero hard disk occupava 320 GB di memoria e la partizione solo 33 GB.

Affinché la virtualizzazione del Domain Controller sia del tutto completa, manca ancora un ultimo passaggio, l'installazione dei VirtIO Drivers. Questi sono driver paravirtualizzati per KVM / Linux; essi consentono l'accesso diretto (paravirtualizzato) per i dispositivi e le periferiche alle macchine virtuali che li utilizzano, al posto di quelli emulati, che sono più lenti.

Capitolo 4

Conclusione

Questo lavoro di tesi ha permesso di analizzare la virtualizzazione e Proxomox a fondo, rendendo ciò che è stato fatto nella pratica più chiaro e logico. Riepilogando, abbiamo creato due macchine virtuali (una dedicata all'archiviazione delle e-mail e l'altra farà da Domain Controller) che occupano circa 70 GB in totale su 1 TB disponibile, comportando così un guadagno in termini di spazio di memoria e quindi economici, senza considerare tutte le funzionalità, spiegate nei capitoli precedenti, che sia la virtualizzazione sia Proxmox offrono (monitoraggio delle risorse, snapshot, backup, High Availability ecc..). Questo è solo un piccolo assaggio di ciò che Proxmox e la virtualizzazione possono fare ovviamente. Tuttavia non era necessario fare altro, considerando le richieste del cliente. Abbiamo però generato le nostre macchine virtuali occupando meno spazio possibile, dando così modo, in futuro, alla eventuale aggiunta di altre macchine virtuali che potranno avere nuovi compiti da svolgere.

Uno dei principali svantaggi che comporta l'utilizzo di questa piattaforma è purtroppo il leggero decadimento delle prestazioni delle macchine virtuali rispetto ai sistemi fisici che sono stati virtualizzati; prezzo da pagare se si vuole ottenere tutte le agevolazioni che ne comportano.

Ringraziamenti

Alla fine di questo lungo percorso mi sento in dovere di fare due ringraziamenti. In primo luogo ringrazio la mia famiglia che mi sono stati vicini e mi hanno sostenuto in questo periodo. In secondo luogo ringrazio il Prof. Fausto Marcantoni ed il Dott. Francesco Compagnucci che mi hanno dato l'opportunità di eseguire un lavoro su un argomento molto attuale e di notevole importanza. L'interesse e la competizione che questo argomento sta destando a livello mondiale sono stati per me molto stimolanti. Durante questo studio ho potuto constatare in prima persona cosa significhi contribuire ad attività di ricerca ed inoltre ho avuto la possibilità di crescere molto rapidamente a livello professionale.

Bibliografia

[1] Domenico Francesco De Angelis, 2014, “La macchina nella macchina: Virtual Machine”, http://dadf.altervista.org/blog/virtual-machine/

[2] Margaret Rouse, 2016, “hypervisor”, http://searchservervirtualization.techtarget.com/definition/hypervisor

[3] Margaret Rouse, 2016, “bare-metal hypervisor”, http://searchservervirtualization.techtarget.com/definition/bare-metal-hypervisor

[4] Margaret Rouse, 2016, “Type 2 hypervisor (hosted hypervisor)”, http://searchservervirtualization.techtarget.com/definition/hosted-hypervisor-Type-2-hypervisor

[5] Wikipedia, 2017, “Storage Area Network”, https://it.wikipedia.org/wiki/Storage_Area_Network

[6] Wikipedia, 2016, “Network Attached Storage”, https://it.wikipedia.org/wiki/Network_Attached_Storage

[7] Enrico Giacomin, 2009, “Tipi di virtualizzazione” https://www.serverlab.it/2009/09/11/tipi-virtualizzazione/

[8] Corsinvest, 2017, “Virtualizzazione Proxmox”
http://www.corsinvest.it/proxmox/

[9] Wikipedia, 2017, “Grafical User Interface”
https://pve.proxmox.com/wiki/Graphical_User_Interface

Elenco delle Figure