Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Informatica

VMWare e tecniche di virtualizzazione: un caso di studio

Relatore:
Chiar.mo Prof.
Fabio Panzieri

Presentata da:
Fabrizio Amici

Correlatore:
Ing.
Massimiliano Casali

I Sessione
Anno Accademico 2009/2010
Laurea Specialistica

"Once Zhuangzi dreamt he was a butterfly, a butterfly flitting and fluttering around, happy with himself and doing as he pleased. He didn't know he was Zhuangzi. Suddenly he woke up and there he was, solid and unmistakable Zhuangzi. But he didn't know if he was Zhuangzi who had dreamt he was a butterfly, or a butterfly dreaming he was Zhuangzi. Between Zhuangzi and a butterfly there must be some distinction! This is called the Transformation of Things."

(2, tr. Burton Watson 1968:49)

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Lo scopo della tesi è quello di andare ad analizzare le prestazioni di un sistema che utilizza prodotti di virtualizzazione, quando questo si trova sotto enormi carichi di lavoro. Per generare il carico di lavoro verranno utilizzati una serie di benchmark e si cercherà di evidenziare i comportamenti del sistema ed i colli di bottiglia con l'aumentare del carico e delle macchine virtuali e nel contempo di portarlo al collasso.
La tesi si ricollega in questo alla virtualizzazione. Uno dei maggiori trend degli ultimi anni in ambito informatico riguarda infatti proprio la virtualizzazione. Questo termine, così abusato, viene utilizzato spesso in contesti che non gli appartengono essendoci una ricerca spasmodica da parte di produttori di software e di hardware di legare a questo termine i propri prodotti; questo perché il mercato dei prodotti software legati alla virtualizzazione oppure quello di hardware con supporto alla virtualizzazione è in continuo aumento. Ogni anno aziende come Microsoft, VMWare e Citrix investono ingenti quantità di denaro per cercare di accaparrarsi i grossi profitti derivanti da questo mercato e di prevalere sulle rivali.
La fama raggiunta dalla virtualizzazione si deve soprattutto ad un azienda: VMWare. VMWare è ad oggi il leader incontrastato del mercato mondiale riguardante prodotti software di virtualizzazione con più del 90% della fet-

ta di mercato negli U.S.A. tra le Fortune 1000[1]: la lista delle 1000 aziende statunitensi con più fatturato.

Questa continua crescita (vedi figura 1.1) è dovuta al sempre maggior nu-

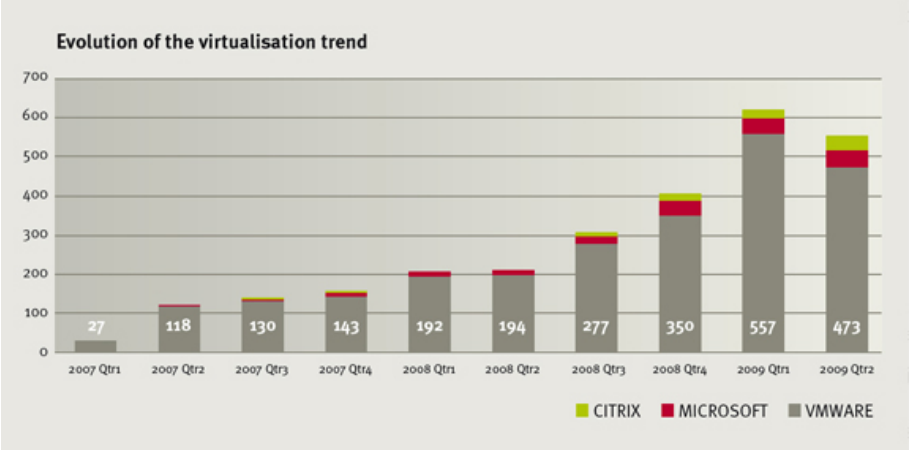

Figura 1.1: Grafico di crescita della richiesta di sistemi di virtualizzazione; a confronto: VMWare, Microsoft e Citrix [fonte: http://computerprofile-blog.com/?tag=server-virtualization]

mero di aziende che richiedono un passaggio alla virtualizzazione nell'ottica di poter usufruire di tutto quell'insieme di funzionalità che questa promette di realizzare. Tra queste promesse abbiamo continuità del servizio, alta affidabilità e consolidamento dell'infrastruttura. Tutto ciò, unito alle numerosissime storie di successo di aziende passate alla virtualizzazione, ha portato l'opinione pubblica e specializzata a ritenere questi prodotti come la panacea per ogni male. La verità purtroppo è ben lontana da ciò.

Nel 2008 ho avuto l'opportunità di iniziare a lavorare in un centro di dati sammarinese che nel 2009 ha affrontato un passaggio alla virtualizzazione. Alla SAN in Fibre Channel già presente nel centro è stata aggiunta un infrastruttura di virtualizzazione Virtual Infrastructure 3 di VMWare (VI3[2]). Durante questo passaggio mi sono sorti tutto un insieme di interrogativi a cui non avevo saputo rispondere, data anche la mia ignoranza, al tempo, in materia di virtualizzazione. Tutto questo ha però fatto nascere in me anche

[1] http://www.vmware.com/company/customers/

[2] http://www.vmware.com/products/vi/

la voglia di scoprire questo nuovo mondo e di capirne nel dettaglio il funzionamento. È proprio questa curiosità ad avermi spinto ad affrontare una tesi in questa materia, dapprima approfondendo il tutto a livello teorico, per poi passare ad un livello più pratico. Durante le fasi iniziali di avvicinamento mi sono ritrovato spesso spiazzato nel trovare quasi totalmente in letteratura opere che enfatizzavano i pregi della virtualizzazione senza analizzarne le relative debolezze e svantaggi che non fossero gli elevati costi. Per quanto riguarda la mia esperienza è infatti pratica comune sentire parlare di virtualizzazione in linea teorica senza avere un reale riscontro pratico. Questo mi ha portato a pensare che tutta questa "fiducia", bene o male che fosse riposta, necessitasse anche di un riscontro a livello pratico. Infatti, così com'è giusto "teorizzare" la pratica, è anche necessario "mettere in pratica" la teoria. Data poi la novità della materia, si sente la quasi totale mancanza di casistiche, dati e test a livello pratico, per aiutare chi ha bisogno ad orientarsi per quanto riguarda il design, la configurazione e l'amministrazione di una architettura virtualizzata.

In una serie di confronti con il mio correlatore Massimiliano Casali siamo giunti alla conclusione che sarebbe stato interessante realizzare una piccola architettura virtualizzata sulla quale poter eseguire tutta una serie di esperimenti e benchmark con lo scopo finale di mandarla in uno stato di criticità dovuta al troppo carico. Lo scopo della tesi è quindi divenuto proprio quello accennato inizialmente: la creazione attraverso dei benchmark di varie tipologie di carico con cui mettere sotto sforzo un server, che utilizza prodotti di virtualizzazione per gestire una serie di macchine virtuali, e all'aumentare progressivo di queste, registrare i vari comportamenti, colli di bottiglia e andamenti inaspettati del sistema, fino a che non è più in grado di erogare il servizio.

Per affrontare il problema è stato innanzi tutto indispensabile la scelta del software che ci permettesse di sfruttare la virtualizzazione e le macchine virtuali. Per una questione di coerenza con i prodotti di virtualizzazione utilizzati in sede lavorativa la scelta è ricaduta sulla versione gratuita di VMWare

ESXi 4 (che descriveremo in dettaglio nel Capitolo 2). In un secondo momento si è poi passati alla realizzazione di una piccola architettura di rete virtualizzata. Quest'ultima è stata realizzata con il meglio della tecnologia che avevo a disposizione, anche se distante da un reale caso di studio possibile solo all'interno di un centro di dati.
Lo stato dell'arte in materia di virtualizzazione, infatti, imporrebbe l'utilizzo di tutta una serie di tecnologie a cui non potevo accedere per problemi di costo come:

- Storage Area Network (SAN) con Fibre Channel su fibra ottica.
- Dischi Solid State Disk (SSD) o Serial Attached SCSI (SAS).
- Schede di rete e hardware di tipologia server.
- Prodotti di virtualizzazione con relativa licenza per sfruttare le funzionalità più avanzate altrimenti inaccessibili.

Infatti l'architettura virtualizzata di prova è stata realizzata con:

- Rete Gigabit Ethernet a 1 GBit/s.
- Dischi Serial Advanced Technology Attachment (SATA) in locale.
- Schede di rete e hardware di tipologia desktop.
- Prodotti di virtualizzazione versione gratuita.

Il lavoro svolto non comporta, e nemmeno vuole comportare, un avanzamento dello stato dell'arte, ma come accennato sopra, si prefigge l'obiettivo di analizzare le prestazioni di un architettura virtualizzata per fornire dati e tutta una serie di considerazioni di cui tutti possano beneficiare.
Da una lettura finale dei test realizzati possiamo affermare che alcuni erano attesi come:

- Inefficienza nel virtualizzare motori di database.
- I dischi risultano essere uno dei maggiori colli di bottiglia.

- Dopo una determinata soglia i vari overhead della virtualizzazione fanno degradare il sistema esponenzialmente .
- Un sistema virtualizzato deve evitare di raggiungere il 70%/80% della componente più critica.

Altri invece si sono rilevati essere una sorpresa:

- Elevato consumo di RAM utilizzando macchine virtuali con più processori.
- Sotto carichi molto grossi abilitare l'Hyper-Threading risulta in un maggior degrado delle prestazioni senza nessun beneficio aggiuntivo.
- In un sistema con molto carico evitare di assegnare tanti processori virtuali ad una macchina virtuale quanti ne dispone fisicamente il sistema. Per esempio in una macchina quad-core assegnare al massimo due processori virtuali ad una macchina virtuale.

I test hanno fatto affiorare tutta una serie di problematiche a cui inizialmente non avevo pensato, come per esempio la gestione dei processi da parte dello scheduler e la frammentazione dei processi quando si assegnano processori virtuali in modo che il sistema non possa gestirli insieme (per esempio 3 e 2 su una macchina quad-core). Queste problematiche possono essere prese come spunto per lavori futuri di benchmarking e approfondimento.
Nella fase iniziale erano stati studiati vari approcci possibili per la realizzazione dei test. Questi comprendevano l'utilizzo di particolari configurazioni come iSCSI, Link Aggregation e RAID che non è stato possibile realizzare in pratica. Per quanto riguarda iSCSI e Link Aggregation era stata provata una configurazione con un singolo client, il server e la terza macchina che forniva lo spazio di memorizzazione attraverso la rete con iSCSI. Purtroppo in questa configurazione c'erano vari problemi; il primo era che il client non riusciva a generare traffico a sufficienza per portare il server in uno stato di criticità nelle configurazioni dove venivano messe sotto sforzo poche macchine virtuali e quindi sarebbero state necessarie più macchine che non avevo

a disposizione. Il secondo era che i risultati non potevano essere comparati con quelli ottenuti nei test con lo spazio di memorizzazione in locale dal momento che le macchine utilizzate utilizzavano hardware differente tra loro.
Per quanto riguarda il RAID invece la sua realizzazione non è stata possibile per mancanza di un controller che fosse supportato nella *Hardware Compatibility List (HCL)* di VMWare[3].
Tutto il lavoro pratico e di configurazione svolto è stato comunque incluso nelle appendici perché considerato sia parte del lavoro di tesi che potrà essere ripreso come spunto in previsione di lavori futuri, sia perché complementa attraverso un livello pratico alcuni punti affrontati durante il percorso teorico.
Ed ora una sguardo agli argomenti trattati all'interno della tesi.

Capitolo 2: Virtualizzazione
Nel secondo capitolo si analizzeranno i concetti base legati alla virtualizzazione: la sua definizione, le macchine virtuali, virtualizzazione hosted e nativa, virtualizzazione della piattaforma x86. Si presenterà uno spunto di riflessione sui vantaggi e gli svantaggi di un eventuale passaggio di un azienda alla virtualizzazione. Si mostrerà lo stato dell'arte per quanto riguarda i prodotti di virtualizzazione forniti da VMWare e nello specifico si analizzerà l'infrastruttura fornita da vSphere e le varie funzionalità fornite dai suoi prodotti. Infine si affronterà il tema di come vSphere gestisce la rete. Si introdurranno i concetti realtivi ai virtual switch, alle Virtual Local Area Network (VLAN) e al Link Aggregation e al Multipathing.

Capitolo 3: VMWare e sistemi di memorizzazione
Nel terzo capitolo si analizzeranno i vantaggi e gli svantaggi delle principali soluzioni per quanto riguarda i sistemi di memorizzazione utilizzati da VMWare. Si presenteranno in dettaglio due macro topologie di rete: Storage Area Network (SAN) e Network Attached Storage (NAS). Si sviscereranno

[3] http://www.vmware.com/resources/compatibility/search.php

molti dei concetti legati alla SAN come: dischi, array di dischi, iSCSI, SCSI, Fibre Channel.

Capitolo 4: Valutazione Sperimentale
Nel quarto capitolo si illustrerà come è stato realizzare una semplice architettura virtualizzata. Si elencheranno le varie componenti hardware e software utilizzate per i test e si mostreranno i vari passaggi effettuati per configurare il sistema. Si forniranno anche varie foto scattate durante il design e l'assemblaggio dell'architettura.

Capitolo 5: Test
Nel quinto capitolo si elencheranno le varie tipologie di test che si andranno ad effettuare. Seguiranno una serie di grafici che mostreranno i risultati riassuntivi dell'andamento dei test con tanto di considerazioni sul loro andamento. Durante l'analisi dei risultati si evidenzieranno eventuali colli di bottiglia e criticità del sistema.

Capitolo 6: Conclusioni e sviluppi futuri
Nell'ultimo capitolo si mostreranno le conclusioni a cui si è giunti dopo tutto il percorso di questa tesi. Si cercherà di capire quali obiettivi sono stati raggiunti e quali no. Verranno evidenziati eventuali spunti per progetti e lavori futuri. Infine si ricollegherà il tema della virtualizzazione a quello del Cloud Computing.

Capitolo 2

Virtualizzazione

Il termine virtualizzazione è molto utilizzato in ambito informatico. Alcuni esempi sono:

1. Macchina Virtuale
    - Virtualizzazione della piattaforma
    - Virtualizzazione dell'applicazione
2. Memoria Virtuale
3. Virtualizzazione dei supporti di memorizzazione (*storage*)

Virtualizzazione di un'applicazione
La virtualizzazione a livello di applicazione può avvenire in vari modi come per esempio attraverso l'emulazione o la virtualizzazione multi piattaforma. Per effettuare virtualizzazione multi piattaforma, è necessario scrivere un programma in un linguaggio che viene interpretato dalla macchina virtuale software, trasformandolo in operazioni eseguibili sul sistema operativo. A questo punto il programma può essere eseguito su tutte quelle piattaforme che hanno un implementazione della macchina virtuale rendendo di fatto il programma portabile. Un linguaggio moderno che fa uso della macchina virtuale è Java: i programmi scritti in Java vengono infatti compilati in *bytecode*

che è la forma intermedia di linguaggio interpretabile dalla Java Virtual Machine (JVM), che poi ha il compito di trasformarlo in istruzioni eseguibili sul sistema operativo.

Memoria Virtuale

Un programma ha l'impressione di avere uno spazio di memoria contiguo, per quanto riguarda lo spazio logico degli indirizzi, quando in realtà non è mai così (vedi figura 2.1). Questo permette una più semplice scrittura

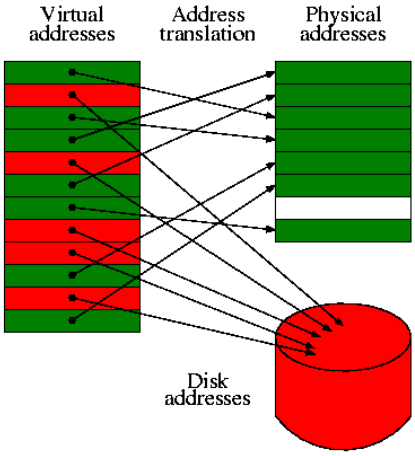

Figura 2.1: Spazio logico della memoria dal punto di vista di un programma [fonte: http://cs.nyu.edu/~gottlieb/courses/2001-02-fall/arch/lectures/lecture-22.html]

dei programmi da parte del programmatore dal momento che non si deve preoccupare della reale gestione degli indirizzi della memoria. Lo spazio di indirizzamento logico percepito è sempre formato dai 2GB (2 alla 31 indirizzi logici) che vanno dall'indirizzo 0x00000000 all'indirizzo 0x7FFFFFFF. Il compito di tramutare gli indirizzi virtuali in indirizzi fisici poi spetta alla Memory Management Unit (MMU), che fa capo a due strutture dati: il Translation Lookaside Buffer (TLB) e la Page Table (PT) (vedi figura 2.2). Il TLB è un buffer o, nelle implementazioni più sofisticate una cache nella CPU, che contiene porzioni della Page Table, una struttura dati presente su disco, che mantiene la mappatura tra gli indirizzi di memoria virtuali e

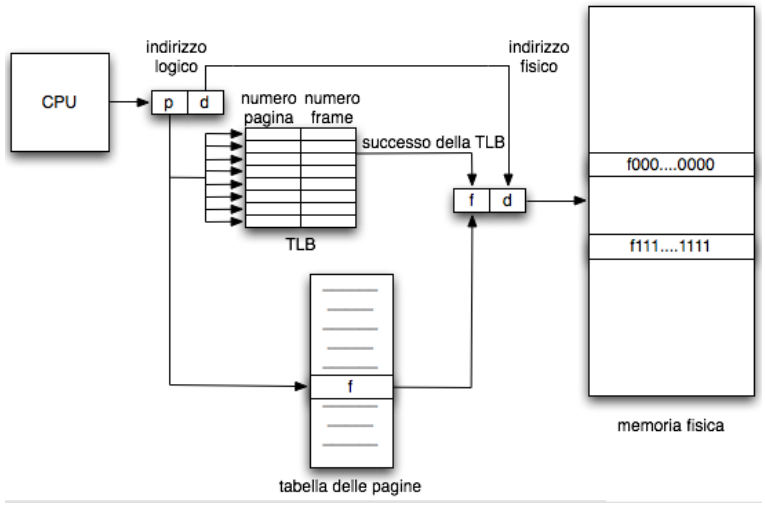

Figura 2.2: MMU: TLB e Page Table [fonte: https://forum.lithium.it/portal/articoli-menucontenuti-19/363-sistemi-operativi-memoria-centrale?start=5]

quelli reali. Ora senza voler entrare troppo nello specifico possiamo dire che gli indirizzi fisici sono univoci da un punto di vista della CPU, mentre gli indirizzi virtuali sono univoci da un punto di vista del programma. In caso di esecuzione di più programmi contemporaneamente la Page Table potrebbe dover mappare lo stesso indirizzo virtuale (appartenente però uno ad un programma ed uno all'altro), su diversi indirizzi fisici e questo genererebbe dei conflitti. Per evitare questo problema viene associato ad ogni spazio di indirizzamento virtuale un valore univoco detto ASID (Adress Space ID), che permette quindi di rendere non ambigui quelli che prima erano due indirizzi virtuali identici. Da notare che l'ASID non corrisponde al PID (Process ID) dal momento che thread appartenenti allo stesso processo possiedono PID diversi, ma hanno lo stesso ASID dal momento che condividono lo stesso spazio di memoria.

Virtualizzazione dei supporti di memorizzazione

Il sistema ci fornisce uno spazio logico che viene poi mappato su uno spazio fisico preso solitamente da un insieme di dischi che possono essere eterogenei

tra loro. Lo spazio fornito viene visto da chi ne usufruisce come spazio logico contiguo mentre in verità è distribuito su più dischi fisici. In Linux, per esempio, è possibile fare ciò attraverso l'utilizzo del pacchetto `LVM2`.

2.1 Virtualizzazione, Emulazione e Traduzione Binaria

Un emulatore, in principio, era una macchina hardware che permetteva l'esecuzione di un programma scritto per un altro tipo di hardware replicandone il funzionamento. Al giorno d'oggi viene considerata emulazione anche quella software. L'emulazione, in questa ottica, si basa su una replica software dell'hardware di una macchina, come la memoria, la CPU ed eventuali periferiche di I/O. L'esecuzione di un programma scritto per Amiga, per esempio, può essere eseguito su un moderno PC sfruttando un emulatore. Tutto ciò avviene attraverso un processo di lettura, decodifica, esecuzione (ciclo read-decode-execute) delle operazioni del programma che altrimenti non sarebbero comprensibili al sistema su cui sta eseguendo l'emulatore.
Un altra forma di emulazione è quella che viene chiamata traduzione binaria (*binary translation*). Nella traduzione binaria le operazioni di un file sorgente, progettato per eseguire su una architettura hardware, vengono tradotte in operazioni adatte ad eseguire su un altro tipo di architettura. La traduzione binaria permette un esecuzione più veloce rispetto all'emulazione, dal momento che le istruzioni sono trasformate, eliminando il ciclo di lettura, decodifica ed esecuzione, che è la causa dell'enorme lentezza di un emulatore. L'iniziale *overhead* dovuto alla traduzione, viene compensato dalla veloce esecuzione successiva di codice già trasformato. Senza addentrarci oltre ricordiamo che la traduzione binaria può essere statica o dinamica. Un esempio di traduzione binaria dinamica è quella del Just In Time Compiler (JIT Compiler) di Java. Anche prodotti di virtualizzazione come VMWare

fanno utilizzo di traduzione binaria dinamica [25].
La virtualizzazione differisce dall'emulazione dal momento che le istruzioni del software virtualizzato devono essere per forza compatibili con quelle dell'architettura sottostante. Per esempio su un architettura x86 è possibile virtualizzare un sistema Windows, Linux per x86 ma non un Macintosh per PPC. Allo stesso tempo un processore Intel non potrà eseguire le operazioni di un programma Amiga, che necessita di un processore Motorola. Il fatto che la virtualizzazione differisca dall'emulazione, non implica che prodotti di virtualizzazione non possano farne uso. Prodotti di virtualizzazione come VMWare Workstation per esempio utilizzano l'emulazione nella gestione delle periferiche virtuali. Il comportamento di una periferica video virtuale viene emulato dallo strato di virtualizzazione in modo che sia comprensibile al driver della scheda video fisica installata nel sistema [13].

2.2 Il passaggio alla virtualizzazione

La virtualizzazione è un tema molto scottante e d'attualità in questi ultimi anni. Grosse aziende come VMWare, Xen e Microsoft hanno rilasciato prodotti che permettono di virtualizzare la piattaforma x86. Dopo poco tempo questi prodotti hanno cominciato ad essere abbracciati da tantissime aziende in tutto il mondo, per poter beneficiare delle funzionalità e dei vantaggi che questi prodotti offrono: consolidamento, continuità del servizio, recupero in caso di disastro, ecc...
Vediamo in seguito come mai un'azienda dovrebbe, oppure no, considerare un passaggio alla virtualizzazione.

2.2.1 Situazione attuale di molte aziende

Molte organizzazioni, centri di dati e compagnie, al giorno d'oggi, si ritrovano a dover convivere con infrastrutture complesse, eterogenee e a dover gestire piattaforme e software diversi tra loro. Queste complessità derivano spesso dal lungo periodo di attività di un azienda che deve convivere con

tipologie diverse di applicazioni (client-server, web, ecc...), scritte in vari linguaggi di programmazione, per funzionare sopra sistemi operativi diversi per tipologia e versione. Per molte aziende è poi impossibile abbandonare queste vecchie applicazioni perché ancora costrette a fornire agli acquirenti la manutenzione. Tutto questo porta a:

- Aumento nella quantità di server utilizzati. Spesso questi server gestiscono solo un applicativo inutilizzando gran parte delle risorse della macchina.
- Infrastruttura complessa dovuta all'utilizzo di sistemi operativi e motori di database differenti.
- Isole di connettività con dati non centralizzati.
- Alti consumi energetici dovuti proprio alla grande quantità di server utilizzati.
- Bassa scalabilità (dipendente anche dalla buona o cattiva progettazione di un centro).

2.2.2 Perché passare alla virtualizzazione

La motivazione più grande che spinge un azienda ad intraprendere la strada della virtualizzazione è quella di sfruttare le grosse funzionalità che vengono fornite dai prodotti odierni di virtualizzazione. Tra i benefici [28] dovuti a questi prodotti troviamo:

- **Scalabilità dell'infrastruttura**
  Un infrastruttura virtualizzata è facilmente scalabile, sia per un eventuale richiesta di maggiore potenza di calcolo, sia di disponibilità di spazio di memorizzazione.
- **Consolidamento dell'infrastruttura**
  Più un infrastruttura è semplice più si abbassano i costi di realizzazione, gestione e manutenzione.

- **Ritorno d'investimento**
  La minor spesa nella gestione e nei consumi dell'infrastruttura porta nel lungo termine a recuperare la grossa spesa iniziale.

- **Centralizzazione dei dati**
  Un infrastruttura virtualizzata permette una gestione centralizzata dei dati (N.B.: è una centralizzazione a livello logico ma distribuita a livello fisico) semplificando la gestione dei *backup*, diminuendo la possibilità di avere dati inconsistenti e sfruttando al massimo lo spazio di memorizzazione.

- **Continuità del servizio**
  Un infrastruttura virtualizzata favorisce la continuità del servizio. Un azienda deve fornire i propri servizi continuamente 24 ore su 24, 7 giorni su 7, minimizzando i periodi di disservizio e pianificando una strategia per tornare operativa in caso di guasti o di disastri.

2.2.3 Perché non passare alla virtualizzazione

Nonostante la virtualizzazione venga spacciata come una panacea per ogni male, non è sempre consigliabile intraprendere un'opera di virtualizzazione, soprattutto non prima di avere analizzato con cura le necessità della propria azienda e preso in considerazione strade alternative. Il principale fattore negativo è dovuto al costo molto elevato che il passaggio alla virtualizzazione comporta e questo è un fattore che ogni azienda dovrebbe considerare. Seguendo una serie di scelte non ben ponderate, un'azienda potrebbe ritrovarsi in una situazione in cui l'infrastruttura finale non la soddisfa. Per esempio, il ritorno d'investimento sotto forma di risparmio energetico, garantito dalla virtualizzazione, è proporzionale alla grandezza del centro e al numero di macchine che si vogliono virtualizzare. Per poche macchine sarà esiguo e per molte più cospicuo. Nel caso di virtualizzazione di grossi centri però entra anche in gioco il fattore licenze, che in caso di prodotti di virtualizzazione proprietari come VMWare, comporta un aumento dei costi, riducendo

il ritorno di investimento. Per questo motivo esistono soluzioni alternative più economiche con cui molte aziende, soprattutto quelle piccole, si possono trovare meglio. È quindi d'importanza vitale spendere tutto il tempo necessario per fare un bilancio fra i costi, i bisogni dell'azienda ed i reali vantaggi che si possono trarre da un'opera di virtualizzazione, per non ritrovarsi alla fine con qualcosa di costoso e diverso da quanto ci si aspettava, solo per aver seguito "la moda" del momento.

2.3 Concetti di base sulla virtualizzazione

Possiamo definire in generale la virtualizzazione come [23]:

> *"La virtualizzazione è la separazione di una risorsa o di una richiesta di servizio dalla metodologia con cui questa richiesta viene effettivamente servita a livello fisico"*

Per esempio, attraverso l'utilizzo di memoria virtuale un programma ha la possibilità di accedere a più memoria di quanta ce ne sia fisicamente installata sulla macchina attraverso l'utilizzo di *swap* su disco. Questo ragionamento chiaramente può essere esteso anche ad altre componenti come la rete, il disco, il sistema operativo, ecc... Comunemente la virtualizzazione è stata presentata in maniera semplificata al pubblico, come la possibilità di far eseguire contemporaneamente sulla stessa macchina fisica, sistemi operativi diversi. Questa affermazione è corretta solo in parte, dal momento che ciò che è importante, non è tanto il sistema operativo installato, ma l'ambiente virtuale che ne permette l'esecuzione, indipendentemente da quale sistema operativo poi si scelga di utilizzare. Un ambiente virtuale non è altro che un insieme di componenti hardware e periferiche implementate via software, conosciuto con il nome di macchina virtuale (vedi figura 2.3). Una macchina virtuale può permettere l'esecuzione di un sistema operativo "ospite" ed uno o più applicazioni che eseguono sopra di esso. Ogni macchina virtuale a sua volta esegue sopra ad uno strato di virtualizzazione chiamato tecnicamente *"hypervisor"* (supervisore) e possiede un Virtual Machine Monitor (VMM),

facente sempre parte dello strato di virtualizzazione, che ne soddisfa i bisogni di CPU, memoria e periferiche. Ogni *hypervisor* contiene più VMM, che soddisfano le macchine virtuali e competono sulle risorse fisiche della macchina "ospitante". Esistono due tipologie di *hypervisor* [23]:

1. **Tipo 1** o **"bare-metal, nativo"**
2. **Tipo 2** o **"hosted, ospitato"**

Un *hypervisor hosted* è situato al di sopra di un sistema operativo detto ospitante (Windows, Linux, ecc...). In questo caso si parla di *"Hosted Virtualization"*. Un esempio di prodotto che permette la Hosted Virtualization è VMWare Server.
Un *hypervisor bare-metal* è invece situato direttamente sopra l'hardware della macchina e ne possiede il controllo diretto e privilegiato. In questo caso si parla anche di virtualizzazione nativa. Un esempio di virtualizzazione nativa è data da prodotti come VMWare ESX o XenServer.
Attraverso la virtualizzazione nativa è possibile ottenere prestazioni di gran lunga migliori rispetto alla virtualizzazione *hosted*, perché ci sono meno strati per arrivare all'hardware. Entrambe le tipologie, permettono di eseguire macchine virtuali al loro interno e dal punto di vista del filesystem (qualunque esso sia) vengono considerate come un insieme di file (file del disco fisso, file della RAM, file per lo swap, file di configurazione, ecc...). Una macchina virtuale può essere creata in più modi:

1. Seguendo un normale processo di creazione da menù.
2. Trasformando una macchina fisica in una virtuale attraverso un programma che effettua questa conversione (chiamata P2V: Physical to Virtual), come per esempio VMWare Converter [29].
3. Clonandone una già esistente.

Una macchina virtuale rispetto la controparte fisica fornisce numerosi vantaggi:

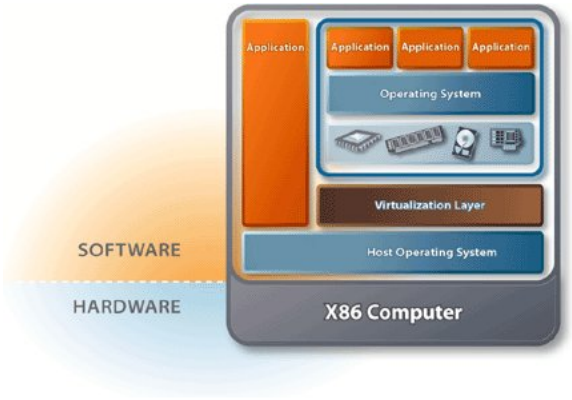

Figura 2.3: Architettura VMWare, quella cerchiata in blu è una Macchina Virtuale (VM)
[fonte: http://www.pds-site.com/vmware/ws/default.htm]

1. Può eseguire concorrentemente con altre macchine virtuali, sfruttando al massimo l'utilizzo delle risorse hardware del sistema ospite.

2. Può essere facilmente clonata, archiviata o ripristinata.

3. Può eseguire i programmi in completo isolamento.

4. È portabile tra sistemi fisici che supportano la stessa tecnologia di virtualizzazione.

5. Può essere generata attraverso *template* per una rapida creazione, configurazione e attivazione.

2.4 Virtualizzazione della piattaforma x86

VMWare riuscì nel lontano 1998 a virtualizzare per la prima volta la piattaforma x86, attraverso una combinazione di traduzione binaria ed esecuzione diretta sull'hardware, che permise a più macchine virtuali di operare sullo stesso computer in completo isolamento con un *overhead* in termini prestazionali accettabile. Le principali sfide che VMWare ha dovuto affrontare riguardano la virtualizzazione di queste componenti principali:

- CPU
- Memoria
- Periferiche

2.4.1 Virtualizzazione della CPU

Un sistema operativo quando viene installato su una macchina fisica crede di "possedere" l'hardware su cui è installato. È infatti a diretto contatto con esso e lo gestisce in maniera privilegiata. Con la virtualizzazione invece, viene messo, tra il sistema operativo e l'hardware, uno strato software di virtualizzazione, che ne impedisce il contatto diretto. A questo punto è lo strato di virtualizzazione a gestire le richieste hardware per conto del sistema operativo. La sfida maggiore nel virtualizzare la piattaforma x86 però, risiede nel fatto, che alcune operazioni non possono essere eseguite correttamente dal sistema operativo, se non a diretto contatto con l'hardware. L'errata esecuzione di queste operazioni porta l'intero sistema al malfunzionamento. Per questo motivo è stato necessario da parte di VMWare trovare una metodologia per intrappolare queste richieste e trasformarle, attraverso un processo di traduzione binaria dinamica, in operazioni che è possono essere eseguire correttamente dal sistema. Al giorno d'oggi esistono tre possibilità per virtualizzare la CPU di un sistema operativo:

1. *Full Virtualization* (Virtualizzazione completa)
2. *Paravirtualization* (Paravirtualizzazione)
3. *Hardware Assisted Virtualization* (Virtualizzazione assistita dall'hardware)

Full Virtualization

La Full Virtualization [25] (figura 2.4) è il sistema più utilizzato, visto che permette di ottenere prestazioni molto buone con il massimo grado di

portabilità. La portabilità è data dal fatto che il sistema operativo non è stato minimamente modificato e quindi può eseguire allo stesso modo sia in maniera nativa, sia virtualizzata senza esserne a conoscenza. Le operazioni che non sono pericolose sono direttamente eseguite sull'hardware in maniera molto veloce, mentre quelle che comportano problemi vengono intrappolate e trasformate dinamicamente in operazioni sicure attraverso traduzione binaria e memorizzate in una cache per velocizzare eventuali accessi futuri.

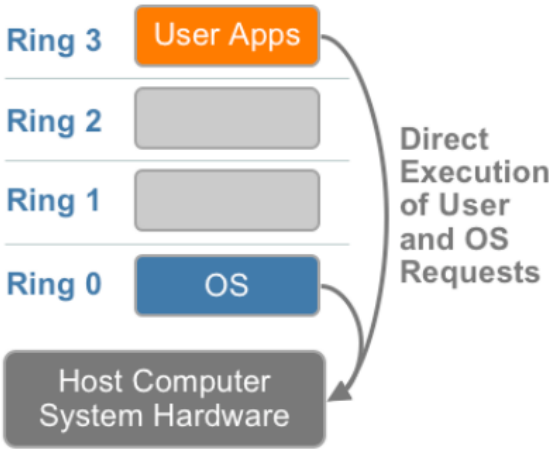

Figura 2.4: Esempio di Full Virtualization [fonte: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf]

Paravirtualization

La Paravirtualization [25] (vedi figura 2.5) consiste nel modificare un sistema operativo in modo che le operazioni che non possono essere virtualizzate, invece che intrappolate e poi tradotte, vengano direttamente richieste all'hypervisor. Questo si traduce in un minor overhead rispetto alla full virtualization. Le modifiche che devono essere apportate al sistema operativo, in alcuni casi sono invasive, e richiedono che il sistema sia modificabile. Questo rende la paravirtualizzazione difficile da effettuare su sistemi Windows.

Xen ha introdotto per primo la paravirtualization[1] per sfruttarla all'interno dei sistemi operativi Linux, adattando il kernel ai propri fini e modificando i vari driver. Quello che si ottiene attraverso la paravirtualizzazione, è una maggiore velocità d'esecuzione, ma anche una riduzione in portabilità. Un altro esempio famoso di utilizzo di paravirtualizzazione sono i VMWare Tools utilizzati da VMWare per ottimizzare i driver di una macchina.

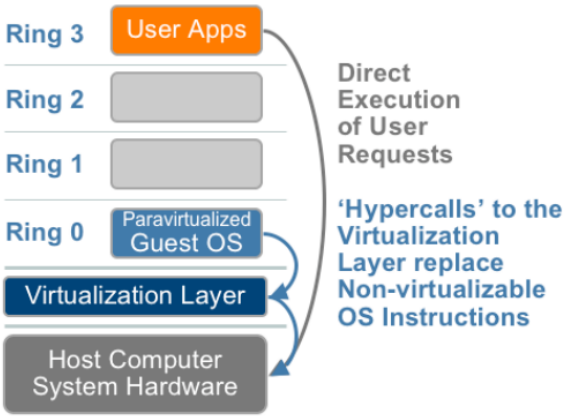

Figura 2.5: Esempio di Paravirtualization [fonte: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf]

Hardware Assisted Virtualization

L'Hardware Assisted Virtualization [25] (vedi figura 2.6) è una tecnica nuova, che può essere utilizzata solo nel caso di acquisto di uno dei nuovi processori, che supportano la tecnologia di virtualizzazione di casa Intel e AMD chiamata Intel Virtualization Technology (VT-x) in un caso, e AMD-V nell'altro. Queste classi di processori facilitano l'accesso all'hardware della macchina da parte delle macchine virtuali, permettendo ai VMM di eseguire in una speciale modalità privilegiata, al di sotto del sistema operativo. A questo punto ogni chiamata ad operazioni che hanno bisogno di accesso diretto all'hardware vengono automaticamente intrappolate e reindirizzate ai

[1] http://www.xen.org/community/xenhistory.html

VMM. Questo tipo di virtualizzazione si prospetta come quello più veloce rispetto alle due tipologie precedenti. La verità purtroppo è che la rigidità di questa prima generazione di Hardware Assisted Virtualization da parte di Intel e AMD non ha portato sempre ai risultati sperati ed in alcuni casi [2] è risultata meno performante della Full Virtualization, mostrando una mancanza in fatto di flessibilità e maneggevolezza.

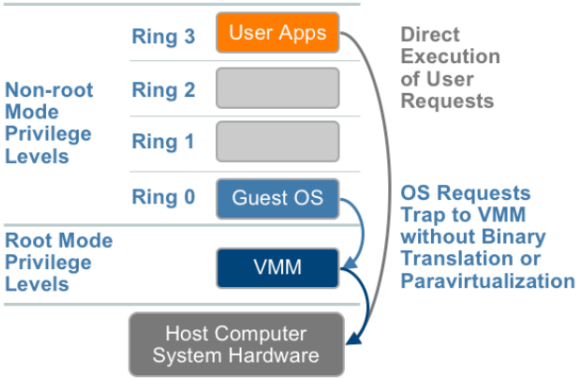

Figura 2.6: Esempio di Hardware Virtualization [fonte: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf]

2.4.2 Virtualizzazione della memoria

Il secondo aspetto critico è la virtualizzazione della memoria [25]. I concetti di base non sono tanto diversi da quelli della virtualizzazione della memoria fornita dai moderni sistemi operativi, con la sostanziale differenza, che la Memory Management Unit (MMU) delle macchine virtuali, deve essere virtualizzata. La MMU di ogni sistema operativo ospite mappa indirizzi virtuali su indirizzi fisici, che però in realtà sono altri indirizzi virtuali (figura 2.7). Nel nostro caso la memoria fisica della macchina ospitante è situata ad un livello più basso. È compito dei VMM trasformare questi indirizzi virtuali, ma che il sistema operativo ospite ritiene fisici, in indirizzi fisici reali. Nella virtualizzazione abbiamo quindi una gestione della memoria a due strati, invece che ad uno strato soltanto, come capita in un sistema senza virtualiz-

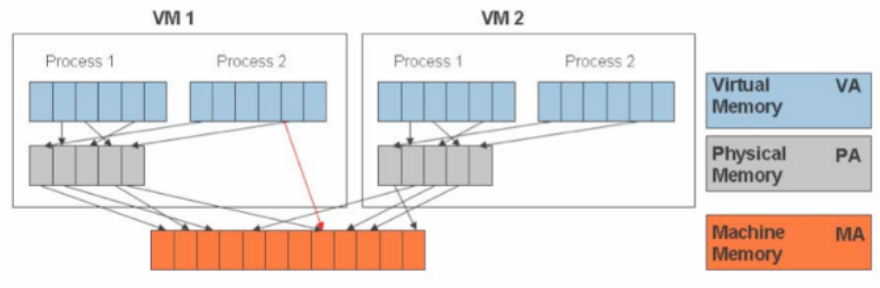

Figura 2.7: Gestione della memoria virtualizzata a due livelli [fonte: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf]

zazione. Questo doppio livello di trasformazione si traduce in un maggiore *overhead* ed è per questo motivo che i VMM utilizzano una Shadow Page Table che mantiene la mappatura diretta tra gli indirizzi virtuali del sistema ospite a quelli fisici del sistema ospitante, senza passare per quelli virtuali del sistema ospite. A questo punto la TLB sfrutterà la shadow page table permettendo di accelerare gli accessi alla memoria fisica accedendovi direttamente. Questa metodologia comporta comunque un notevole sforzo dato dal fatto, che le Shadow Page Table dei VMM e le page table delle macchine virtuali devono essere mantenute coerenti fra loro: questo implica che una modifica ad un indirizzo virtuale deve essere segnalato, sia sulla Page Table che sulla Shadow Page Table.

Come esiste per i processori la tecnologia VT-x e AMD-V, esiste per la memoria la possibilità di fornire un supporto hardware per virtualizzare la MMU, sia nella nuova generazione di processori preposti alla virtualizzazione di Intel, dove prende il nome di **Extended Page Tables (EPT)**, e di AMD, dove prende il nome di **Rapid Virtualization Indexing (RVI)** o **Nested Page Tables (NPT)**. Questo supporto hardware si basa sul fornire un livello di page table che mappa direttamente gli indirizzi fisici di un sistema ospite in indirizzi fisici di un sistema ospitante, senza il bisogno che *l'hypervisor* intervenga virtualizzando la MMU via software. Questo permette di avere prestazioni migliori eliminando del tutto la gestione delle Shadow Page Table

ed avendo un supporto hardware diretto della MMU senza dover passare per *l'hypervisor*.

2.4.3 Virtualizzazione delle periferiche

Ogni macchina virtuale può scegliere tra un insieme base predefinito di periferiche virtuali emulate. Tra queste troviamo: Floppy, DVDROM, tastiera, periferiche USB, controller IDE e SCSI, scheda video, scheda di rete Ethernet, scheda audio, ecc... Le periferiche scelte da VMWare sono periferiche standard non invasive. La scelta di un insieme predefinito di periferiche standard serve per aumentare la portabilità. Anche spostando una macchina virtuale da un computer ad un altro, questa continua ancora ad utilizzare lo stesso insieme di periferiche virtuali, indipendentemente dalle periferiche fisiche installate sul computer dove è stata spostata.
Analizziamo per esempio i passi [13] che compongono la comunicazione tra una periferica fisica di rete e quella virtuale in un ambiente virtualizzato (vedi figura 2.8):

1. La periferica fisica utilizza Direct Memory Access (DMA) per scrivere i pacchetti in un buffer gestito dall'*hypervisor*.

2. La periferica fisica avvisa l'*hypervisor* attraverso un interrupt non appena terminata la copia dei dati.

3. L'*hypervisor* esamina i pacchetti per cercarne la destinazione. Una volta trovata la macchina virtuale destinataria copia i pacchetti nel suo buffer di memoria.

4. La periferica virtuale avvisa il sistema operativo della macchina virtuale a cui appartiene che sono arrivati dei dati.

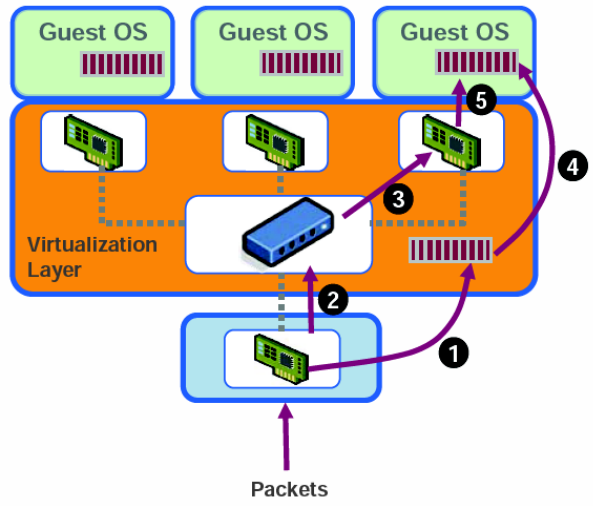

Figura 2.8: Esempio dei passi che compongono la comunicazione I/O di una scheda di rete in un ambiente virtualizzato [fonte: http://download3.vmware.com/vmworld/2006/tac0080.pdf]

2.5 VMWare vSphere

VMWare vSphere [28] è il nome della suite di programmi e funzionalità fornite da VMWare per la virtualizzazione di un infrastruttura. Tra i prodotti e le funzionalità che la compongono troviamo:

1. ESX e ESXi
2. Virtual Simmetric Multi-Processing (vSMP)
3. Virtual Machine File System (VMFS)
4. vCenter
5. VMotion
6. Distributed Resource Scheduler (DRS)
7. High Availability (HA)
8. Fault Tolerance (FT)

2.5.1 ESX ed ESXi

ESX ed ESXi [30, 27] sono entrambi *hypervisor* di tipo nativo, condividono le stesso motore di virtualizzazione e forniscono lo stesso insieme di funzionalità ad esclusione della Service Console (per un maggiore approfondimento vedi [10]). ESXi infatti non dispone di Service Console. La Service Console non è altro che una console basata su Linux, che fornisce un insieme di comandi (ridotti rispetto ad una console Linux dal momento che include solo i comandi strettamente necessari) per interagire con il vmkernel. Il principale vantaggio risiede proprio nella gestione di ESX, attraverso linea di comando e scripting, molto cara a chi è abituato a lavorare con Linux. D'altro canto, la mancanza di questo componente rende ESXi molto leggero (solo 32MB di dimensioni) e più sicuro rispetto ad ESX, eliminando potenziali attacchi passando attraverso eventuali vulnerabilità nei comandi della console.

L'*hypervisor* a sua volta è suddiviso in due componenti: Virtual Machine Monitor e vmkernel. Il vmkernel è il cuore del sistema e gestisce le principali funzioni del sistema di virtualizzazione come la pianificazione della CPU (*scheduling*), gestione della memoria ed elaborazione dei dati provenienti dai virtual switch (che vedremo in dettaglio in seguito). Ogni macchina virtuale viene gestita da un Virtual Machine Monitor separato (vedi figura 2.9), situato all'interno dell'*hypervisor*, che ne soddisfa le richieste di CPU, memoria e I/O.

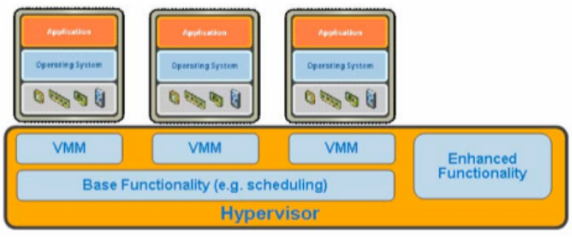

Figura 2.9: Hypervisor e VMM distinti per ogni macchina virtuale [fonte: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf]

Gestione delle risorse

ESXi ed ESX forniscono la possibilità di gestire e limitare la quantità di CPU e RAM da assegnare alle macchine virtuali ospitate al loro interno attraverso l'utilizzo di avanzati meccanismi. Questi meccanismi sono:

- **Reservation**
  Serve per riservare un quantitativo "minimo" di risorse senza le quali una macchina virtuale nemmeno si accende. Nel caso della RAM questa deve essere fisicamente presente e non può essere fornita da swap su disco.

- **Limit**
  Serve per limitare l'utilizzo di risorse. Questa opzione può essere utilizzata per creare una macchina virtuale con 8 GB di RAM per poi limitarla al solo utilizzo di 512 MB. Sorge spontaneo chiedersi come mai non creare una macchina virtuale direttamente con 512 MB di RAM. La risposta è che qualche volta potrebbe esserci imposto di creare una macchina virtuale con 8 GB di RAM. A questo punto è possibile intervenire con questo parametro per limitare lo spreco di memoria.

- **Share**
  Questo parametro è quello di "mezzo" che sta tra Reservation e Limit ed è una percentuale che indica la priorità di una macchina rispetto alle altre di accedere alle risorse. Mettiamo che due macchine virtuali siano entrambe impostate con 512 MB di RAM riservata, 1 GB di RAM limite e 1 come valore di Share. Se una macchina ESX possiede 1,5 GB di RAM ne allocherà 1 GB per soddisfare la richiesta minima di 512 MB a testa per le due macchine virtuali, come imposto dal parametro Reservation. A questo punto i restanti 512 MB di RAM disponibili verranno distribuiti equamente (entrambe le macchine hanno Share impostato a 1) tra le due macchine dandone 256 MB a testa. Se invece il parametro Share fosse impostato a 2 per una macchina e ad 1 per l'altra, avremmo

che la macchina con Share più elevato riceverebbe 2 pagine della RAM ogni 3 (quindi il 66% contro il 33% dell'altra).

I meccanismi che abbiamo visto possono essere sfruttati sia per quanto riguarda la CPU che per quanto riguarda la RAM, anche se quest'ultima possiede anche altri meccanismi più avanzati. Uno di questi è il Memory Overcommitment, e cioè la possibilità da parte di ESX ed ESXi di fornire più memoria di quanta fisicamente installata sulla macchina server. Questo implica che è possibile avviare due macchine virtuali configurate con 8 GB di RAM su una macchina server con solamente 4 GB di RAM fisica disponibile. ESX ed ESXi copriranno le necessità delle 2 macchine virtuali fornendo RAM fisica fino a che è disponibile; appena questa non sarà più sufficiente le restanti richieste verranno fornite attraverso l'utilizzo di swap su disco.

2.5.2 Virtual SMP

Attraverso Virtual SMP [27] una macchina virtuale ha la possibilità di sfruttare più processori fisici (vedi figura 2.10) per portare a termine compiti particolarmente pesanti. Con l'ultima versione di ESX (ESX 4) si è raggiunta la possibilità di assegnare fino ad 8 processori virtuali ad una macchina virtuale [10]. Partendo da ESX 2.1 è inoltre possibile sfruttare l'Hyper-Threading [21]. L'Hyper-Threading[2] è una tecnologia Intel, che permette di sfruttare un processore per l'esecuzione contemporanea di due thread diversi, duplicando al suo interno alcune delle unità di elaborazione maggiormente utilizzate, al fine di poter eseguire simultaneamente alcune operazioni, grazie a tecniche di *multithreading*. Un singolo core è, quindi, in grado di gestire due thread in contemporanea; quando le istruzioni di un thread rimangono bloccate nella pipeline il processore procede ad elaborare un secondo thread al fine di mantenere le unità di elaborazione sempre attive. Questo può portare ad un incremento delle prestazioni che varia però a seconda della quantità di lavoro e dalla sua tipologia. In certi casi di carichi molto pesanti

[2] http://www.intel.com/technology/platform-technology/hyper-threading/

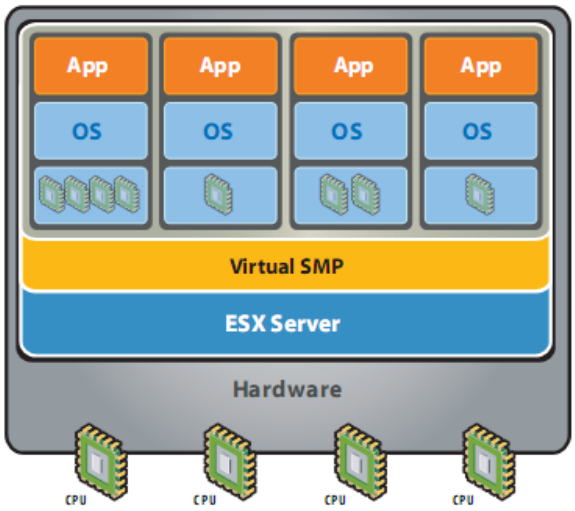

Figura 2.10: vSMP: macchine virtuali a cui possono essere assegnati più di un processore virtuale [fonte: http://www.vmware.com/pdf/vmware_infrastructure_wp.pdf]

le prestazioni potrebbero anche diminuire. Infatti, i due thread potrebbero finire con il competere sulle risorse condivise, come la cache L2, la cache L3 e le unità funzionali, come l'Arithmetic Logic Unit (ALU), finendo per rallentarsi a vicenda [22]. Le prestazioni di un processore con Hyper-Treading abilitato non saranno mai superiori a quelle di un sistema con due processori. Abilitando l'Hyper-Threading vSphere permetterà di vedere un quantitativo di processori logici da assegnare alle macchine virtuali pari al doppio di quelli fisici.

Una delle controindicazioni, dovuta all'SMP, è che per ogni macchina virtuale deve essere allocata una parte di RAM aggiuntiva per la gestione da parte di ESX [10]. L'*overhead* in questione è direttamente proporzionale alla quantità di RAM assegnata alla macchina virtuale e al numero di processori virtuali assegnati; nello specifico sale man mano che o aumenta la RAM o aumentano i processori virtuali.

2.5.3 Virtual Machine File System

Una macchina virtuale dal punto di vista del filesystem viene percepita come un insieme di file: disco, swap, configurazione ecc... Questi file possono essere allocati localmente sul server ESX oppure in maniera logicamente centralizzata utilizzando Storage Area Network e Network Attached Storage. Quando lo spazio dove risiedono le macchine virtuali è centralizzato c'è però bisogno di un meccanismo per gestire la concorrenza. Questo avviene attraverso l'utilizzo di un filesystem concorrente come VMFS [31, 27]. Attraverso questo filesystem, più server ESX possono accedere concorrentemente alle risorse presenti all'interno di uno stesso spazio di memorizzazione condiviso (vedi figura 2.11). Un sistema di locking sui metadati permette

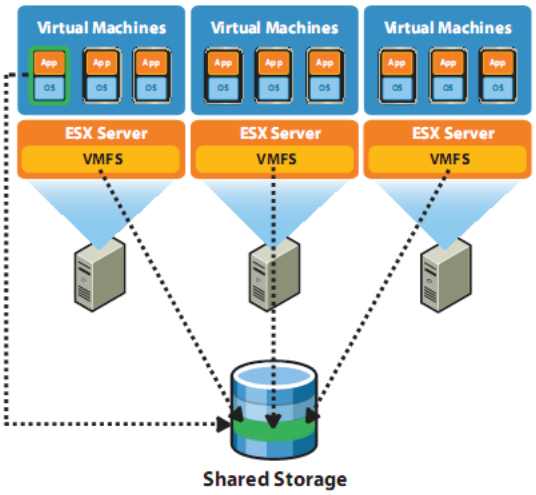

Figura 2.11: Server ESX che accedono concorrentemente a macchine virtuali situate all'interno dello stesso spazio di memorizzazione formattato con VMFS [fonte: http://www.vmware.com/pdf/vmware_infrastructure_wp.pdf]

ad un solo server ESX di avviare una macchina virtuale; in caso di guasto di una macchina virtuale viene gestito in automatico un sistema di rilascio dei lock per sbloccare le risorse. VMFS permette la creazione di file fino a 2TB e può utilizzare blocchi da 1 a 8 MB. I blocchi sono di grosse dimensioni per favorire l'I/O dal momento che un filesystem utilizzato per memorizzare

macchine virtuali contiene pochi file di grosse dimensioni. VMFS è la base di partenza per funzionalità più avanzate come VMotion e DRS che vedremo tra poco. VMFS fornisce anche le funzionalità e i meccanismi di consistenza e recovery tipici di un file system di classe enterprise come, per esempio, funzionalità di *journaling* distribuito.

2.5.4 vCenter

Attraverso vCenter[32, 27] (vedi figura 2.12) è possibile gestire in maniera centralizzata tanti aspetti relativi all'infrastruttura VMWare. Oltre al fatto che vCenter è una componente obbligatoria per poter usufruire delle più avanzate funzionalità di ESX come Fault Tolerance, DRS, VMotion, Storage VMotion (che altrimenti sarebbero disabilitate) ci permette anche di:

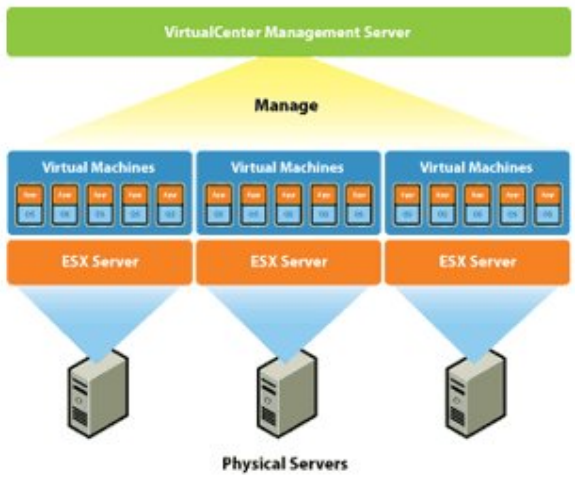

Figura 2.12: vCenter e gestione centralizzata [fonte: `http://www.vmware.com/pdf/vmware_infrastructure_wp.pdf`]

1. Gestire in maniera centralizzata tutti i componenti facenti parte del nostro sistema di virtualizzazione ed avere un sistema centralizzato di autenticazione.

2. Monitorare attraverso grafici l'utilizzo delle risorse (come CPU, disco, memoria) delle varie macchine virtuali e dei server ESX.

3. Pianificare operazioni in determinati momenti o allertare gli amministratori in caso si verifichino situazioni anomale che si vogliono gestire.

4. Aggiungere velocemente nuove macchine virtuali utilizzando *template*.

5. Creare cluster di server ESX (vedi figura 2.13) dove le risorse globali del cluster sono la somma delle risorse di ogni componente. Ogni macchina virtuale appartenente ad un cluster avrà poi a disposizione tutte le sue risorse.

6. Creare insiemi di risorse. È possibile specificare un limite massimo di utilizzo di risorse per ogni insieme. Le macchine virtuali all'interno di un insieme (*pool*) saranno costrette a suddividersi le risorse che gli sono state assegnate.

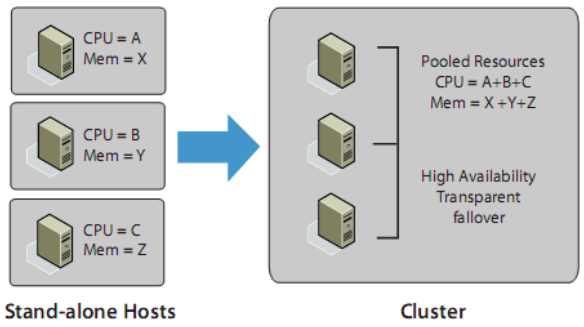

Figura 2.13: Cluster di server ESX [fonte: http://www.vmware.com/pdf/vmware_infrastructure_wp.pdf]

2.5.5 VMotion, Storage VMotion

Con VMotion [33, 27] abbiamo la possibilità di fare migrare una macchina virtuale da un server ESX ad un altro "a caldo" (cioè senza dover spegnere il

sistema), senza dovere interrompere la sua esecuzione. Questa funzionalità è molto utile, perché ci permette in caso di guasto o spegnimento di un Server ESX, di spostare le sue macchine virtuali su altri server ESX. Inoltre VMotion sta alla base anche di alcune tecniche di bilanciamento di carico; infatti se i nostri server sono troppo sotto sforzo è possibile intervenire spostando alcune macchine virtuali su server meno carichi di lavoro o addirittura su nuovi nodi. Per potere sfruttare VMotion sono necessarie tre cose [10]:

1. Un architettura di memorizzazione condivisa (Network Attached Storage, Storage Area Network) accessibile dai server ESX coinvolti nell'operazione di VMotion, su cui risiederà l'immagine della macchina virtuale.

2. Una rete veloce (almeno Gigabit) che collega i server ESX e che gli permetta il trasferimento veloce dello stato (registri, memoria...) della macchina virtuale da spostare.

3. La medesima configurazione di rete su entrambe le macchine ESX: vSwitch, Port Group, VLAN, ecc...

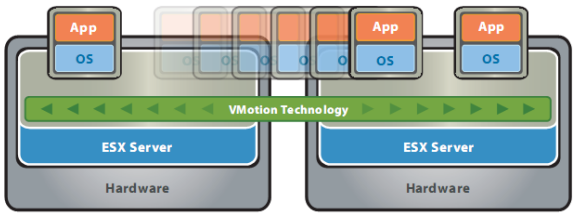

Figura 2.14: Un operazione di VMotion tra due server ESX [fonte: `http://www.rareyroth.com/virtualization.htm`]

Esempio di trasferimento da un server ESX `A` ad un server `B` a caldo:

1. Si inizia a spostare lo stato della macchina (e non la macchina che invece non viene spostata) da `A` a `B`. `A` intanto comincia a salvarsi le modifiche che avvengono alla sua memoria e nel frattempo continua a servire eventuali richieste.

2. Non appena lo stato è stato completamente copiato su `B` viene fermata l'esecuzione della macchina virtuale su `A` e viene ripristinato completamente il suo stato su `B`. L'operazione eseguita in questa maniera si riduce ad un periodo di disservizio (*downtime*) della macchina virtuale di soli pochi secondi su una rete Gigabit (il tempo che ci vuole ad effettuare il ripristino delle operazioni compiute sulla macchina `A` durante il trasferimento dello stato su `B`).

3. ESX `B` utilizza RARP per avvisare lo switch fisico che lo collega ad `A` che la porta corretta su cui inviare i dati ora è la sua e non più quella che portava ad `A`.

4. La macchina riprende la sua esecuzione su `B`. Questo è possibile perché si ritrova con la stessa configurazione di vSwitch, port group e VLAN. Inoltre la macchina mantiene lo stesso MAC Address e lo stesso indirizzo IP per poter continuare a servire le richieste che erano in atto su `A`.

Oltre a VMotion esiste una variante chiamata Storage VMotion [34], che funziona in maniera simile e permette di trasferire a caldo una macchina virtuale da un *datastore* ad un altro. Questa funzionalità è presente a partire da ESX 3.5 ed è più pesante di un operazione di VMotion, dal momento che deve essere spostata l'intera macchina virtuale (non solo lo stato come prima), da uno spazio di memorizzazione all'altro. Un operazione di Storage VMotion può essere esemplificata in quattro fasi:

1. Prima di iniziare la copia verso la destinazione, la macchina virtuale sorgente attiva una registrazione su un log (REDO log) delle modifiche a cui viene sottoposta durante quest'ultima.

2. Inizia la copia della macchina sorgente verso la destinazione rimanendo comunque accessibile (niente disservizio quindi) e registrando ogni cambiamento che avviene sul REDO log visto al punto 1.

3. Al termine della copia della macchina sorgente verso la macchina destinazione, è possibile che questa sia disallineata da quella sorgente, che durante la copia potrebbe aver subito modifiche. A questo punto si procede con la sincronizzazione della macchina destinazione, copiando le modifiche registrate nel REDO log durante la fase 2. Inoltre durante questa copia la macchina sorgente riprende a registrare le modifiche che avvengono su di lei su un nuovo REDO log. Questa terza operazione va ripetuta fino a quando le modifiche da effettuare sulla destinazione sono così poche che possono essere effettuate quasi istantaneamente (solitamente qualche secondo).

4. Nell'ultima fase si disattiva la macchina sorgente, così da non potere subire ulteriori modifiche, e si allineano le due macchine scrivendo sulla macchina destinazione le modifiche rimanenti. Fatto ciò, è possibile concludere l'operazione di Storage VMotion ed attivare la macchina destinazione, che si prenderà carico delle operazioni che erano in corso sulla macchina sorgente, che a questo punto viene cancellata; il tutto avviene senza generare disservizi.

2.5.6 Distributed Resource Scheduler

Il DRS [35, 27] permette di ottimizzare l'utilizzo delle risorse tra vari server fisici che utilizzano ESX, ma necessita, come ogni altra funzionalità avanzata di VMWare, della presenza di vCenter. L'ottimizzazione dipende innanzi tutto da un insieme di regole definite dall'amministratore che può decidere quali macchine virtuali devono avere una maggiore priorità rispetto ad altre e quante risorse minime e massime una macchina può avere. Il DRS cerca di bilanciare il carico di lavoro sui server ESX esistenti in maniera adattiva. Nel caso un server sia scarico ed un altro no, il DRS attraverso VMotion sposta automaticamente alcune macchine virtuali sul server più scarico così da bilanciarle (vedi figura 2.16). In caso che tutti i server siano carichi è possibile perfino accendere ed aggiungere automaticamente a caldo un nuo-

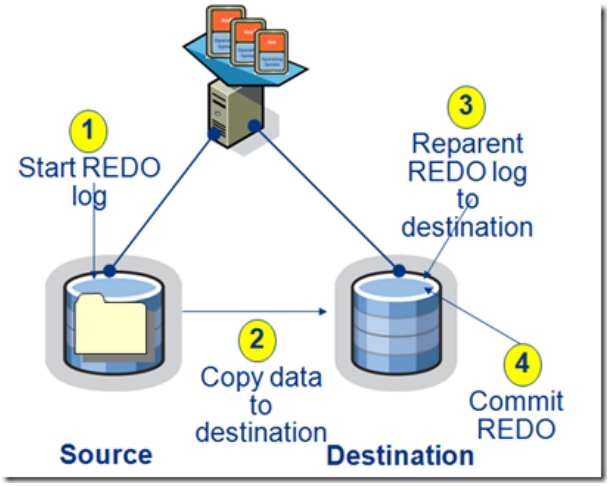

Figura 2.15: Esempio dei 4 passi da effettuare per un operazione di Storage VMotion tra due spazi di memorizzazione [fonte: http://virtualgeek.typepad.com/virtual_geek/2008/12/real-world-experiences-using-storage-vmotion.html]

vo server ESX, che nel frattempo era rimasto in attesa in modalità passiva. Questa operazione avviene anche in maniera inversa e cioè, se i server cominciano ad essere troppo scarichi, quelli in eccesso vengono spenti e le risorse ridistribuite sugli altri contribuendo a non utilizzare risorse in eccesso.
DRS permette inoltre di espandere la potenza di calcolo di un centro aggiungendo nuovi server le cui risorse vengono subito messe a disposizione.

2.5.7 High Availability

L'High Availability [36, 27] in VMWare è fornita dall'unione di DRS, VMotion e VMFS. In caso di guasto di un server o di un sistema operativo, il DRS ha il compito di ridistribuire automaticamente tutte le macchine virtuali che stava servendo sugli altri server attraverso operazioni di VMotion (vedi figura 2.17). Per rilevare e gestire automaticamente eventuali guasti viene utilizzato un servizio di monitoraggio dei server e delle macchine virtuali di tipo *Heartbeat*.
È importante notare però che prima che gli altri server ESX possano prendersi

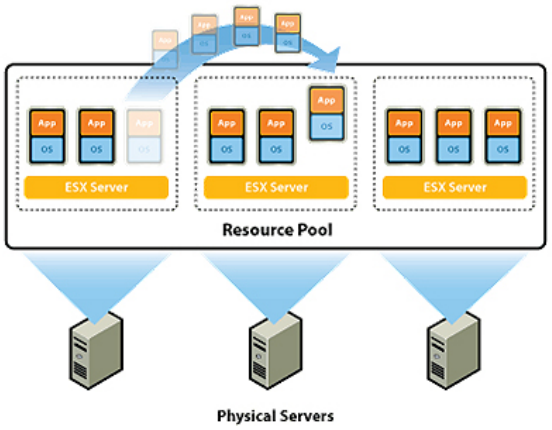

Figura 2.16: Un operazione di ridistribuzione di carico [fonte: http://www.vmware.com/fr/solutions/consolidation/automate.html]

carico delle macchine virtuali rimaste orfane, queste generano un disservizio per il periodo necessario al loro riavvio.

2.5.8 Fault Tolerance

Come visto sopra DRS e VMotion vengono utilizzate da VMWare per fornire High Availability. Nonostante questo l'HA di VMWare non riesce ad evitare periodi di disservizio, pur se brevi, in caso di guasti di un server ESX. Per certe aziende questi brevi periodi (che si aggirano nell'ordine di 5-10 minuti, anche se è dipendente dal numero di macchine virtuali da far ripartire) però non sono tollerabili e questo ha portato VMWare ad aggiungere un ulteriore funzionalità chiamata Fault Tolerance [37]. VMWare Fault Tolerance si ripropone di evitare completamente periodi di disservizio, facendo si che ogni macchina virtuale crei una copia clone di se stessa su un server ESX, differente da quello su cui sta eseguendo in quel momento, così da evitare periodi di disservizio in caso di guasto di un nodo ESX (vedi figura 2.18). La macchina `clone` è in uno stato di completo isolamento e viene continuamente sincronizzata ad ogni aggiornamento di quella `originale`. A questo punto si possono verificare tre situazioni: cade il nodo ESX dove è

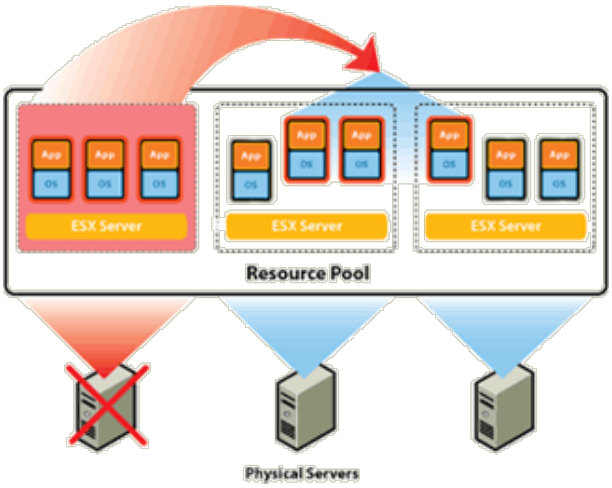

Figura 2.17: High Availability ottenuta attraverso l'utilizzo di DRS, VMotion e VMFS [fonte: http://www.majentasolutions.com/solutions/storage-solutions/vmware/vmware-virtualcenter/vmware-high-availability/]

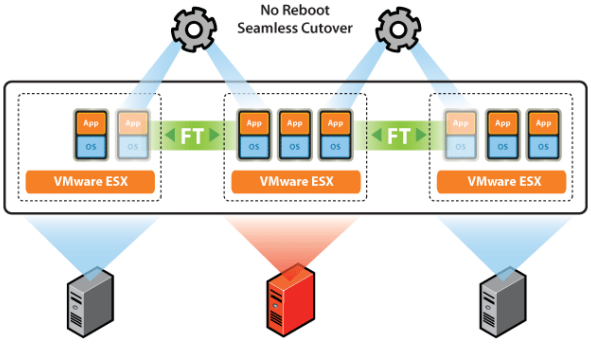

Figura 2.18: Fault Tolerance [fonte: http://www.vmguru.nl/wordpress/2009/01/new-vmware-products-in-2009/]

situata la macchina `originale`, cade il nodo ESX dove è situata la macchina `clone`, oppure cadono entrambi. Ovviamente, il caso in cui cadano entrambi i nodi, è il caso sfortunato che porterà ad un disservizio. Se invece dovesse cadere il nodo dove è situata la macchina `originale`, la macchina `clone` viene subito attivata per sostituirla e a sua volta crea una nuova copia di se stessa su un altro nodo ESX per proteggersi da ulteriori guasti. Questo stesso ragionamento si applica in caso che a cadere sia la macchina `clone`, solo che questa volta sarà la macchina `originale` a creare un altra copia su un nuovo nodo ESX.

2.6 VMWare e la rete

All'interno di VMWare avviene tutta una gestione software della rete dove il funzionamento logico è molto simile a quanto avviene in una normale rete fisica. La progettazione della rete insieme a quello dello spazio di memorizzazione è uno degli aspetti più critici di un architettura virtualizzata. Uno dei componenti principali su cui è stata poi costruita la rete software di VMWare è il **Virtual Switch (vSwitch)** (vedi figura 2.19).

2.6.1 Virtual Switch

Un vSwitch possiede molte caratteristiche in comune con quelle di uno switch fisico [10, 26]:

- Gestisce traffico di rete di livello Data Link.
- Mantiene tabelle di filtering con i vari MAC address.
- Supporta le VLAN ed è capace di effettuare il *trunking* utilizzando il protocollo IEEE 802.1Q.

Ma ha anche alcune caratteristiche che lo differenziano:

- Non supporta il Dynamic Trunking Protocol (DTP) o il Port Aggregation Protocol (PAgP).

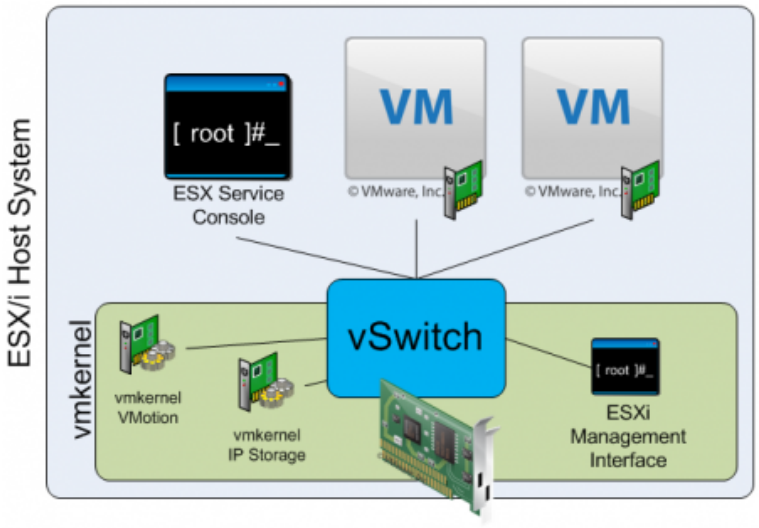

Figura 2.19: Esempio di un vSwitch e delle varie tipologie di traffico che supporta [fonte: http://kensvirtualreality.wordpress.com/2009/03/29/the-great-vswitch-debate-part-1/]

- Non utilizza lo Spanning Tree Protocol (STP) dal momento che non è permesso collegare vSwitch tra loro direttamente evitando in questo modo possibili cicli.

Un vSwitch può gestire varie tipologie di traffico di rete differente, basta però che prima di tutto vengano create delle porte (`port`) o dei gruppi di porte (`port group`) al suo interno. Esistono tre tipologie di porte [10, 26] creabili all'interno di un vSwitch e queste sono:

1. **Service Console Port**:
   Una porta specializzata configurata con un indirizzo IP che permette l'accesso alla Console di Servizio di ESX.

2. **VMKernel Port**:
   Una porta specializzata configurata con un indirizzo IP che gestisce:

   - Traffico generato da VMotion.
   - Traffico iSCSI e NFS da e verso datastore remoti.

- Traffico di gestione nel caso di ESXi (la rete detta in gergo: *"rete di management"*).

3. **Virtual Machine Port Group**:
   Un gruppo di porte che condividono la stessa configurazione e permettono il passaggio del traffico delle macchine virtuali verso altre macchine virtuali o la rete fisica.

Un host con ESX può arrivare ad avere fino a ben 127 vSwitch ed è buona norma utilizzarli per separare diverse tipologie di traffico di rete. Come accennato sopra, i vSwitch non possono essere collegati tra loro direttamente (vedi figura 2.20); questo significa che se li si vuole mettere in comunicazione

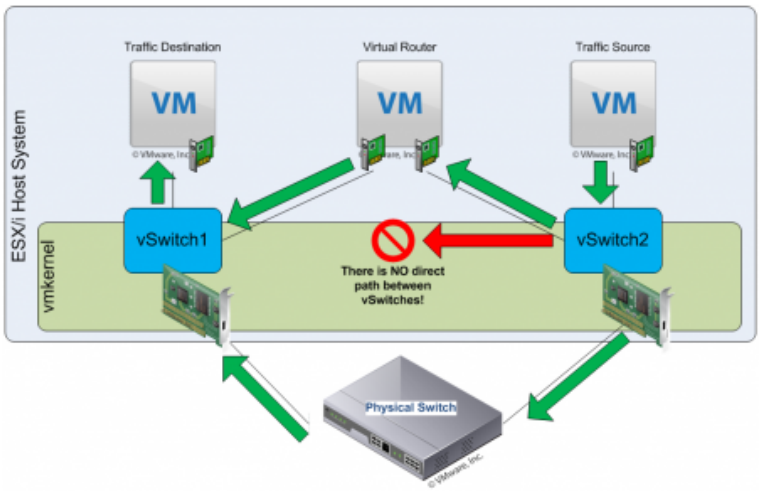

Figura 2.20: Impossibilità di collegare direttamente tra loro due vSwitch [fonte: http://kensvirtualreality.wordpress.com/2009/03/29/the-great-vswitch-debate-part-1/]

è necessario creare una macchina virtuale che faccia da router. Un vSwitch è anche colui che deve garantire la tolleranza ai guasti. In una rete fisica la tolleranza ai guasti viene raggiunta replicandone i componenti come gli switch e le schede di rete. Per quanto riguarda le schede di rete queste devono almeno essere due e solitamente accoppiate in modalità di Link Aggregation per fornire il massimo della banda. Se abbiamo, per esempio, 4 server fisici ed ognuno dispone di due schede di rete, ci troveremo ad avere ben presto

8 schede di rete. Al contrario in una rete virtuale è possibile utilizzare un vSwitch, 4 schede di rete virtuali e solamente 2 schede di rete fisiche. Questo non solo ci permette di raggiungere lo stesso livello di tolleranza ai guasti con un numero minore di componenti, ma in più, ci permette di semplificare l'architettura di rete. Questo beneficio ottenuto attraverso il consolidamento a livello di rete (vedi figura 2.21) viene comunque mitigato da una minor banda disponibile per i 4 server.

Inizialmente abbiamo accennato al fatto che i vSwitch supportano il proto-

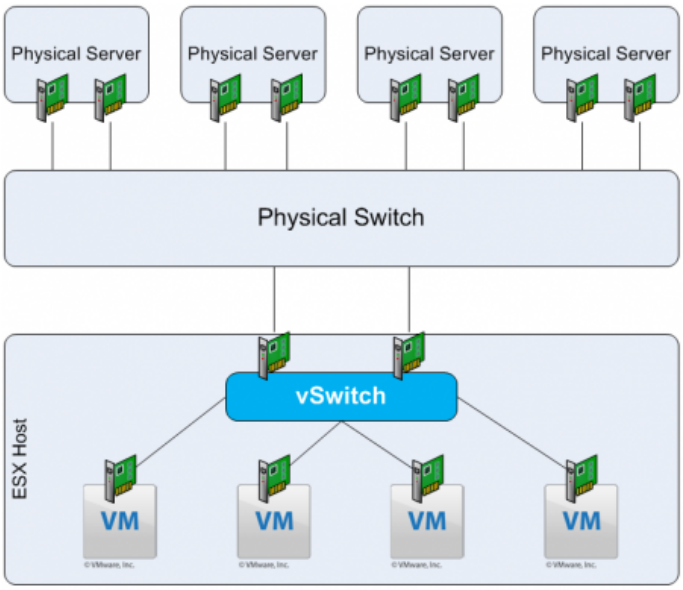

Figura 2.21: Consolidamento di rete ottenuto sfruttando la virtualizzazione [fonte: http://kensvirtualreality.wordpress.com/2009/03/29/the-great-vswitch-debate-part-1/]

collo IEEE 802.1Q[3], che riguarda le VLAN. Andiamo quindi a scoprire ora cosa sono e qual'è il loro funzionamento.

2.6.2 VLAN

Esistono varie possibili definizioni di VLAN (vedi figura 2.22). Quella che preferisco è la seguente [14]:

[3] http://www.ieee802.org/1/pages/802.1Q.html

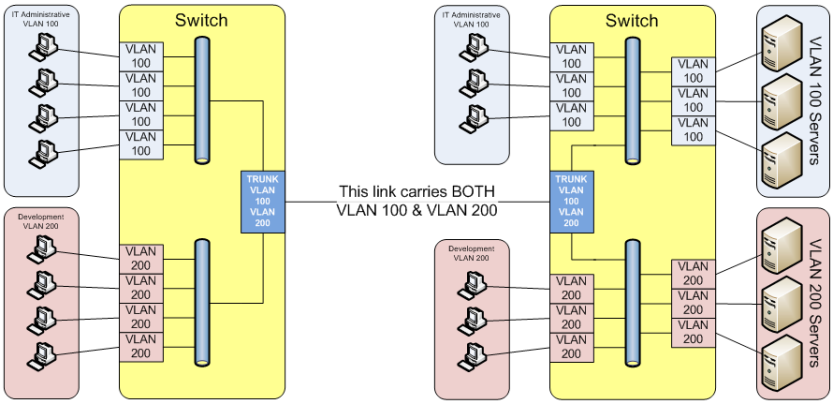

Figura 2.22: Esempio di separazione in reti logiche attraverso l'utilizzo di VLAN [fonte: http://kensvirtualreality.wordpress.com/2009/03/29/the-great-vswitch-debate-part-1/]

> *"Una VLAN è un sottoinsieme logico di una LAN, su questa fisicamente insistente ma da essa logicamente separata, costituente un dominio di collisione a sé stante"*

In termini pratici, le porte di uno switch assegnate ad una VLAN, costituiscono un dominio di broadcast logicamente separato da quello di default dello switch (che, normalmente, si estende a tutte le sue porte); i pacchetti ricevuti da una porta appartenente ad una data VLAN sono visibili solo all'interno della VLAN stessa (N.B.: da non confondere con il *port trunking* che vedremo in seguito), siano essi unicast, multicast o broadcast. Le VLAN vengono utilizzate per:

- **Creare sottoreti logiche**: Permettere la comunicazione tra sottoinsiemi di host appartenenti a LAN fisiche differenti come se fossero logicamente appartenenti alla stessa LAN fisica (ma che in realtà è una VLAN).
- **Ridurre il dominio di broadcast di una LAN**: Dal momento che una VLAN non inoltra i pacchetti broadcast sulle porte che non le competono, riduce il dominio di collisione dei pacchetti braodcast evitando inutile consumo di banda (un problema tipico è il *"broadcast storming"*[4]).

VLAN e IEEE 802.1Q

In VMWare lo standard utilizzato per creare VLAN [5] è l'IEEE 802.1Q che definisce l'estensione di un frame ethernet per mezzo di un *tag* (da cui poi deriva la definizione di traffico che possiede il tag, detto *tagged*, ed il traffico che non lo possiede, detto *untagged*) di 32 bit (vedi figura 2.23), contenente informazioni sulla VLAN a cui il frame stesso appartiene. Il tag è diviso in più

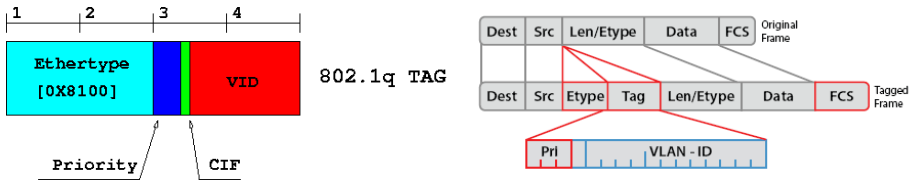

Figura 2.23: Tag VLAN a sinistra e suo inserimento all'interno di un frame ethernet [fonte: http://novalis.mib.infn.it/site/media/tech/vlan.html]

campi [14]. Abbiamo quello relativo all'`Ethertype` (2 byte) con valore fisso 0x8100 che identifica lo standard IEEE 802.1Q, il `Priority Field` (3 bit), il `Canonical Format Indicator` ([CIF] 1 bit) ed infine il `VLAN Identifier` ([VLAN ID]). I 3 bit del campo della priorità permettono di gestire QoS all'interno della propria rete. È infatti possibile assegnare una priorità diversa al traffico delle VLAN per favorire quelli con priorità più elevata a livello di switch. Il campo del VLAN ID invece è quello che identifica a che VLAN appartiene il frame ethernet. I 12 bit che lo compongono permettono di avere un totale di 4096 possibili VLAN coesistenti sulla stessa LAN.
Non tutti gli switch sono in grado di gestire il traffico 802.1Q, ma quelli che ci riescono permettono di assegnare ogni porta dello switch ad un `Port VLAN Identifier` ([PVID]) in modalità `untagged` o `tagged`. Il PVID indica a quale VLAN deve appartenere il traffico in entrata o in uscita su quella porta dello switch, mentre le modalità `tagged` e `untagged` permettono di definire che comportamento la porta dello switch deve assumere quando i pacchetti la attraversano.
Più approfonditamente si seguono queste regole (vedi figura 2.24): quando un pacchetto `untagged` (privo cioè di tag), esce da una porta `untagged` rimane

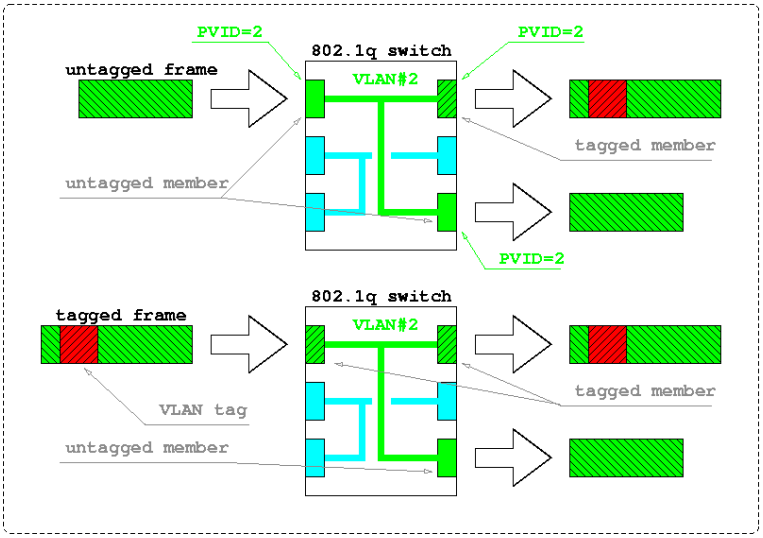

Figura 2.24: Regole di tagging e untagging [fonte: http://novalis.mib.infn.it/site/media/tech/vlan.html]

`untagged`, mentre un pacchetto `tagged` viene privato del tag e trasformato in un pacchetto `untagged`. Il comportamento di una porta `tagged` è invece opposto: se un pacchetto `untagged` esce da una porta `tagged` al frame ethernet viene aggiunto un tag con VLAN ID uguale al PVID la cui porta appartiene, mentre un pacchetto `tagged` viene lasciato inalterato, ed in particolare il VLAN ID originale presente nel tag (che può anche differire dal PVID della porta in questione) non viene cambiato. Quando due switch vengono connessi tra di loro attraverso due porte `tagged` si dice che è stato creato un VLAN Trunk (detto anche `trunking`). Il `trunking` permette al traffico di VLAN differenti, appartenenti al primo switch, di raggiungere il secondo switch.

VLAN e vSwitch

Le possibilità che ci vengono offerte da parte di uno switch virtuale in VMWare per gestire le VLAN sono tre [16, 3](vedi figura 2.25):

1. **Virtual Guest Tagging (VGT)**
   Il traffico passa attraverso il virtual switch ed arriva alle macchine virtuali senza che vengano rimossi i tag. Il compito di gestire i pacchetti

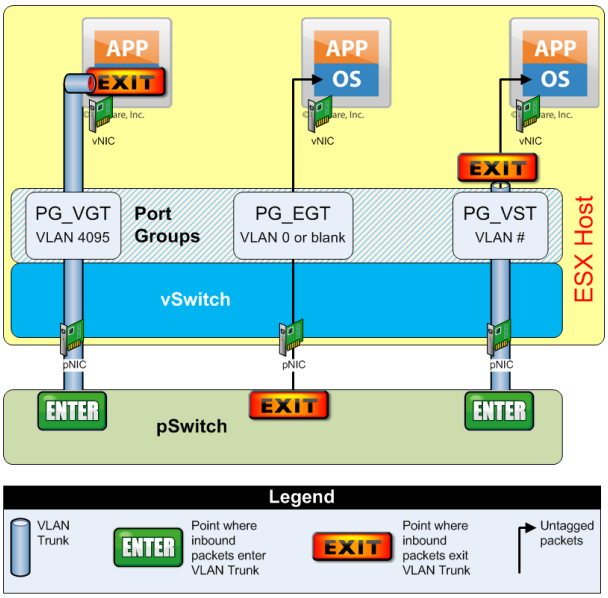

Figura 2.25: Le tre modalità con cui un vSwitch può gestire le VLAN [fonte: `http://kensvirtualreality.wordpress.com/2009/03/29/the-great-vswitch-debate-part-1/`]

taggati viene quindi lasciato come responsabilità al sistema operativo delle macchine virtuali. Per abilitare la modalità VGT è necessario utilizzare nel vSwitch `VLAN ID 4095`.

2. **External Switch Tagging (EST)**
   Il traffico e i tag delle VLAN non vengono gestiti dal vSwitch ma esternamente quando entrano o escono dalla porta di uno switch fisico. Il traffico che raggiunge il vSwitch e le macchine virtuali dietro di esso è senza tag. Per abilitare la modalità EST è necessario utilizzare nel vSwitch `VLAN ID 0`.

3. **Virtual Switch Tagging (VST)**
   Il vSwitch gestisce il traffico VLAN. Lascia passare in ricezione, dall'esterno verso l'interno dove sono situate le macchine virtuali, solamente il traffico che possiede all'interno del tag il VLAN ID che gli è stato associato; a tutti i pacchetti che vengono lasciati passare viene rimosso però il tag. Trasmette invece il traffico delle macchine virtuali

che stanno dietro ad esso, quindi dall'interno verso l'esterno, impostando come tag il proprio VLAN ID. Questo implica che le comunicazioni tra macchine virtuali e vSwitch avviene attraverso traffico senza tag (come appena detto infatti il vSwitch rimuove i tag dai pacchetti in entrata e lo rimette nei pacchetti in uscita). Per abilitare la modalità VST è necessario utilizzare nel vSwitch un `VLAN ID compreso tra 1 e 4094`, fornendo la possibilità di creare fino a 4094 diverse VLAN.

2.6.3 Link Aggregation

Link Aggregation viene utilizzato per fornire ad una macchina una maggiore tolleranza ai guasti e bilanciamento di carico facendo lavorare insieme una o più schede di rete. Link Aggregation è anche chiamato [19]:

- **NIC Bonding** (Mondo Linux)
- **NIC Teaming** (Mondo Windows)
- **Port Trunking** (Mondo Hardware)

Esistono varie modalità di aggregazione su cui impostare le schede di rete di una macchina, che vanno suddivise tra modalità statiche o dinamiche. L'unica modalità dinamica che si andrà ad esaminare sarà quella fornita dal portocollo IEEE 802.3ad[4] che permette un utilizzo del mezzo di trasmissione più intelligente rispetto le controparti statiche (ricordiamo però che IEEE 802.3ad può essere anche configurato in maniera statica). Importante notare come schede aggregate tra loro risultino all'esterno come un unica interfaccia di rete logica (`bond0` per esempio in Linux), con il medesimo MAC address [7]. Le principali modalità di funzionamento sono:

Modalità 0 detta "balance-rr" (statica)

Politica Round Robin: i pacchetti vengono trasmessi in ordine sequenziale dalla prima interfaccia fino all'ultima in maniera ciclica. I vantaggi che si

[4] http://www.ieee802.org/3/ad/public/index.html

ottengono da questa modalità sono bilanciamento di carico e tolleranza ai guasti.

Modalità 1 detta "active-backup" (statica)

Politica Active-backup: solamente un interfaccia di rete alla volta è attiva, le altre restano passive ed entrano in gioco (una per volta), solo in caso di guasto di quella attiva. Questa modalità nonostante fornisca tolleranza ai guasti è particolarmente sconsigliata soprattutto all'aumentare delle schede di rete in backup, poiché vengono lasciate inutilizzate.

Modalità 2 detta "balance-xor" (statica)

Politica XOR: trasmissione basata sulla formula [(source MAC address XOR destination MAC address) modulo numero di interfacce]. Ogni comunicazione avente un determinato indirizzo MAC di destinazione continuerà a comunicare sempre con la stessa interfaccia di rete. Il source MAC address è sempre lo stesso ed è quello d'aggregazione. Anche se ipoteticamente si può ottenere ugual bilanciamento su ogni scheda, è pure possibile avere un caso pessimo dove tutto il traffico passa solo su una scheda. In generale all'aumentare delle schede diminuirà la possibilità che ciò accada, ma resta di fatto che il bilanciamento che si ottiene non sarà quasi mai distribuito in ugual misura su ogni scheda. Questa modalità permette bilanciamento di carico e tolleranza ai guasti.

Modalità 3 detta "broadcast" (statica)

Politica Broadcast: ogni pacchetto viene trasmesso su tutte le interfacce di rete. Questa modalità fornisce tolleranza ai guasti, ma è sconsigliabile perché all'aumentare delle schede genera molto traffico di rete.

Modalità 4 detta "IEEE 802.3ad" (dinamica o statica)

IEEE 802.3ad è l'unica modalità tra quelle citate che possiede una versione dinamica. Nella sua versione dinamica il protocollo IEEE 802.3ad sfrutta il Link Aggregation Control Protocol (LACP) per capire in maniera automatica quali porte dello switch devono essere aggregate tra loro in gruppi d'aggregazione a seconda della tipologia delle schede di rete (half/full duplex) e della loro velocità (10/100/1000). Verranno quindi a crearsi dei gruppi di aggregazione a cui viene associato un `Active Aggregation ID`. Questi gruppi verranno poi considerati dallo switch come un unico mezzo di trasmissione logico. È necessario ribadire che per essere aggregate insieme le schede di rete devono avere la stessa velocità e la stessa tipologia di esecuzione (half/full duplex).
Nella sua versione statica (l'unica spportata da VMWare ad oggi) non viene utilizzato il LACP per capire quali schede aggregare tra loro, ma queste devono essere impostate manualmente. L'impostazione statica permette, tra le altre cose, l'utilizzo di schede di velocità diversa tra loro.
La differenza sostanziale tra IEEE 802.3ad (sia statico che dinamico), rispetto la modalità 0 vista sopra, è che qui ogni connessione una volta stabilita continua a spedire i pacchetti utilizzando sempre la stessa interfaccia di rete. Quando si utilizza la modalità IEEE 802.3ad è importante capire che il bilanciamento di carico non si riferisce ad un bilanciamento della banda tra le schede aggregate tra loro, ma ad un bilanciamento delle connessioni. Questo può portare ad un bilanciamento della banda, ma può anche, nel caso pessimo, fare passare tutte le connessioni per la stessa scheda e quindi generare di fatto nessun bilanciamento. È per questo che è molto importante fare in modo che la politica di bilanciamento scelta sia la più corretta per le proprie esigenze.

Tra i vantaggi forniti dal Link Aggregation troviamo [6, 20]:

- **Bilanciamento**:

Possibilità di bilanciare le connessioni tra le varie schede di rete.

- **Tolleranza ai guasti**:
  Non abbiamo più un singolo punto di rottura per quanto riguarda le schede di rete.

Tra i possibili contro:

- **Necessità di switch sofisticati**:
  Lo switch deve supportare il protocollo IEEE 802.3ad ed inoltre diventa il singolo punto di rottura. Nasce quindi la necessità di separare il traffico di un'aggregazione di schede su più switch fisici, che però devono saper gestire la cosa. Nortel, Cisco e Juniper per esempio forniscono apparecchiature di rete in grado di permettere la suddivisione di più porte di un gruppo d'aggrgazione su più switch fisici e di gestirne il traffico.

- **Bilancimento minimo o assente**:
  Potenziale mancanza di bilanciamento dovuta a poche connessioni o distribuzione molto asimmetrica del carico di lavoro.

- **Stessa velocità per le schede di rete**:
  Le schede di rete, che vogliono lavorare insieme attraverso aggregazione dinamica con LACP, devono essere tutte full duplex e della medesima velocità. N.B.: In modalità statica è possibile invece aggregare tra loro anche schede di rete con velocità diversa.

2.6.4 VMWare e NIC Teaming

In VMWare è possibile attuare delle politiche di Link Aggregation; seguendo la terminologia di VMWare il Link Aggregation viene chiamato NIC-Teaming [20] (vedi figura 2.26) e nello specifico quella supportata è la modalità IEEE 802.3ad statica. Le politiche di bilanciamento di carico da impostare nel vSwitch sono tre (tutte di natura statica) [26]:

1. **Bilanciamento basato sulla porta del vSwitch** (vSwitch port-based load balancing).
2. **Bilanciamento basato sul MAC address sorgente** (Source MAC-based load balancing).
3. **Bilanciamento basato sull'hash dell'indirizzo IP** (IP hash-based load balancing).

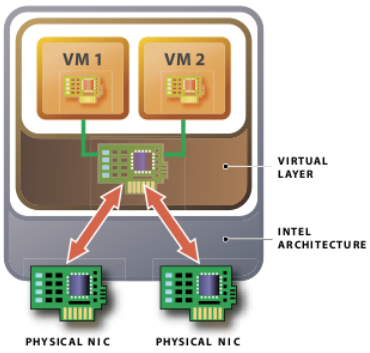

Figura 2.26: Un esempio di NIC Teaming [fonte: [20]]

Bilanciamento basato sulla porta del vSwitch

In questa modalità viene utilizzato un algoritmo, che cerca di stabilire un uguale numero di connessioni tra le porte d'entrata in utilizzo del vSwitch e le schede di rete fisiche. Ogni porta d'ingresso, una volta assegnata ad una scheda di rete, continuerà ad utilizzarla per tutta la sua vita a meno di guasti e failover su altra scheda di rete disponibile. Nel caso che ogni macchina virtuale possieda solamente una interfaccia di rete virtuale, questa modalità non permette bilanciamento di carico, visto che come abbiamo detto il traffico passerà solamente per una scheda di rete fisica. Per poter avere bilanciamento di carico in questa modalità, è necessario che una macchina virtuale possieda due o più interfacce di rete virtuali (che quindi utilizzano due differenti porte del vSwitch), e che queste vengano assegnate a schede di

rete differenti, prestando attenzione al fatto che potenzialmente potrebbero essere tutte assegnate alla stessa scheda di rete, negando il bilanciamento di carico.
Un altro problema è dato dal fatto che se abbiamo più interfacce di rete fisiche rispetto a quelle virtuali (e quindi utilizziamo meno porte del vSwitch), alcune delle schede di rete fisiche rimaranno inutilizzate. È quindi buona pratica utilizzare questa politica di bilanciamento solamente quando il numero di interfacce virtuali utilizzate supera quelle fisiche.

Bilanciamento basato sul MAC address sorgente

Questà modalità soffre degli stessi problemi della modalità descritta sopra con l'unica differenza che non è la porta d'entrata del vSwitch a stabilire la politica di bilanciamento, ma il source MAC address della scheda di rete delle interfacce virtuali presenti sulle macchine. Questa modalità quindi soffre di tutti gli stessi problemi della modalità descritta sopra ed è consigliabile il suo utilizzo solo quando il numero di interfacce virtuali (e relativi MAC address) superano quello delle interfacce fisiche.

Bilanciamento basato sull'hash dell'indirizzo IP

Questa terza modalità supplisce alle mancanze delle prime due che non permettono a macchine virtuali, con una sola interfaccia di rete virtuale, di utilizzare due o più interfacce fisiche di rete per trasmettere il loro traffico, impedendo ogni possibilità di bilanciamento di carico. La politica di bilanciamento di questa modalità è infatti basata su un calcolo (uno xor), compiuto tra l'indirizzo IP sorgente e l'IP destinazione di ogni connessione, per decidere quale interfaccia di rete fisica utilizzare per la trasmissione. È quindi possibile bilanciare le connessioni di una macchina virtuale con una sola interfaccia virtuale, su più interfacce di rete diverse, a condizione di comunicare con indirizzi IP differenti. Ricordiamo comunque che anche utilizzando indirizzi IP di destinazione diversi è possibile finire nel caso pessimo

con nessun bilanciamento delle connessioni (se per caso lo xor calcolato finisce con indicare la medesima scheda di rete).

2.6.5 Multipathing

Il concetto che sta alla base del multipathing [24] (vedi figura 2.27) è quello di cercare di avere due o più sentieri, che possano raggiungere una stessa destinazione replicando i componenti di rete come i cavi, gli switch, le schede di rete, ecc...
Il multipathing è una tecnica molto utilizzata all'interno di un infrastruttura di rete perché serve per rafforzare la disponibilità di un sistema e la sua tolleranza ai guasti.
Multipathing fornisce questi vantaggi:

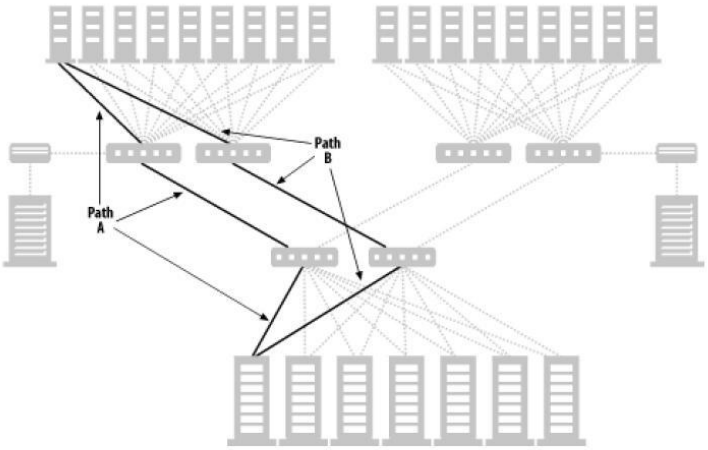

Figura 2.27: Realizzazione di multipathing con due sentieri [fonte: [15]]

- **Bilanciamento di carico (statico o dinamico)**:
  Come detto sopra il multipathing rende ogni risorsa raggiungibile seguendo più strade diverse. È possibile sfruttare questa peculiarità per ridistribuire in maniera dinamica o statica il traffico di rete, indiriz-

zandolo verso il sentiero meno utilizzato. Un bilanciamento di carico statico utilizza una politica di smistamento scelta a priori (Round Robin, pesata, ecc...) ma che non cambia nel tempo. Un bilanciamento dinamico permette invece di smistare il traffico dinamicamente su sentieri diversi, in tempo reale, monitorando e prendendo provvedimenti ad ogni cambiamento di situazione del traffico dei nodi.

- **Tolleranza ai guasti**:
  In caso di guasto di uno dei sentieri è sempre possibile continuare il lavoro reindirizzando il traffico verso uno degli altri. Più sono i sentieri, più un sistema sarà tollerante ai guasti, ma con esso aumenteranno anche il numero di componenti di rete necessarie a realizzarlo.

2.7 Considerazioni sul capitolo

All'interno di questo capitolo abbiamo affrontato vari argomenti partendo da un analisi iniziale dei concetti generali legati alla virtualizzazione, dandone una panoramica generale, via via addentrandoci sempre più nello specifico nella trattazione delle funzionalità offerte dai prodotti di virtualizzazione di VMWare. Capire il funzionamento di questi prodotti è la base di partenza per poter studiare configurazioni che possono metterle sotto sforzo durante la fase di testing. Come è necessario comprendere la differenza tra hypervisor hosted e nativo, da cui poi dipendono fortemente le prestazioni dei test è necessario anche prendere in considerazione tutta un altra serie di aspetti che vengono utilizzati in ambito aziendale. Per esempio non utilizzare VMotion snellisce il sistema, quando invece un suo utilizzo necessita di un'attenta pianificazione a livello progettuale di rete. Inoltre per quanto l'utilizzo di VLAN, Link Aggregation e Multipathing possa essere evitato è fortemente consigliato il loro utilizzo per avvicinarsi maggiormente ad un caso di studio che sia il più verosimile a quello di un azienda.
Molti argomenti di questo capitolo sono stati trattati sia per fornire una completezza a livello teorico sia per fornire una base di partenza per tutti coloro

che vorranno intraprendere un lavoro estensivo su questi argomenti. Per di più nella fase decisionale su quali test effettuare si sono provate molte di queste configurazioni (come ad esempio Link Aggregation e varie politiche di bilanciamento) ma che non è stato poi possibile portare avanti per mancanza di hardware specializzato, PC e licenze VMWare. Qualora si fosse a disposizione di un laboratorio tutti i concetti trattati in questo capitolo possono essere ripresi ed estesi.

Capitolo 3

VMWare e sistemi di memorizzazione

L'infrastruttura di memorizzazione, che fa da supporto a VMWare, è un altro aspetto importante da considerare durante la fase di design; questo aspetto è così importante e vasto a livello di contenuti che ho deciso di dedicargli un capitolo tutto suo. Prima di iniziare la trattazione di questo capitolo è importante chiarire il concetto di spazio di memorizzazione centralizzato che ricorrerà più volte all'interno della tesi. Con spazio di memorizzazione centralizzato nel contesto della virtualizzazione, come da letteratura [10], ci si riferirà ad una sua visione logicamente centralizzata ma fisicamente distribuita. Detto questo deve essere sempre favorita un'architettura di memorizzazione logicamente centralizzata e condivisa perché da questa dipendono [38]:

- **Funzionalità avanzate di VMWare**
  Funzionalità avanzate come High Availability, Fault Tolerance, Distributed Resource Scheduler, VMotion.

- **Prestazioni**
  Le prestazioni del sistema dipendono direttamente da come viene realizzata l'infrastruttura di memorizzazione condivisa.

- **Disponibilità**
  La disponibilità delle macchine virtuali dipende direttamente dall'infrastruttura di memorizzazione condivisa. Se lo spazio di memorizzazione non è raggiungibile in ogni momento, non è possibile ripristinare una macchina virtuale su un nuovo server ESX.

- **Consolidamento**
  Come vSphere ESX 4 permette di consolidare più macchine virtuali su uno stesso server, così un'architettura di memorizzazione condivisa permette di consolidare i bisogni di spazio delle varie macchine virtuali.

VMWare mette a disposizione tutta una serie di configurazioni dello spazio di memorizzazione che ESX riconosce [10]:

- Fibre Channel
- Fibre Channel over Ethernet
- iSCSI (hardware e sofware initiators)
- NAS (basati su NFS)
- Dischi Locali di tipo SAS/SATA/SCSI/SSD
- Infiniband

Data la complessità e la varietà delle tecnologie di memorizzazione disponibili, VMWare ha introdotto il concetto di *datastore* proprio per nascondere tutte queste complessità. Per le applicazioni e i sistemi operativi installati sulle macchine virtuali, il sottosistema di memorizzazione si presenta come un semplice adattatore Bus Logic o LSI SCSI collegato a uno o più dischi SCSI virtuali. Il datastore eroga spazio disco per molte macchine virtuali attraverso più macchine fisiche fornendo un modello molto semplificato per allocare spazio di memorizzazione dedicato ad ogni macchina virtuale senza che queste siano soggette alle complessità e alla varietà delle tecnologie di memorizzazione disponibili.

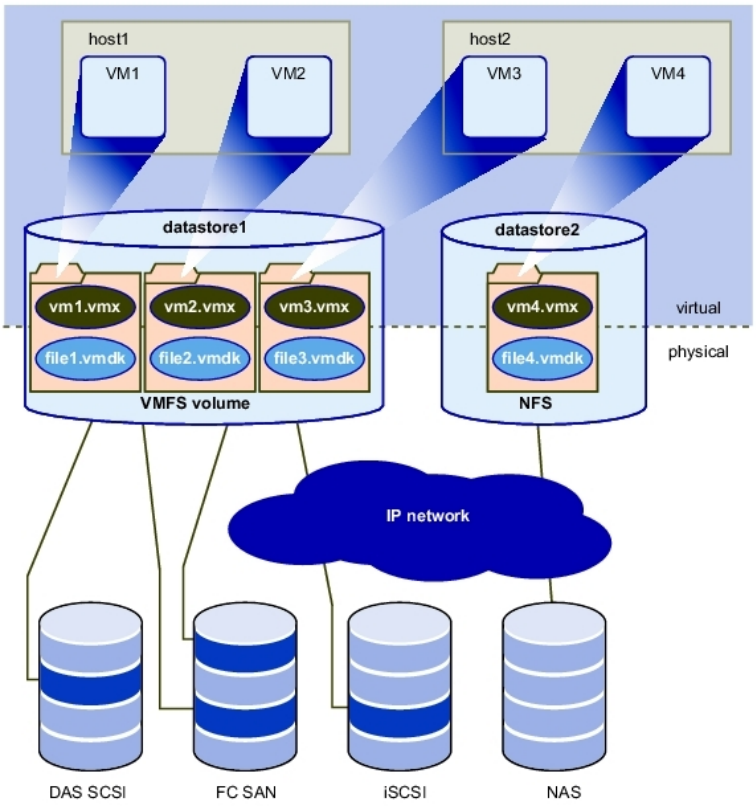

Figura 3.1: Architettura di memorizzazione di VMWare basata su datastore, LUN e sistemi di memorizzazione fisici eterogenei [fonte: http://www.clever-consulting.it/cms/General/vmware-architettura-di-storage-virtuale]

Una macchina virtuale è contenuta all'interno di un insieme di file situali all'interno di una cartella del datastore, dando quindi la possibilità di operare su questa, copiarla per esempio, accedendola come un semplice file. I datastore possono essere distribuiti su sottosistemi di memorizzazione fisici multipli ed eterogenei tra loro (vedi figura 3.1). Per esempio, un singolo volume VMFS può contenere una o più LUN prese, o dal disco fisico SCSI locale della propria macchina, o da una SAN, o da iSCSI, ecc...
Ogni nuova LUN aggiunta, a qualsiasi sottosistema di memorizzazione fisico, è automaticamente rilevata e resa disponibile a tutti i datastore. Un datastore, precedentemente creato, può essere espanso a caldo aggiungendo nuove LUN fisiche da qualsiasi sottosistema di memorizzazione ad essa visibile.
Ora che abbiamo visto cos'è e come funziona un datastore possiamo andare ad analizzare la parte fisica dei sottosistemi di memorizzazione partendo dalle due grosse macro categorie di topologie di rete: Storage Area Network (SAN), attraverso Fibre Channel o iSCSI, e Network Attached Storage (NAS) con Network File System (NFS).

3.1 Topologie di rete SAN e NAS

Storage Area Network:

> *"Una SAN [15](vedi figura 3.2) è una rete specializzata robusta e ad alta velocità, collegata a varie tipologie di sistemi di memorizzazione, che possono essere eterogenei tra loro, permettendone il trasferimento di dati attraverso l'utilizzo di comandi SCSI."*

I protocolli attualmente più utilizzati per trasportare i comandi SCSI in una SAN sono iSCSI e Fibre Channel. Una SAN differisce dal NAS perché lavora a livello di I/O di blocco e non di file. Lo spazio condiviso di una SAN viene poi presentato all'esterno sotto forma di spazio "grezzo" (*raw*) non formattato, a cui viene dato il nome di `LUN` (Logical Unit Number). Una LUN non è altro che una "porzione" di spazio logico della SAN, che può essere

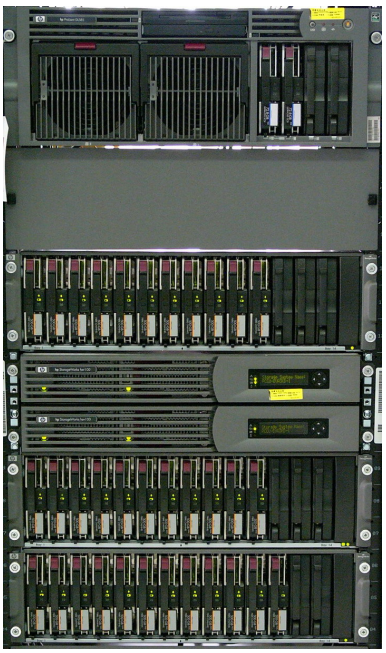

Figura 3.2: Storage Area Network [fonte: http://www.cic.pccu.edu.tw/hpc/hardware2.htm]

utilizzata localmente e formattata a piacimento. Una LUN non corrisponde quasi mai (anche se potrebbe), ad un unità disco fisica della SAN.

Network Attached Server:

> *"Un NAS [15](vedi figura 3.3) è formato da hardware specializzato (ma può anche essere formato da semplici computer), collegato attraverso la rete, che grazie a filesystem di rete come NFS o SMB/CIFS, fornisce un servizio per accedere al proprio spazio di memorizzazione."*

Un NAS, fornendo un servizio che si basa su TCP/IP, possiede un indirizzo IP che permette ai vari server di accedere attraverso la rete ai file contenuti in esso. Anche il NAS è una topologia di rete e non specifica quale protocollo di rete deve essere utilizzato per comunicare. Un NAS lavora a livello di file e non di blocco, come la SAN.

Figura 3.3: Network Attached Storage [fonte: http://www.qnap.com/]

3.1.1 SAN contro NAS

Ora andiamo ad analizzare una serie di differenze [15] tra NAS e SAN. Tra i pregi di entrambe le topologie:

- **Aumento delle prestazioni dei server**:
  I server che utilizzano SAN o NAS non consumano CPU nella gestione dei file.

- **Centralizzazione e consolidamento dei dati**:
  I dati sono in un unico posto con conseguente centralizzazione e consolidamento dovuto al fatto che non si possiede spazio inutilizzato su dischi locali.

- **Facilità d'integrazione**:
  NAS e SAN basate su iSCSI possono essere integrate direttamente con la rete aziendale Ethernet.

- **Sistema ibrido**
  SAN e NAS possono convivere insieme formando un servizio di memorizzazione ibrido.

Vantaggi NAS:

- **Basso costo**:
  Costo dei componenti inferiore rispetto a quello della SAN.

- **Acesso concorrente ai file**:
  I NAS gestiscono nativamente attraverso l'utilizzo del filesystem l'accesso concorrente ai file.

Svantaggi NAS:

- **Velocità dipendente dal traffico di rete**:
  La velocità del servizio dipende dalla velocità del traffico sulla rete LAN.

- **Maggior overhead**:
  Non ha la stessa capacità di trasporto di una SAN, dato che NFS o CIFS su TCP genera più overhead di SCSI su Fibre Channel (ma non di SCSI su iSCSI).

Vantaggi SAN:

- **Velocità**:
  Connessioni di rete ad alta velocità se si utilizza fibra ottica.

- **Flessibilità**:
  Possibilità di utilizzare iSCSI o Fibre Channel sia su fibra che sui normali cavi in rame.

- **Maggior capacità di trasporto**:
  Dovuto al minor overhead tra Fibre Channel e TCP/IP, all'elevata sicurezza che limita al minimo le ritasmissioni dei frame e all'utilizzo di un encoding di tipo 8b/10b.

Svantaggi SAN:

- **Maggior costo**:
  Costo maggiore dei componenti di una SAN in Fibre Channel rispetto a quelle di un NAS.
- **Aumento di complessità**:
  Switch, interfacce, cavi e tutte le altri componenti che fanno parte della tecnologia Fibre Channel aumentano la complessità della rete e la curva di apprendimento nel suo utilizzo.
- **Accesso concorrente alle LUN**:
  Permette accesso concorrente alle LUN, ma non gestisce la concorrenza a livello di file. Per gestire la concorrenza a livello di file necessita di un filesystem che ne permetta la gestione come VMFS.

La SAN, come detto in principio, è un termine astratto che indica una topologia di rete, anche se il termine viene comunemente utilizzato per indicare un sistema che utilizza Fibre Channel. Se quello che si desidera è un sistema ad alte prestazioni e l'azienda non possiede particolari problemi di risorse finanziarie, una SAN con Fibre Channel risulta essere la scelta più azzeccata. Oltre al Fibre Channel esistono anche altre possibilità meno performanti, ma anche meno costose, per creare una SAN, come ad esempio l'utilizzo di iSCSI. Procederemo quindi ad analizzare altri aspetti che devono essere presi in considerazione durante la progettazione di una SAN partendo da:

1. Dischi e Array di dischi.
2. Protocolli di rete: Fibre Channel e iSCSI.
3. Componenti di connettività di basso livello.
4. Visibilità e sicurezza nella SAN.

3.2 Dischi e array di dischi

In una SAN possiamo trovare supporti di memorizzazione eterogenei tra loro come dischi, tape library, jukebox ottici, ecc... Mentre i nastri vengono

utilizzati esclusivamente per i backup, i dischi vengono utilizzati per contenere i dati della SAN. Anche se una SAN può essere creata partendo da dischi sfusi collegati tra loro, è pratica comune quella di utilizzare array di dischi (vedi figura 3.4). I vantaggi nell'utilizzo di array di dischi rispetto a

Figura 3.4: Array di dischi [fonte: http://servextreme.skyrock.com/]

dischi sfusi sono molteplici [10]:

- Supporto per la cache tra dischi.
- Miglior disponibilità, tolleranza ai guasti dovuta alla ridondanza di controller, alimentatori, ventole, ecc...
- Miglior velocità di interconnessione tra i dischi.
- Controller specifici per array di dischi che permettono un migliore ottimizzazione delle operazioni di I/O e calcolo dei blocchi (*stripe*) di parità del RAID.

Ogni array di dischi è poi diverso dagli altri per una serie di cose:

- Tipologia di connessioni alla rete che offre (Ethernet o Fibre Channel per esempio).

- Processori per gestire le richieste di I/O ed eseguire il software dell'array.
- Software che fornisce varie funzionalità avanzate a livello di array come Thin Provisioning, clonazione dei dati, replicazione dei dati, ecc...
- Quantità di memoria cache.
- Tipologia di dischi che compongono l'array.

Le tipologie di dischi che possono essere utilizzate all'interno dell'array sono varie (SATA, SAS, SSD, ecc...) ed insieme ad altri aspetti determinano la velocità e la robustezza che la SAN può raggiungere. È sconsigliato l'utilizzo di dischi SATA per una SAN ad alte prestazioni, poiché nonostante il basso costo questi dischi non sono progettati per essere sottoposti a continuo carico 24 ore su 24, 7 giorni su 7, e tendono a generare più errori o malfunzionamenti rispetto ai dischi SAS. Esistono infatti delle misure chiamate `BER` (Bit Error Rate) e `MTBF` (Mean Time Between Failure) per indicare la robustezza di un disco e i SAS battono i dischi SATA in affidabilità[8, 1].
Se la SAN invece deve essere a basse prestazioni, ma fornire molto spazio, allora il costo per GB dei dischi SATA è nettamente inferiore a quello dei dischi SAS. I dischi SAS, d'altro canto, sono più costosi, più affidabili e più veloci rispetto la controparte SATA.
Lanciando uno sguardo al futuro però non si può che pensare all'utilizzo di dischi SSD. Questi dischi utilizzano memorie di tipo flash e si distinguono in tecnologia Multi Layer Cell (MLC) o Single Layer Cell (SLC). La tecnologia utilizzata ne modifica il costo e la velocità delle prestazioni in scrittura e lettura. La tecnologia SLC è di gran lunga più costosa e performante di quella MLC. A parte questo gli SSD possiedono pregi e difetti comuni. Tra i vantaggi troviamo: silenziosità, minore possibilità di rottura dovuta alla mancanza di parti meccaniche in movimento, minor consumo energetico durante le operazioni di lettura e scrittura, tempi di accesso ridotti (parlando di letture e scritture casuali mancando il tempo di latenza necessario a spostare le testine del disco) e minor produzione di calore. Il

principale contro di questa tecnologia è l'esorbitante costo per GB che ancora possiedono. Un altra considerazione da fare è il fatto che il disco ha una vita limitata, dipendente dal numero di scritture possibili prima che questo diventi inutilizzabile. Questo fattore viene mitigato dall'introduzione di nuovi algoritmi che distribuiscono le scritture uniformemente su tutto il disco (*wear levelling*). Ora vediamo un esempio sulla quantità di operazioni

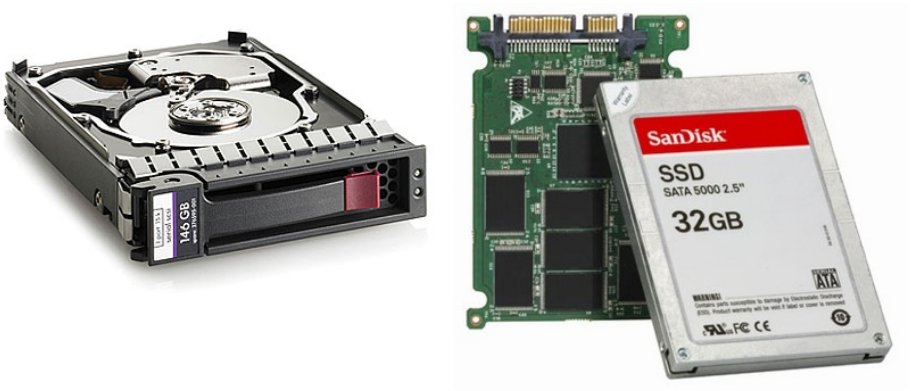

Figura 3.5: Disco SAS a sinistra e SSD a destra

eseguite al secondo, che alcune tipologie di dischi riescono a fornire sotto un generico carico di lavoro con ugual distribuzione di letture e scritture [10]:

```
7,2K RPM SATA: 80 IOps
10K RPM SATA: 120 IOps
15K RPM SAS: 180 IOps
SSD MLC: 1000-2000 IOps
```

Questi valori sono una media generica e possono differire a seconda del produttore di un disco e della tipologia del carico di lavoro. Sono misurati in IOps (Input/Output per second), una misurazione molto utlizzata in ambito di prestazioni di dischi. In generale la formula è [11]:

```
(1 / (ms di latenza media + ms di seek time media))
```

RAID (Redundant Array of Inexpensive/Independent Disks)

Nella maggioranza dei casi i dischi che compongono lo spazio di memorizzazione della SAN vengono poi messi in configurazione RAID. Esistono varie tipologie di configurazioni RAID ed ogni configurazione prende il nome dalla sua numerazione. I benefici, che si ottengono nell'utilizzare una configurazione RAID, dipendono direttamente dalla configurazione RAID scelta. Mentre in RAID 0 abbiamo un aumento prestazionale senza alcuna tolleranza ai guasti (poco utile in ambiente server), in RAID 5 abbiamo una leggera diminuzione di prestazioni ma un aumento alla tolleranza dei guasti.
Le configurazioni RAID vanno dalla 0 alla 6 (escludendo composizioni di configurazioni RAID come la 0+1 o 1+0 ecc...), anche se tra le implementazione più popolari in ambito aziendale troviamo RAID 5 e RAID 6. In RAID 5 i dati vengono partizionati in segmenti di uguale lunghezza e scritti su dischi differenti (*data striping*). La grandezza della partizione si chiama unità di *striping* (stripe letteralmente in inglese è striscia, ma possiamo leggerlo in questo contesto come striscia/blocchetto/fascetta di dati). Le partizioni, comprese quelle contenenti i bit di parità, vengono solitamente distribuite fra i dischi usando un algoritmo round robin. Sono proprio i bit di parità che permettono al sistema di ricostruire un disco in caso di guasto (basta infatti ricalcolarselo), anche se ne riducono di un pò la capacità effettiva di memorizzazione; se mettiamo 5 dischi in configurazione RAID 5 abbiamo una perdita di spazio del 20% (1/5 dello spazio è perso per memorizzare i bit di parità).
Comunque i pro rimangono maggiori dei contro, infatti si aumenta sia la tolleranza ai guasti, visto che il sistema può continuare a funzionare in modalità ridotta, anche se un disco si rompe (anche se con minori prestazioni, dovute al fatto che i dati presenti sul disco mancante vanno calcolati ogni volta che vengono richiesti), sia le prestazioni del sistema, dovute al fatto che possiamo utilizzare più controller disco per leggere i dati in maniera concorrente. La tolleranza ai guasti può venire ulteriormente aumentata con l'utilizzo di dischi *hot-spare*, cioè dischi in modalità passiva, pronti ad atti-

varsi automaticamente "a caldo" in caso il sistema rilevi il guasto di uno dei dischi, che fanno parte attivamente della configurazione RAID.

Il RAID 6 (vedi figura 3.6) è una configurazione simile al RAID 5 ma ab-

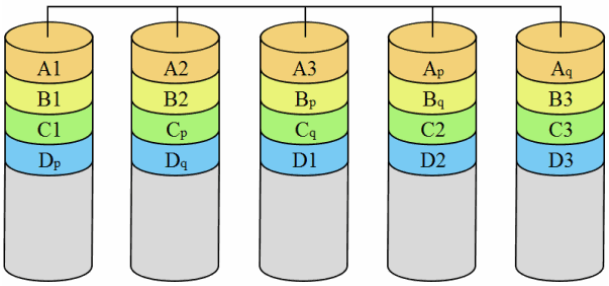

Figura 3.6: Esempio di configurazione RAID 6. Gli stripe `p` e `q` sono di parità [fonte: http://blog.stackoverflow.com/2009/02/]

biamo che ogni *stripe* di parità distribuito due volte tra i vari dischi. Questa configurazione permette quindi una maggiore tolleranza ai guasti rispetto a RAID 5, dal momento che è possibile continuare l'erogazione del servizio anche con due malfunzionamenti. Rispetto al RAID 5 si ha una penalità più alta nelle fasi di scrittura e una maggiore perdita di spazio di memorizzazione dovuta all'utilizzo di due *stripe* di parità invece di uno.

3.3 Protocolli di rete: Fibre Channel e iSCSI

Ogni SAN necessita di uno o più protocolli di rete per poter comunicare internamente tra i vari dischi e/o array di dischi. Tra i vari protocolli di rete che possono essere utilizzati all'interno di una SAN i più utilizzati sono:

- Fibre Channel
- iSCSI

Questi protocolli sono trasparenti a livello d'applicazione; per un applicativo l'utilizzo di una connessione diretta SCSI, Fibre Channel o iSCSI è indifferente, dal momento che tutti e tre forniscono I/O a livello di blocco attraverso comandi SCSI.

3.3.1 Fibre Channel

Il Fibre Channel (vedi figura 3.7) è stato inventato nel 1988 per supplire ad alcune mancanze dello SCSI, non per sostituirlo, come alcuni erroneamente pensano, e per trasportarne i comandi. Tra le limitazioni dello SCSI vi era il

Figura 3.7: Alcuni esempi di strumentazione Fibre Channel: scheda Fibre Channel a sinistra e switch Fibre Channel a destra

fatto che il segnale non si poteva propagare per più di 25 metri per problemi di disallineamento [17]. Fibre Channel si distingue dallo SCSI, prima di tutto per il fatto di essere un interfaccia seriale come il SAS, e non parallela come lo SCSI tradizionale. In principio era stato denominato Fiber Channel. *"Fiber"* è il termine americano che sta per fibra, dal momento che voleva essere chiaro che il protocollo doveva viaggiare su fibra ottica. Poi però si è aggiunta la possibilità di fare viaggiare il protocollo anche sui cavi in rame e quindi si è deciso di utilizzare il termine inglese *"Fibre"* per distinguerlo dall'altro termine (anche se la cosa probabilmente ha generato solamente altra confusione). Il Fibre Channel gode di grandissima fama in ambito SAN e soprattutto nel contesto dei centri di dati. La sua fama è data da:

- Alta capacità di carico e bassa latenza.
- Lunghezza massima raggiungibile con fibra ottica che va dai 300 metri con fibra multimodale a 40 km con fibra monomodale.
- Trasmissione affidabile dei dati con l'abilità di ricevere la conferma che i dati sono stati consegnati senza errori.

- Permette la gestione a caldo dei supporti, non si deve quindi spegnere il sistema per aggiungere o sostituire dei componenti.
- Possibilità di avere Quality of Service (QoS).
- Politiche di controllo di flusso per evitare la congestione della rete.

Topologia di una rete Fibre Channel

Una topologia di rete viene formata collegando tra di loro un certo numero di dispositivi che comunicano attivamente sulla rete. I dispositivi che possono essere collegati sono vari: periferiche disco, schede di rete Fibre Channel, switch Fibre Channel, ecc... Le topologie tipiche in ambiente Fibre Channel sono principalmente tre [15, 17]:

1. **Punto-Punto (FC-P2P)**:
   Due apparati vengono connessi direttamente. È la topologia più semplice, con connettività limitata (vedi figura 3.8).

2. **Arbitrated loop (FC-AL)**:
   In questa topologia (figura 3.8), tutti gli apparati sono disposti su un anello, in modo simile alla tecnologia Token Ring. Aggiungere o togliere un apparato dall'anello causa l'interruzione completa della connettività per tutto l'anello e il guasto di un apparato interrompe l'anello bloccandone completamente la funzionalità. Esistono anche varie tecniche per fare in modo che in caso di guasto di una porta questa venga esclusa. Da notare che un anello minimo costituito da solo due porte parrebbe simile a una connessione FC-P2P, ma in realtà ne differisce molto in termini di protocollo.

3. **Switched fabric (FC-SW)**:
   Tutti gli apparati o insiemi di apparati sono collegati a degli switch Fibre Channel (vedi figura 3.9). Tra i vantaggi di questa configurazione abbiamo il fatto che gli switch sono in grado di gestire l'ottimizzazione delle interconnessioni, il fatto che più coppie di porte possono

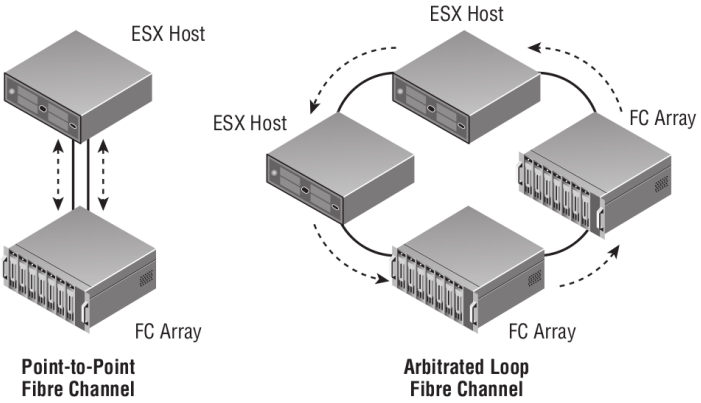

Figura 3.8: Fibre Channel in configurazione Punto-Punto a sinistra e Arbitrated Loop a destra [fonte: Mastering VMWare vSphere 4]

comunicare contemporaneamente ed il fatto che il guasto di una porta viene isolato e non dovrebbe influenzare l'operatività delle altre porte. Questa risulta essere la configurazione di maggior utilizzo.

Architettura Fibre Channel

Fibre Channel definisce un architettura multi livello (vedi figura 3.10) fatta nel seguente modo che assomiglia a quella ISO/OSI [15, 17]:

FC0 : Livello fisico, che comprende cavi, fibre ottiche, connettori, ecc...

FC1 : Livello *data link*, che implementa la codifica e decodifica del segnale (8b/10b o 64b/66b).

FC2 : Livello *network*, che definisce come spezzare in frame i dati che riceve dai livelli superiori e la gestione del flusso di dati.

FC3 : Livello *common service*, un livello "sottile" serve per implementare varie funzioni come per esempio il multicasting.

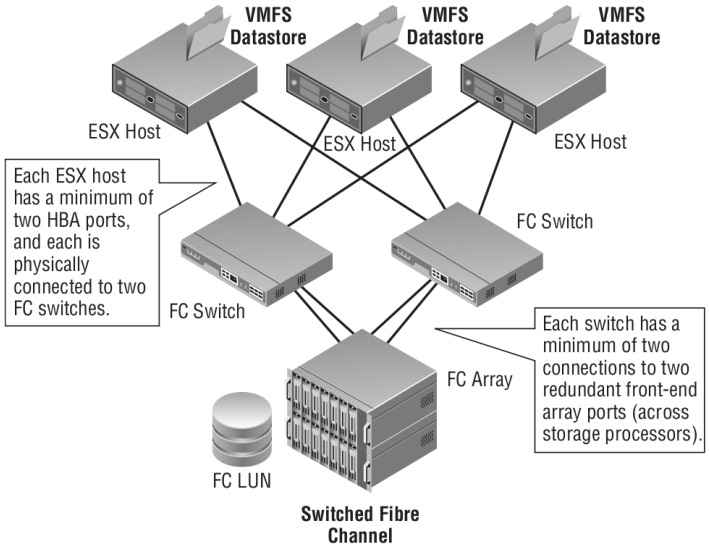

Figura 3.9: Fibre Channel in configurazione Switched Fabric [fonte: Mastering VMWare vSphere 4]

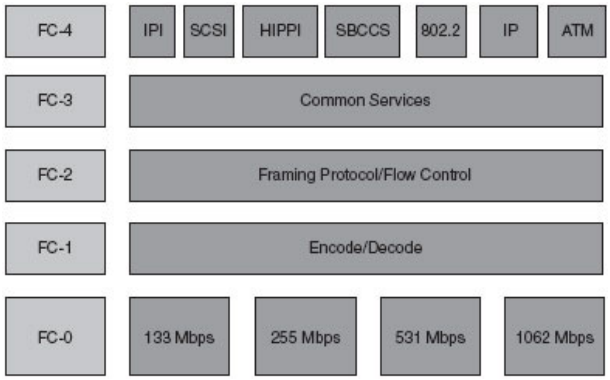

Figura 3.10: Architettura multi livello del Fibre Channel [fonte: `http://searchstoragechannel.techtarget.com/generic/0,295582,sid98_gci1250205,00.html`]

FC4 : Livello *protocol mapping*, nel quale gli altri protocolli vengono mappati per poter essere utilizzabili con Fibre Channel (in pratica viene definito come devono essere incapsulati).

FC0, FC1, e FC2 sono conosciuti anche come `FC-PHY`, i livelli fisici del Fibre Channel. I router Fibre Channel lavorano fino al livello FC4, cioè potrebbero lavorare anche come router SCSI, gli switch fino a FC2 e gli hub solamente al livello FC0. I prodotti Fibre Channel sono disponibili a velocità di 1 Gbit/s, 2 Gbit/s, 4 Gbit/s, 8 Gbit/s, 10 Gbit/s e 20 Gbit/s. I prodotti basati sugli standard 1, 2, 4 e 8 Gbit/s sono compatibili tra loro. Lo standard a 10 Gbit/s e 20 Gbit/s, invece, non è retro compatibile con nessuno degli apparati a minore velocità, poiché è molto differente a livello FC1, utilizzando una codifica diversa a 64b/66b, invece che 8b/10b (la codifica 64b/66b genera meno overhead) e sono principalmente utilizzati per interconnettere fra loro più switch.

Protocolli Fibre Channel

Tra i protocolli che hanno a che fare con Fibre Channel [17] troviamo:

- FCP (Fibre Channel Protocol)
- iFCP (Internet Fibre Channel Protocol)
- FCIP (Fibre Channel IP)
- FCoE (Fibre Channel over Ethernet)

FCP

Il Fibre Channel Protocol (FCP) (vedi figura 3.11) non è altro che un protocollo di livello *transport* (come il TCP), che permette di trasportare comandi SCSI su Fibre Channel (ne abbiamo parlato sopra). È necessario fare attenzione al fatto che FCP, formato dai livelli alti del protocollo, solitamente lavora in concomitanza con FC-PHY (i livelli bassi del protocollo), ma non sempre. Infatti i frame Fibre Channel possono anche essere incapsulati su

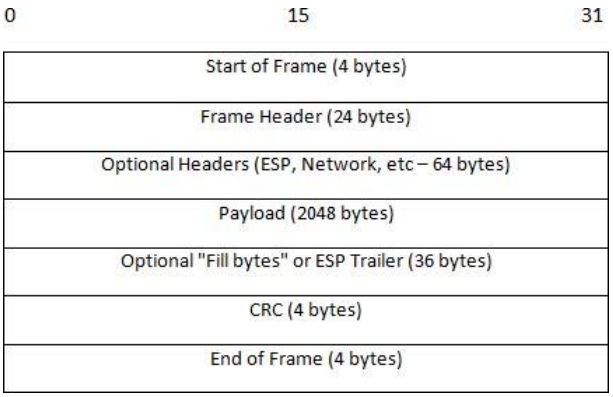

Figura 3.11: Frame FCP [fonte: http://www.enterprisenetworkingplanet.com/netsp/article.php/3690921/Storage-Networking-101--Understanding-the-Fibre-Channel-Protocol.htm]

Ethernet (e abbiamo FCoE) o su TCP/IP (e abbiamo iFCP).
Dal momento che un frame Fibre Channel è di 2148 byte, in caso di utilizzo di iFCP o FCIP, sono necessari dei meccanismi per spezzare i frame Fibre Channel in dimensioni più piccole così da poter viggiare su IP (1460 byte di *payload*). Ridurre le dimensioni dei frame porta però ad un maggiore overhead, per questo motivo esiste l'alternativa di utilizzare i Jumbo Frame. I Jumbo Frame sono frame di grosse dimensioni, che però necessitano di tutta una strumentazione di rete lungo il cammino, che li supporti pena lo scarto brutale del frame.

iFCP

	Periferiche	**Protocollo**
iSCSI	iSCSI/IP	Internet Protocol
FCIP	Fibre Channel	Fibre Channel
iFCP	Fibre Channel	Internet Protocol

Tabella 3.1: Tabella riassuntiva dei mezzi di trasporto e periferiche utilizzate per: iSCSI, FCIP e iFCP

L'iFCP (vedi figura 3.12) è un protocollo che permette di trasportare comandi SCSI incapsulati dentro il livello 4 del Fibre Channel, che a sua volta è

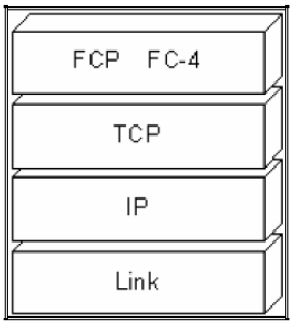

Figura 3.12: iFCP: stack iFCP [fonte: http://www.ixbt.com/storage/iscsi.shtml]

incapsulato su TCP/IP. La differenza tra iFCP e FCIP è che, nonostante entrambi utilizzino TCP/IP, iFCP incapsula solamente il livello 4 del protocollo FCP rimpiazzando i livelli più bassi con quelli del TCP, mentre FCIP incapsula tutto lo *stack* FCP. FCIP è quindi un protocollo di *tunneling*, mentre iFCP di *routing*.
In caso di utilizzo di iFCP le funzionalità di controllo sulla congestione, controllo del flusso e di rilevamento d'errore, sono gestite completamente dal TCP e non più dal Fibre Channel. iFCP può essere utilizzato sia per fare comunicare tra loro periferiche Fibre Channel all'interno della SAN (una SAN per esempio con schede di rete Fibre Channel e rete Ethernet dal momento che iFCP non può viaggiare su fibra) sia per attraversare Internet come FCIP.

FCIP
L'FCIP (vedi figura 3.13) è un protocollo che permette di effettuare *tunneling* tra più SAN Fibre Channel, passando per una rete IP, incapsulando lo *stack* del FCP dentro TCP/IP. FCIP permette quindi di attraversare Internet per collegare fra loro due SAN geograficamente distanti. In caso di utilizzo di FCIP le funzionalità di controllo del flusso dei dati e di rilevamento d'errore sono gestite dal TCP durante il traggitto sulla rete IP. All'arrivo i pacchetti

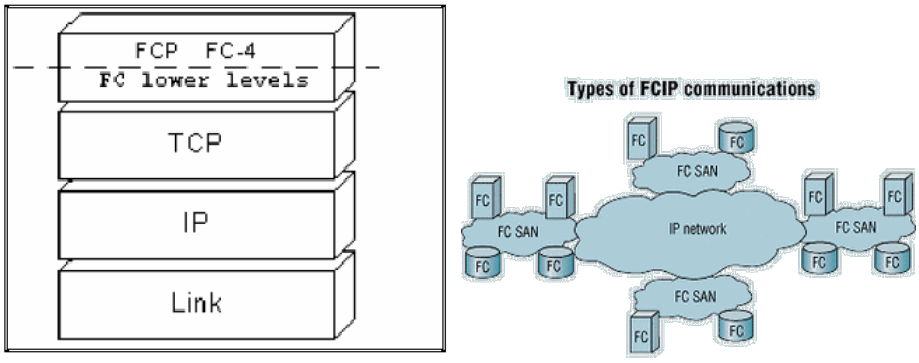

Figura 3.13: FCIP: *stack* FCIP a sinistra e utilizzo di FCIP per interconnettere varie SAN in Fibre Channel attraverso Internet [fonte: http://ixbtlabs.com/articles2/iscsi/]

FCP vengono estratti lasciando inalterate tutte le varie funzionalità del Fibre Channel.

Fibre Channel over Ethernet (FCoE)

Con l'avvento di connessioni Ethernet a 10 GB, FCoE [9, 12] (vedi figura 3.14) si ripromette di trasportare i frame Fibre Channel sopra queste senza

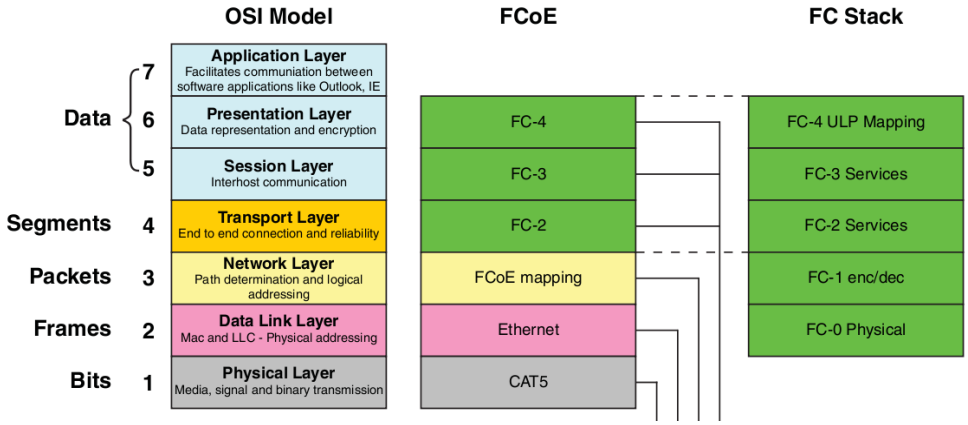

Figura 3.14: Fibre Channel over Ethernet [fonte: [12]]

apportare modifiche al protocollo e alle funzionalità Fibre Channel esistenti. Per operare FCoE ha bisogno di una rete *lossless* Ethernet, che garantisca un trasporto senza perdita di pacchetti indispensabile in ambito SAN. Inoltre

FCoE permette di consolidare il traffico Fibre Channel sulla connessione Ethernet utilizzata per i dati, permettendo così una più semplice connettività dei server all'infrastruttura della SAN, garantendo allo stesso tempo la piena compatibilità con tutto il software sviluppato per Fibre Channel. Il trasporto dei pacchetti Fibre Channel è trasparente e sono garantite tutte le funzionalità native del Fibre Channel, incluso lo zoning.
Rispetto ad una soluzione completamente in Fibre Channel sono necessarie apposite schede di rete, chiamate Converged Network Adapter (CNA), e switch FCoE. Uno switch FCoE possiede porte di tipo Fibre Channel e porte di tipo Ethernet e permette la trasformazione da un tipo di traffico all'altro. Una CNA, invece, è una scheda che contiene sia un controller Fibre Channel, sia una scheda di rete Ethernet. Ha quindi bisogno di entrambi i driver Fibre Channel e Ethernet per potere funzionare. Una tipica spedizione di dati avviene così: i dati vengono incapsulati in frame Fibre Channel dal controller Fibre Channel, che poi manda internamente i dati alla scheda di rete Ethernet che provvede a spedirli.
Uno dei maggiori problemi nell'utilizzare Ethernet per il trasporto è il fatto che è un protocollo, che a differenza di Fibre Channel, prevede lo scarto dei frame (non è *lossless*). Per trasformare Ethernet in un protocollo senza scarto di frame è necessario eliminare i seguenti problemi:

- **Errore di trasmissione dei frame**:
  Facilmente eliminabile, riducendolo a tassi insignificanti, grazie alle moderne tecnologie di cablaggio.

- **Collisioni**:
  Ethernet è un protocollo che ammette la collisione. Per utilizzare FCoE è necessario utilizzare tecnologia full-duplex.

- **Congestione**:
  La congestione avviene quando si verifica un *buffer overflow*, ossia quando non è possibile inviare su una linea tutti i pacchetti che si hanno in ingresso, in quanto la velocità di trasmissione risulta inferiore alla

velocità con cui i pacchetti sono ricevuti. Ethernet non supporta un meccanismo di gestione della congestione a crediti, come si ha per Fibre Channel, ma possiede uno strumento di controllo di flusso che permette di ridurre al minimo le congestioni. Il controllo di flusso si basa sul frame PAUSE, che viene spedito per avvisare chi trasmette che deve fermarsi e attendere un ulteriore segnalazione. Recentemente si sta lavorando proprio per modificare questo frame PAUSE ed estenderlo secondo un meccanismo a priorità, evitando quindi di bloccare tutti i flussi di trasmissione.

Tra i vantaggi sull'utilizzo di FCoE:

- Basso costo dei componenti Ethernet rispetto a quelli Fibre Channel.
- Elevata diffusione di reti Ethernet in ambito aziendale.
- Semplificazione e consolidamento (vedi figura 3.15) dell'architettura della rete data dalla riduzione delle tecnologie utilizzate.

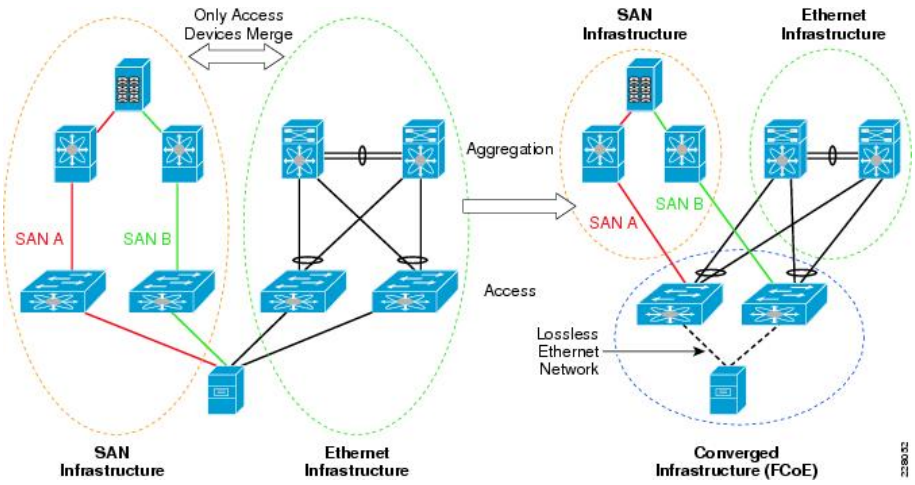

Figura 3.15: Consolidamento dato dall'utilizzo di FCoE [fonte: [12]]

Tra gli svantaggi:

- Fibre Channel è ancora lo standard in ambito SAN.

- Minor capacità di carico (sarà necessario attendere il diffondersi di reti Ethernet a 10GBit/s).
- Necessita di switch che gestiscano il controllo di flusso per evitare le congestioni che porterebbero alla perdita di frame.

3.3.2 Small Computer System Interface (SCSI)

Prima di iniziare a parlare di iSCSI è necessario definire che cos'è e come funziona lo SCSI. Lo SCSI [18, 17] è un sistema standard che definisce come collegare fisicamente e trasferire dati tra varie periferiche come hard disk, CD ROM, stampanti, ecc... Ad oggi SCSI è arrivato alla sua terza versione chiamata per l'appunto SCSI-3. Lo SCSI deve innanzi tutto essere suddiviso

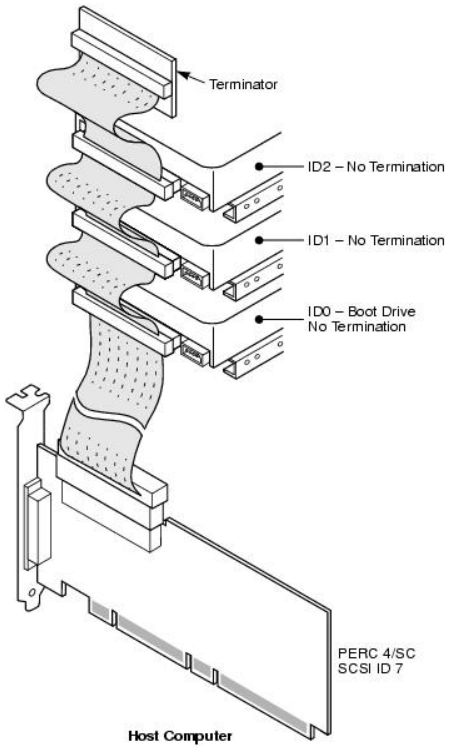

Figura 3.16: Configurazione SCSI [fonte: [17]]

in due entità per non generare confusione:

- la parte fisica di basso livello di trasporto del segnale e di cablaggio.
- la parte di alto livello formata dal protocollo di trasmissione dei dati.

SCSI componente fisica

Le versioni più recenti di SCSI (come per esempio Ultra-640 SCSI) permettono di collegare tra loro fino a 16 periferiche, condividendo lo stesso bus da 16 bit. Questa forma di collegamento è chiamata multi-drop. Una tipica configurazione *multi-drop* (vedi figura 3.16) è quella in cui partecipano un iniziatore (*initiator*), un terminatore (*terminator*) e delle periferiche SCSI (dette *target*). L'iniziatore è un dispositivo che gestisce le richieste verso le periferiche collegate, trasferendo loro i comandi e aspettando le risposte. Un terminatore invece è un componente che serve per evitare delle interferenze del segnale, che potrebbero causare errori di trasmissione. SCSI è quindi un sistema client-server dove i client sono gli initiator e i server sono i target.

Le periferiche SCSI collegate al bus possono arrivare ad un massimo di

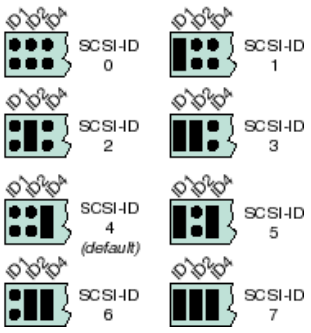

Figura 3.17: Configurazione degli SCSI ID attraverso l'utilizzo dei pin [fonte: http://www.vmguru.nl/wordpress/2009/01/new-vmware-products-in-2009/]

16, anche se solitamente non ne vengono collegate più di 5, pena un decadimento graduale delle prestazioni (che aumenta con l'aumentare delle periferiche collegate), dovuto al fatto che solamente una periferica alla volta può trasmettere sul bus. Ad ogni periferica collegata bisogna associare uno SCSI ID differente che serve per il suo riconoscimento univoco. Mentre una volta

questa operazione andava eseguita a mano (vedi figura 3.17), nei sistemi SCSI moderni lo SCSI ID viene associato in maniera dinamica la prima volta che si dialoga con una periferica. Dal momento che solo una periferica alla volta può utilizzare il bus, è necessario utilizzare un protocollo per arbitrarle nel caso che richiedano contemporaneamente di trasmettere dei dati. Mentre questo protocollo sta decidendo chi deve trasmettere, tutte le periferiche sono ferme e il bus rimane vuoto. Questo overhead riduce le prestazioni di una configurazione *multi-drop* diminuendone la banda a disposizione: la capacità di carico effettiva che si raggiunge è all'incirca la metà di quello potenzialmente raggiungibile dal bus.
Un altro grosso problema dello SCSI è quello dato dal disallineamento (*skew*) dei segnali, che passano sul bus dovuti ai ritardi di propagazione (vedi figura 3.18). Questo problema non permette ai bus SCSI di essere più lunghi di 25 metri e nei casi dove la frequenza di trasmissione dei dati è molto alta, si arriva ad avere anche cavi più corti. Infatti maggiore è la velocità del segnale, maggiore è la capacità di carico che si riesce ad ottenere, ma minore è la distanza che il bus può raggiungere.

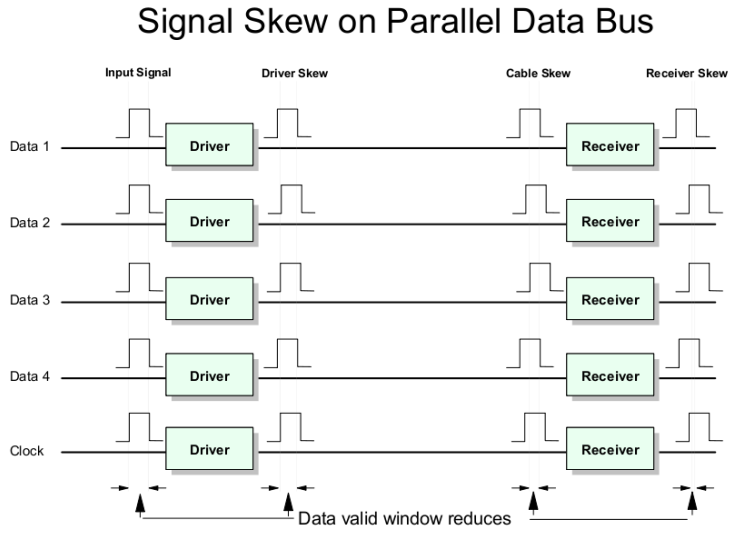

Figura 3.18: Segnale SCSI parallelo e disallineamento del segnale [fonte: [17]]

SCSI Command Protocol

La parte che abbiamo visto fino ad ora parla della componente di comunicazione SCSI fisica. Ma SCSI non è solo quello. SCSI è anche un protocollo che serve per fare eseguire comandi alle periferiche di tipologia SCSI. La comunicazione avviene tra un iniziatore (*inititator*) ed una periferica SCSI (*target*). I comandi vengono spediti attraverso un blocchetto di dati chiamato Command Descriptor Block (CDB), formato da un byte per determinare l'operazione da eseguire e da altri 5 o più byte per i parametri. Anche Fibre Channel utilizza questo protocollo inglobandolo nel livello 4 del Fibre Channel.

3.3.3 iSCSI

L'iSCSI è un protocollo che permette di trasportare comandi SCSI, utilizzando TCP/IP (vedi figura 3.19). iSCSI è una alternativa poco costosa e meno performante al Fibre Channel e solitamente viene utilizzato per:

- Facilitare il trasferimento dei dati su lunghe distanze, utilizzando l'infrastruttura di internet già esistente. Per esempio, per effettuare backup, un'operazione che non necessita di particolare velocità, in un sito remoto per un eventuale recupero in caso di disastro.
- Creare una SAN a basse prestazioni.
- Sfruttare I/O a livello di blocco invece che I/O a livello di file, come nel NAS, eliminando l'overhead del filesystem.

Il grosso problema di iSCSI risiede nell'utilizzo del protocollo TCP/IP, che ne limita le prestazioni, dato l'elevato overhead dovuto a:

- Riassemblaggio dei pacchetti che arrivano in modo disordinato.
- Rilevamento degli errori e conseguente ritrasmissione, visto che non è un protocollo *lossless*.

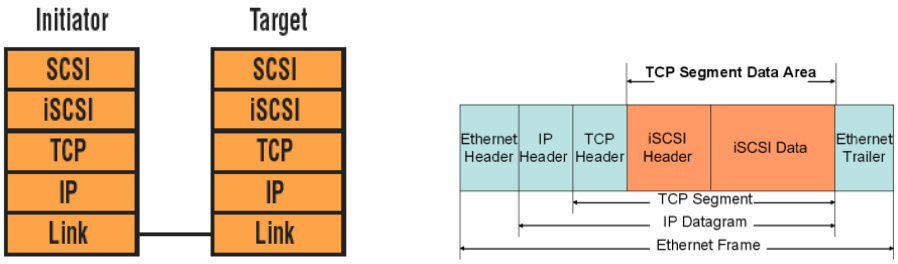

Figura 3.19: Stack iSCSI a sinistra e pacchetto iSCSI a destra [fonte: http://electronicdesign.com/article/boards-modules-systems/iscsi-does-10g-ethernet13285.aspx]

- Spreco di CPU utilizzata per impacchettare e spacchettare i dati, nel e dal pacchetto TCP, che può essere mitigato dall'utilizzo di schede di rete apposite che fanno questo lavoro (chiamate TCP Offload Engine).

I termini maggiormente utilizzati quando si parla di iSCSI sono:

iSCSI initiator : È un componente client (software o hardware), che andrà ad accedere ai vari *target* iSCSI. Può accedere a più di un *target* contemporaneamente. Ha il compito di inserire e rimuovere i pacchetti iSCSI dentro quelli TCP/IP e di effettuare i controlli su eventuali errori di ricezione.

iSCSI target : È un componente logico che viene contattato da un iSCSI initiator, attraverso la rete, per effettuare delle operazioni. Un iSCSI *target* possiede un iSCSI qualified name (*iqn*) ed è fornito di uno o più indirizzi IP per poter essere raggiunto attraverso la rete e servire delle richieste.

discovery : Il processo nel quale si mostrano i vari *target* disponibili ad un *initiator*. Un metodo comune è quello di avere un server iSNS (simile al DNS per scoprire l'indirizzo IP di una macchina dato il nome).

iSCSI naming : I nomi iSCSI possono essere di due tipologie:

1. **iqn**: iSCSI qualified name
2. **eui**: eui-64 bit identifier

Al giorno d'oggi nella maggioranza dei casi vengono utilizzati nomi di tipo iqn. Un iqn come questo di esempio

`iqn.1993-08.org.debian:01.35ef13adb6d`

è formato da:

- `iqn`: definisce che tipologia di nomenclatura viene utilizzata.
- `1993-08`: l'anno ed il mese in cui la naming authority ha acquisito il nome di dominio utilizzato subito di seguito.
- `org.debian`: nome di dominio invertito.
- `01.35ef13adb6d`: stringa univoca definita dalla naming authority.

Nei sistemi Linux le funzionalità iSCSI vengono fornite dai pacchetti `iscsi-target` e `open-iscsi`.

HSCSI

L'HSCSI è un protocollo che permette di trasportare comandi SCSI incapsulandoli all'interno di frame Ethernet. Questo protocollo è scarsamente utilizzato; viene utilizzato per SAN a basso costo, che non utilizzano la tecnologia Fibre Channel. Il principale vantaggio dell'utilizzo di HSCSI su iSCSI sono le maggiori prestazioni che si possono ottenere, date dal fatto che si ha un minor overhead lavorando a livello di frame Ethernet rispetto, a pacchetti TCP/IP. I contro dell'HSCSI sono quelli di non poter sfruttare il routing dell'IP e quindi è possibile collegare tra loro solamente SAN, che sono raggiungibili a livello di switch e quindi di conseguenza non molto lontane tra loro.

3.3.4 Componenti di connettività

A livello di connettività di livello fisico abbiamo:

- Fibra ottica
- Cavo SCSI
- Cavo UTP

Fibra ottica

La fibra ottica (vedi figura 3.20) risulta ottimale in ambito SAN dal momento che ci permette di raggiungere elevate velocità e distanze. Abbiamo

Figura 3.20: Cavo in fibra ottica a sinistra e rifrazione laser a destra [fonte: http://en.wikipedia.org/wiki/File:Laser_in_fibre.jpg]

due tipologie di cavi:

- **Cavi multimodali**
- **Cavi monomodali**

Un cavo monomodale serve per raggiungere distanze maggiori rispetto ad un cavo multimodale, anche se il suo costo è maggiore dal momento che il trasmettitore laser deve essere migliore rispetto ad uno multimodale. Un trasmettitore laser monomodale può arrivare a distanze che vanno dai 2 ai 50 kilometri, trasmette un unico fascio di luce e le dimensioni dei cavi sono pari a 8-10 micrometri. Con l'aumentare della distanza diminuisce la banda.

Il tipico colore della guaina di un cavo monomodale è gialla (anche se il colore non viene sempre rispettato).
Per quanto riguarda i cavi multimodali, invece, possiamo dire che hanno un costo minore rispetto a quelli monomodali e raggiungono distanze minori rispetto a loro. Le distanze raggiungibili con i cavi multimodali vanno dai 200 ai 1500 metri a seconda dell'ampiezza del trasmettitore e della categoria del cavo. Un trasmettitore multimodale trasmette più fasci di luce e le dimensioni del cavo vanno dai 50 ai 100 micrometri circa. Esistono 3 categorie di cavi multimodali:

Categoria	Distanza	Colore Guaina[1]
OM1 (65 nm)	200 m	Grigio
OM2 (50 nm)	500 m	Arancione
OM3 (50 nm)	1500 m	Acqua

Tabella 3.2: Varie distanze raggiungibili attraverso la fibra ottica

Cavo SCSI

Un cavo SCSI (vedi figura 3.21) è un cavo piatto in rame con vari canali che trasportano in parallelo più bit di informazioni. I cavi SCSI vengono utilizzati in locale, per collegare tra loro le periferiche alla scheda madre. L'utilizzo di collegamenti SCSI è comunque ormai in disuso, perché la lunghezza massima che può raggiungere è di solamente 25 metri circa (anche meno se vogliamo aumentare la capacità di carico).

Cavo Twisted Pair

I cavi Twisted Pair (vedi figura 3.22) (doppino ritorto), sono cavi utilizzati normalmente nelle reti Ethernet e Token Ring e sono terminati da un connettore RJ45. Questi sono suddivisi in Unshielded Twisted Pair (UTP: cavi non schermati) e Shielded Twisted Pair (STP: cavi schermati). I cavi

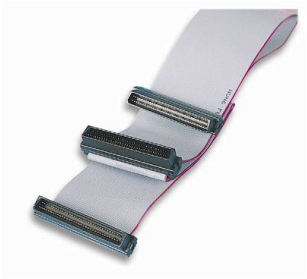

Figura 3.21: Cavo SCSI [fonte: http://www.ianua.com/patrizia/accessoristicaiso/cavi/pagine/CAVI\%20SCSI.htm]

schermati vengono utilizzati per ridurre ulteriormente le interferenze elettromagnetiche che possono disturbare il segnale. I cavi Twisted Pair sono

Figura 3.22: Cavo Ethernet a sinistra e connettore RJ45 a destra

suddivisi in categorie:

Cat1 : Utilizzati per la telefonia e ISDN.

Cat2 : Utilizzati nelle reti Token Ring 4 Mbit/s.

Cat3 : Utilizzati nelle reti Ethernet a 10 Mbit/s. Frequenza massima raggiungibile dal segnale di 16 MHz.

Cat4 : Utilizzati nelle reti Token Ring a 16 Mbit/s. Frequenza massima raggiungibile dal segnale di 20 MHz.

Cat5 : Utilizzati nelle reti Ethernet a 100 Mbit/s. Frequenza massima raggiungibile dal segnale di 100 MHz. Potrebbe non essere sufficiente per le reti a 1000 Mbit/s.

Cat5e : Utilizzati nelle reti Ethernet a 100 Mbit/s e 1000 Mbit/s. Frequenza massima raggiungibile 120 MHz.

Cat6 : Frequenza massima raggiungibile dal segnale di 250 MHz. Raddoppia la frequenza raggiungibile da Cat5 e Cat5e.

Cat6e : Possono essere utilizzati nelle reti Ethernet a 10 Gbit/s. Frequenza massima raggiungibile dal segnale di 500 MHz.

Cat7 : Frequenza massima raggiungibile dal segnale di 600 MHz. Obbligo di una schermatura S/FTP, che implica 4 schermature a coppie per i fili all'interno del cavo, più una schermatura del cavo stesso.

Cat7e : Frequenza massima raggiungibile dal segnale di 1000 MHz. Obbligo di una schermatura S/FTP, che implica 4 schermature a coppie per i fili all'interno del cavo, più una schermatura del cavo stesso.

La lunghezza massima che un cavo Twisted Pair può raggiungere nello standard Gigabit Ethernet a 10 Gbit/s è di 55 metri con cavi Cat6. Utilizzando i nuovi cavi Cat7, che sono più schermati, si ha una maggior riduzione delle interferenze (*crosstalk*) e si possono raggiungere lunghezze di trasmissione di 100 metri. Attenzione però, con il nuovo arrivo della tecnologia Gigabit Ethernet a 40 Gbit/s e 100 Gbit/s prevista a breve, la trasmissione su cavi in rame sarà ancora possibile, ma la distanza raggiungibile sarà minima.

3.4 Visibilità e sicurezza nella SAN

Una delle espressioni dell'elasticità della SAN è la possibilità di far dialogare qualsiasi client con qualsiasi LUN. A volte questa stessa elasticità rende il traffico dei dati sin troppo complesso da gestire. In altri casi si vuole evitare che certi dispositivi comunichino e vedano tutte le LUN disponibili, per ragioni che possono andare da motivi di sicurezza, al fatto di voler evitare possibili corruzioni su dati. Per esempio certi dipartimenti di una ditta potrebbero non essere abilitati a vedere dati appartenenti ad altri dipartimenti. Oppure

si potrebbe voler separare tutte le macchine che utilizzano un determinato filesystem (come EXT3) da tutte quelle che ne utilizzano uno differente, in modo che non possono per sbaglio andare a corromperne il contenuto.
È quindi necessario trovare una maniera in cui sia possibile limitare lo "spazio di manovra" nell'accesso alle risorse di memorizzazione collocate nella SAN da parte di client esterni. Tutto ciò si può ottenere attraverso l'utilizzo di Zoning o di VLAN, con lo scopo finale di avere zone di visibilità distinte tra di loro. Queste metodologie portano i seguenti vantaggi:

- Maggior sicurezza visto che ognuno può accedere solamente alle LUN presenti nelle sue zone d'accesso.
- Integrità sui dati se gli utilizzatori di filesystem diversi vengono messi in zone differenti.
- Minor tempo per effettuare l'avvio di sistema. Infatti il processo di scoperta delle periferiche viene eseguito per ogni zona su un insieme limitato di periferiche e non a livello globale su tutto il sistema (riduzione dello *span*).

Mentre si è già parlato nel Capitolo 2 del funzionamento delle VLAN e di come sia possibile generare zone separate tra loro, nulla si è detto sullo Zoning.

3.4.1 Zoning

Lo zoning può essere eseguito a due livelli di profondità:

1. **Target Level Zoning**, a sua volta da suddividersi in:
   (a) Hard Zoning
   (b) Soft Zoning
2. **LUN Level Zoning**

Prima di inoltrarsi nella spiegazione dello Zoning è necessario introdurre alcuni concetti sul World Wide Name (WWN).

World Wide Name (WWN)

World Wide Name (WWN) (vedi figura 3.23) è il nome della convenzione sui nomi per i componenti Fibre Channel e SCSI. Ad esso sono associate altre due sigle: il World Wide Port Name (WWPN) e il World Wide Node Name (WWNN). Il World Wide Port Name (WWPN) è un indirizzo univoco, che viene dato in fase di costruzione alle porte dei componenti SCSI e Fibre Channel. L'indirizzo è formato da 64 bit, una parte è fissa e identifica l'azienda costruttrice della periferica, mentre l'altra viene gestita e decisa dall'azienda. In ambito SAN può essere considerato il corrispettivo di un indirizzo MAC in ambito Ethernet. Il World Wide Node Name (WWNN) è invece un identificativo che appartiene ad un nodo della SAN, per esempio un server. Qual'è quindi la differenza tra WWPN e WWNN? Prendiamo come esempio un server che possiede due adattatori di rete Fibre Channel. Questi avranno WWPN differenti, dal momento che ognuno identifica univocamente la porta, ma WWNN uguale, visto che appartengono allo stesso server.

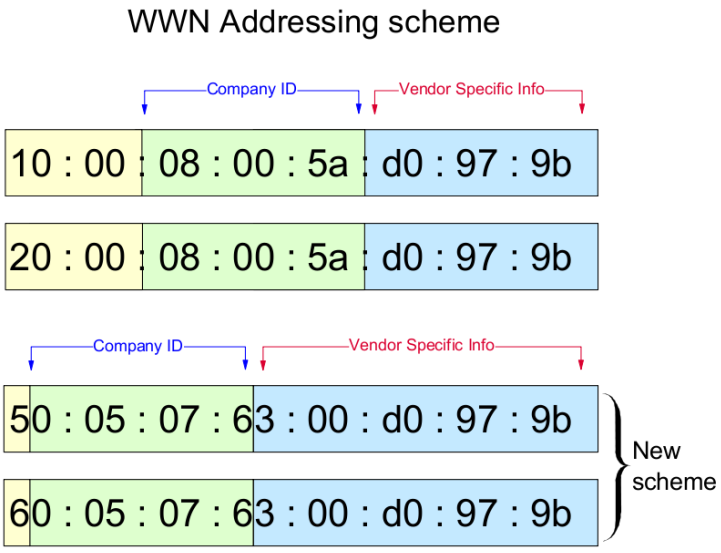

Figura 3.23: Schema di indirizzamento di un World Wide Name [fonte: [17]]

Hard Zoning

L'Hard Zoning si effettua a livello di switch della SAN (livello 2) programmandoli in modo da associare ad ogni server una o più porte collegate ai sistemi di memorizzazione (vedi figura 3.24). Per quel server, in pratica, è come se le altre porte degli switch non esistessero e, in questo modo, può raggiungere solamente un sottoinsieme delle risorse disponibili. Esistono alcuni problemi dati da questa configurazione. Immaginiamo che un server che è collegato alla porta 1 di uno switch appartenente alla zona 1 debba essere spostato sulla porta 2 della zona 2, da quel momento il server cambia zona di appartenenza e non può più, previa riconfigurazione, accedere alla zona 1. Questo esempio porta alla luce anche un'altra problematica di sicurezza. Mettiamo per esempio che qualcuno scolleghi un server A dalla porta 1 di uno switch appartenente alla zona 1 e ci colleghi un server B. A questo punto il server B diventerebbe parte della zona 1 e potrebbe accedere alle sue risorse. Per risolvere questo problema di sicurezza è possibile effettuare il *port binding*.
Il *port binding* consiste nell'assegnare ad una porta dello switch il WWPN della periferica che vogliamo si possa collegare su quella porta, a questo punto solo il traffico proveniente da quella periferica verrà accettato, se essa viene sostituita con un'altra questa possiederà un WWPN differente e quindi il suo traffico verrà scartato.

Soft Zoning

Il Soft Zoning anch'esso si effettua a livello di switch attraverso l'utilizzo di alias e World Wide Name (vedi figura 3.25). Il primo passo da compiere è quello di assegnare a tutti i componenti della SAN, che possiedono un WWPN, un alias e di registrare le associazioni su una tabella. Come si può immaginare questo implica che ogni switch possiederà molteplici alias, uno per porta per la precisione. Una volta fatto ciò è possibile creare un ulteriore tabella per mantenere le associazioni tra le zone e gli alias dei componenti che ne fanno parte. Come si può notare ora la porta di uno switch può ap-

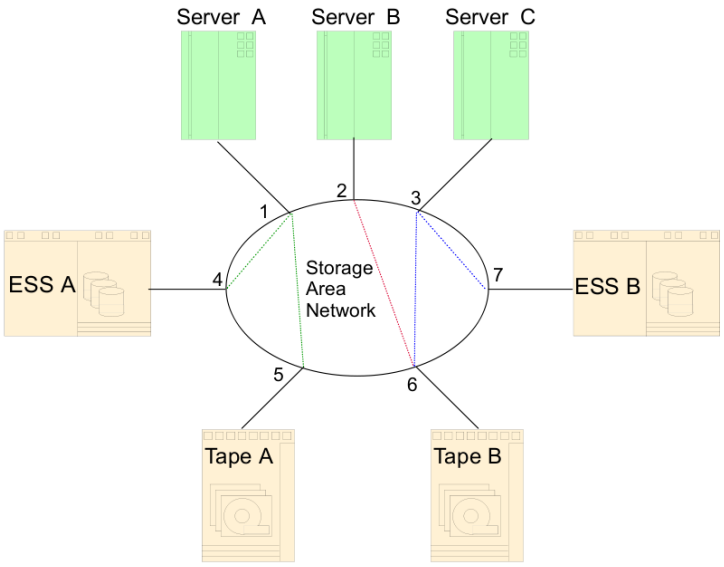

Figura 3.24: Hard Zoning configurato a livello di porte dello switch [fonte: [17]]

partenere a più zone, cosa che non era possibile con l'Hard Zoning. Inoltre se un server viene ricollegato su una porta dello switch diversa, continua a vedere le zone che gli sono di competenza, perché le sue zone non dipendono dalla porta dello switch a cui è collegato, ma dal WWPN del suo adattatore; questo chiaramente comporta flessibilità in caso di riposizionamento di server, ma allo stesso tempo rigidità in caso qualche componente della SAN si rompa, dal momento che è necessario andare ad aggiornare la tabella degli alias con i nuovi WWPN del nuovo componente che ha sostituito quello guasto, se si vuole che possa vedere ed accedere alle sue zone.

Nel Soft Zoning a differenza dell'Hard Zoning, c'è un problema riguardante la sicurezza, dal momento che abbiamo la possibilità di raggiungere, partendo dalla propria zona di appartenenza, anche le risorse che appartengono ad altre zone senza averne l'abilitazione. Ovviamente questa operazione è difficile da compiere, perché si dovrebbe conoscere l'indirizzo esatto della risorse appartenenti alla altre zone, ma non impossibile. Questo modello di sicurezza viene detto "sicurezza tramite segretezza" (*security through obscurity*) ed è un principio che si basa sul controverso assunto che tenere segreto il funzio-

namento interno di un sistema lo renda più sicuro, dato che un eventuale attaccante avrebbe maggiore difficoltà nella scoperta di eventuali falle del sistema stesso.

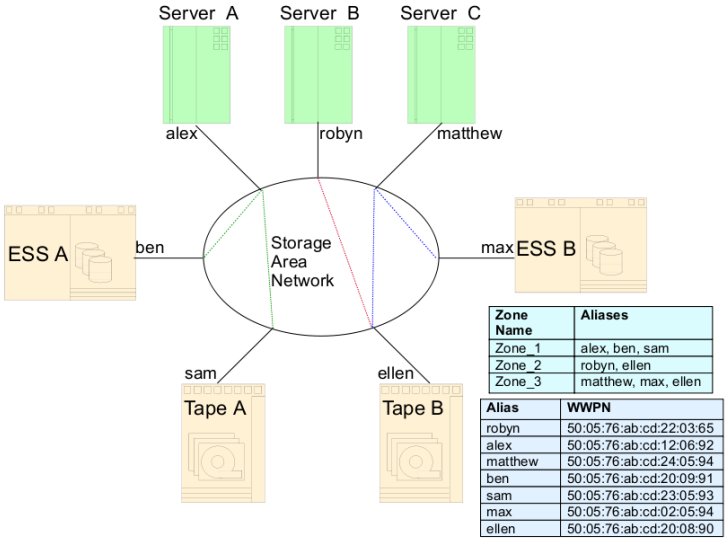

Figura 3.25: Soft Zoning basato sui WWN delle periferiche [fonte: [17]]

LUN Level Zoning (Masking)

Il LUN Level Zoning (LLZ) viene gestito a livello di componenti di alto livello (livello 4).
Può essere effettuato:

- A livello di server esternamente alla SAN.
- A livello di controller RAID internamente alla SAN.
- A livello di traffico intermedio tra server e periferiche di memorizzazione appartenenti alla SAN, utilizzando nella rete della SAN uno switch che permette LUN Level Zoning.

LLZ a livello server

La forma più facile di LUN Zoning viene effettuata a livello di server. Ogni

server possiede delle schede Fibre Channel che possono essere impostate in modo che oscurino le LUN che non rispettano determinate condizioni e di fatto creando delle zone. La scelta di questa tipologia di LUN Zoning risulta ottima quando abbiamo: pochi server, molti supporti di memorizzazione omogenei tra loro e i server che accedono alla SAN sono considerati affidabili. Le motivazioni di questi tre requisiti sono semplici. Primo, perché più server abbiamo più dobbiamo spendere tempo a configurarli e la situazione potrebbe diventare poco gestibile, secondo, perché supporti di memorizzazione eterogenei possono avere una gestione del LUN Zoning non compatibile tra loro, terzo, perché se i server non sono considerati affidabili potrebbero riconfigurare le loro schede di rete ed accedere alle LUN che non gli sono di competenza.

LLZ a livello di controller RAID

Questa forma di LUN Zoning può essere effettuata a livello di controller RAID, impostando il controller in modo che permetta o neghi l'accesso a determinate LUN, a seconda di chi ne fa richiesta, creando di fatto delle zone. Non tutti i venditori però includono nei loro prodotti la possibilità di gestire il LUN Zoning a livello di controller RAID. Se questa possibilità ci è preclusa è necessario rifarsi ad una delle altre due tipologie di LLZ.
Questa opzione di zoning è però molto accattivante per quelle SAN composte da molti server e pochi array di dischi in RAID, perché non è necessario mettere mano alla configurazione dei server. Inoltre questa tipologia di LUN Zoning può convivere piacevolmente con l'hard zoning, dal momento che mentre l'hard zoning divide il traffico a livello di porta dello switch, il LUN Zoning permette un ulteriore livello di granularità, andando a separare il traffico che passa per una singola porta in ulteriori sotto zone. Facciamo un esempio. Se una porta di uno switch permette attraverso hard zoning a due server di accedere a quattro LUN è possibile celare attraverso il LUN Zoning due di queste ad un server e due all'altro.

LLZ a livello di Switch che permette LUN Masking

Una soluzione intermedia è quella di comprare uno switch, che permetta il LUN Zoning e piazzarla nella rete SAN, tra i server e i supporti di memorizzazione (vedi figura 3.26). A questo punto siamo riusciti ad ottenere una soluzione intermedia, che non dipende né dai server né dai supporti di memorizzazione.

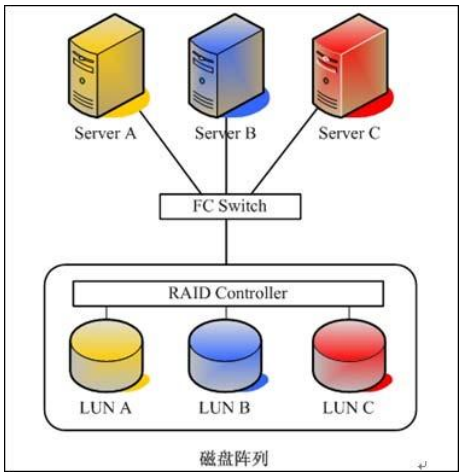

Figura 3.26: Lun Level Zoning a livello di switch

3.5 Considerazioni sul capitolo

In questo capitolo abbiamo trattato l'argomento relativo a VMWare ed ai sistemi di memorizzazione che supporta; ci siamo addentrati poi all'interno di due macro categorie di topologie di rete come la SAN ed i NAS, utilizzatissime in ambito aziendale, svelandone i principali punti di forza e debolezza. Si è anche dato uno sguardo a tutta una serie di tecnologie utilizzate per realizzare queste topologie di rete come Fibre Channel, iSCSI, Fibre Channel over Ethernet, ecc...

Tutta questa panoramica è servita per andare a delineare la complessità e la moltitudine di scelte implementative che possono essere effettuate nella realizzazione di un'architettura di memorizzazione condivisa, dimostrando

quanto questa scelta incida anche sulle prestazioni finali del sistema. Se da una parte un sistema di memorizzazione condiviso può anche non essere utilizzato, dall'altro diventa un requisito fondamentale per quanto riguarda la possibilità di sfruttare tutte le funzionalità avanzate di VMWare e per quanto riguarda l'efficienza del sistema di memorizzazione.
In questa ottica è necessario comprendere il grande sforzo che deve essere compiuto per analizzare le prestazioni di ogni singola configurazione possibile, visto che in questo senso, i risultati di un test su NAS non possono essere d'aiuto in caso di utilizzo di SAN in Fibre Channel, a meno di non venire utilizzati puramente a livello comparativo. Il campo degli esperimenti può quindi essere esteso in futuro anche ad altre configurazioni come per esempio il Fibre Channel e l'iSCSI, di cui comunque si era provata una configurazione, senza però poter eseguire i test per mancanza di hardware.

Capitolo 4

Valutazione sperimentale

Siamo ora giunti alla trattazione della parte pratica di questa tesi. Lo scopo è quello già citato all'interno del capitolo introduttivo: attraverso l'utilizzo di alcuni benchmark si andranno ad analizzare i colli di bottiglia e le prestazioni di un sistema di virtualizzazione, basato su ESXi 4, all'aumentare del carico e del numero di macchine virtuali ospitate al suo interno. Il tutto è stato concepito principalmente come un caso di studio, dal momento che non era possibile usufruire dell'hardware e della potenza di un centro di calcolo. Per quanto riguarda la parte hardware dell'ambiente di lavoro le componenti utilizzate, che vanno da schede madri, alla RAM, alle schede di rete, sono versioni desktop a basso costo e non server. Come rete è stata utilizzata una rete LAN Gigabit Ethernet, dato l'elevato costo dei componenti in fibra ottica necessari per uno studio sul Fibre Channel. Una volta fatte queste precisazioni sono stati assemblati e collegati tra di loro tre PC attraverso l'utilizzo di un switch. Per rendere più verosimile l'esperienza di lavoro e per trarre beneficio dalla riduzione del dominio di broadcast è stato introdotto l'utilizzo di VLAN all'interno della configurazione di rete. Dei tre PC disponibili, ne sono stati utilizzati due come client per effettuare le richieste su un terzo, che era la macchina server di cui vogliamo misurare le prestazioni. Ed ecco ora una foto dell'architettura virtualizzata in fase iniziale di design (vedi figura 4.1) ed una al termine del lavoro (vedi figura 4.2). Abbiamo introdotto

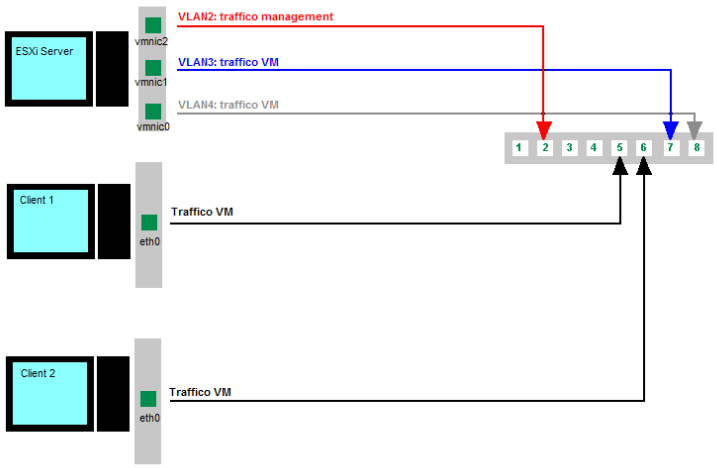

Figura 4.1: Design iniziale dell'architettura da realizzare

solamente alcuni concetti legati all'ambiente degli esperimenti. Andremo ora ad analizzarlo con cura partendo da:

- Specifiche hardware dei PC.
- Configurazione della rete e delle VLAN.
- Benchmark e software utilizzato durante i test.

4.1 Specifiche hardware dei PC

È importante precisare, che come client, sono state utilizzate due macchine con hardware differente, di cui avevo la disponibilità, dove una è leggermente più lenta dell'altra. Questa disparità tra i due client non è voluta e il caso ottimale sarebbe stato quello di effettuare i test con macchine con hardware identico. Questo ha portato a dei risultati dove il client più veloce termina in anticipo, rispetto a quello più lento, in maniera proporzionale al carico della macchina server. Per tutti quei test, dove la macchina server è poco sotto sforzo, questo comportamento si nota bene. Al crescere del

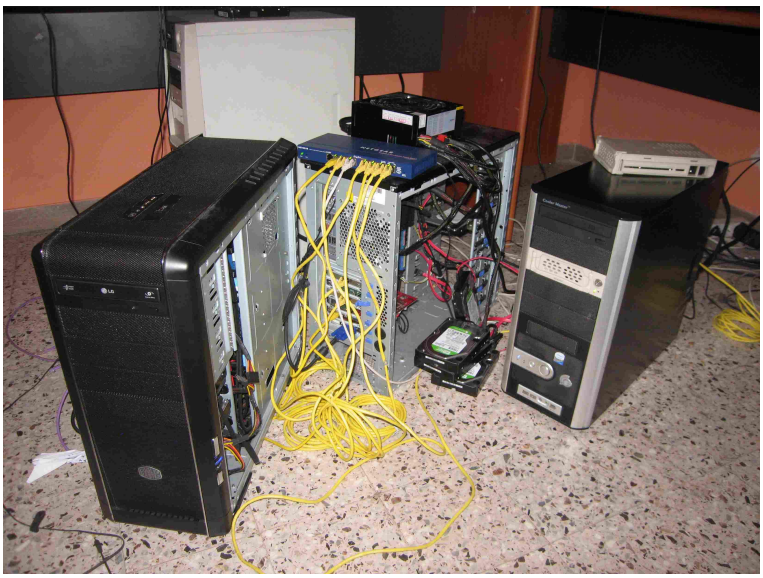

Figura 4.2: Foto della realizzazione dell'architettura

carico si nota sempre meno, fino ad arrivare ad un punto dove i due client terminano insieme. Tutto questo perché il congestionamento della macchina server la porta in uno stato dove non riesce a servire più richieste del client veloce rispetto a quello lento. Questo comportamento quindi non incide in maniera significativa sui test con molto carico, che sono proprio quelli a cui si è interessati.

Macchina ESXi Server

Una macchina (vedi figura 4.3) con queste specifiche hardware:

```
CPU: Intel Core i7 CPU 950 @ 3.07GHz
     Quad Core
     Hyper Threading Technology
     Intel Virtualization Technology

Scheda Madre: Asus Rampage II Extreme
              6 porte SATA @ 3 Gb/s
```

```
RAM: Corsair Dominator 12 GB RAM DDR3 Triple Channel 1600 MHz
     (2 Scatole di 3 moduli da 2 GB l'uno)

Dischi: 4xWestern Digital SATA 750 GB 7200 RPM 32 MB Cache
        1xMaxtor SATA 500 GB 7200 RPM 32 MB Cache

Schede di rete: 3xSchede di rete 1Gbps
                2xIntel Pro 1000 PT slot PCI-Express 1x
                1xIntel Pro 1000 GT slot PCI
```

Figura 4.3: Foto della macchina server

I 4 dischi da 750 GB sono utilizzati come spazio di memorizzazione per le macchine virtuali dei nostri test (sono i quattro datastore utilizzando la terminologia di VMWare) e sono formattati con VMFS. Il quinto disco da 500 GB invece è quello sul quale è installato il sistema ESXi 4.0.0 Releasebuild-171294. Nel BIOS è stata abilitata la tecnologia Intel-VT.

Macchina Client 1

Una macchina (vedi figura 4.4) con queste specifiche hardware:

```
CPU: Intel Core i5 CPU 750 @ 2.67 Ghz
     Quad Core
     Intel Virtualization Technology

Scheda Madre: Asus 1156 P7P55D
              6 porte SATA @ 3 Gb/s

RAM: Kingston 4 GB RAM DDR3 Dual Channel 1600 MHz
     (1 scatola di 2 moduli da 2 GB l'uno)

Dischi: 1xMaxtor 500 GB SATA 7200 RPM 32 MB Cache

Schede di rete: 2xSchede di rete 1 Gbps
                2xIntel Pro 1000GT slot PCI
```

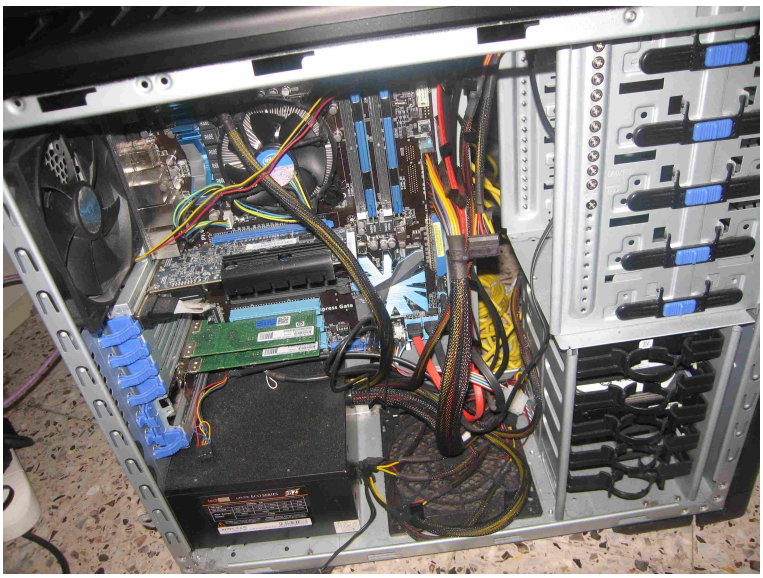

Figura 4.4: Foto della prima macchina client

Su questa macchina è stato installato un sistema operativo Xubuntu 9.10.

Macchina Client 2

Una macchina (vedi figura 4.5) con queste specifiche hardware:

```
CPU: Intel Core 2 Duo E6600 @ 2.40GHz
     Dual Core

Scheda Madre: Asus Commando
              6 porte SATA @ 3Gb/s

RAM: OCZ 4GB RAM DDR2 Dual Channel 400 MHz
     (1 scatola di 2 moduli da 2 GB l'uno)

Dischi: 1xMaxtor 500 GB SATA 7200 RPM 32 MB Cache

Schede di rete: 2xSchede di rete 1 Gbps
                2xMarvell Yukon Gigabit NIC
```

Su questa macchina è stato installato un sistema operativo Xubuntu 9.10.

Strumentazione di rete

Ecco la strumentazione[1] di rete utilizzata (vedi figura 4.6):

```
Cavi Cat-5e UTP

Switch a 8 porte Netgear GS108T che possiede queste funzionalità:
   Porte full duplex con controllo di flusso a 10/100/1000 Mbit/s
   Supporto per il Link Aggregation
   Supporto del protocollo 802.1Q riguardante le VLAN
```

[1] http://www.netgear.it/prodotti/ft_pdf/gs108t.pdf

Figura 4.5: Foto della seconda macchina client

Figura 4.6: Foto dello switch

4.2 Configurazione della rete e delle VLAN

Per simulare in parte un ambiente virtualizzato si è deciso di separare la rete in tre zone tramite l'utilizzo di VLAN. Ad ognuna di queste zone è stato assegnato un VLAN ID. Il `VLAN ID 2` è stato assegnato alla rete di *management*, cioè al traffico di rete generato da ESXi, per gestire il sistema che, anche se in minima parte, non doveva andare ad influire sul traffico delle macchine virtuali utilizzate durante i test. Il `VLAN ID 3` e il `VLAN ID 4` sono stati invece assegnati a due gruppi separati di macchine virtuali, per fare in modo di ridurre il loro dominio di broadcast e di separare logicamente il loro traffico di rete. La rete della macchina server con ESXi è stata configurata

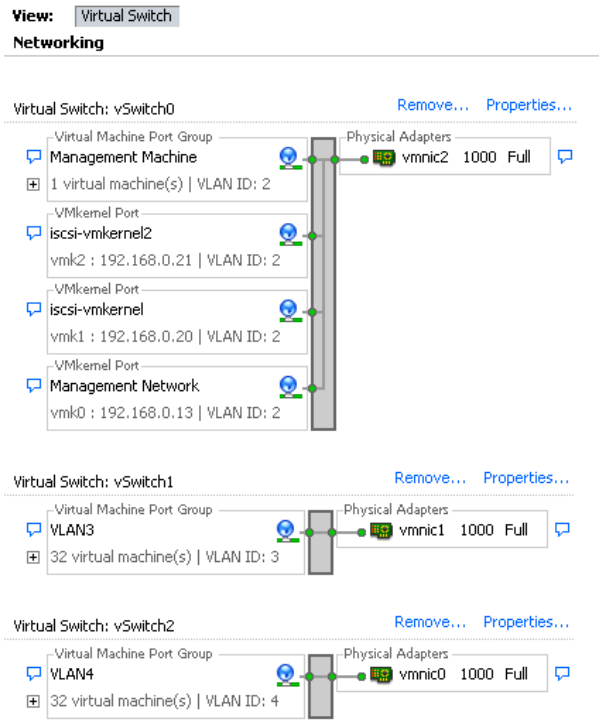

Figura 4.7: Configurazione del vSwitch sulla macchina server

in questo modo (figura 4.7):

- Il virtual switch 0 gestisce la rete di management di ESXi con `VLAN ID` 2 attraverso vmk0.

- Il virtual switch 1 gestisce le macchine virtuali cha appartengono alla VLAN3 con `VLAN ID 3`.

- Il virtual switch 2 gestisce le macchine virtuali cha appartengono alla VLAN4 con `VLAN ID 4`.

All'interno dello switch invece si è dovuto provvedere alla configurazione della VLAN attraverso tre passi: la definizione delle VLAN (VLAN Configuration), la definizione di quale insieme di regole applicare al traffico di ingresso in una particolare porta dello switch (PVID Configuration) e la definizione degli insieme di regole da applicare al traffico di ogni VLAN (VLAN Membership).

4.2.1 Configurazione VLAN

Innanzi tutto abilitiamo nello switch il supporto allo standard delle VLAN IEEE 802.1Q. A questo punto definiamo tre VLAN differenti (vedi figura 4.8) con `VLAN ID 2` per internet e rete di *management* di ESXi, `VLAN ID 3` e `VLAN ID 4` per le macchine virtuali.

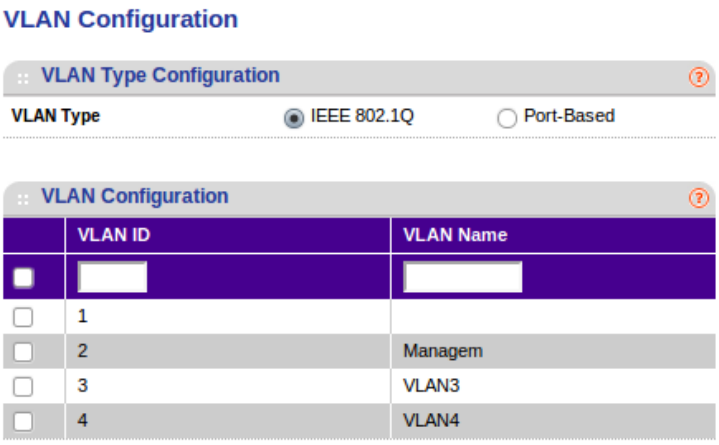

Figura 4.8: Configurazione delle VLAN sullo switch

4.2.2 Port Virtual ID (PVID)

Definiamo ora quali set di regole devono essere utilizzate per i pacchetti che entrano in una particolare porta (PVID) dello switch (vedi figura 4.9). È molto importante capire come funziona il PVID, dal momento che tutto il traffico che entrerà attraverso quella porta, utilizzerà l'insieme di regole (che spiegheremo successivamente nella sezione VLAN Memebership), definito per quella porta per trovare l'uscita. Nel nostro caso abbiamo che il traffico in ingresso sulle porte `g1 g2 g3 g4` dello switch fanno parte della `VLAN 2`. Le porte `g5` e `g7` fanno parte della `VLAN 3` mentre le porte `g6` e `g8` fanno parte della `VLAN 4`.

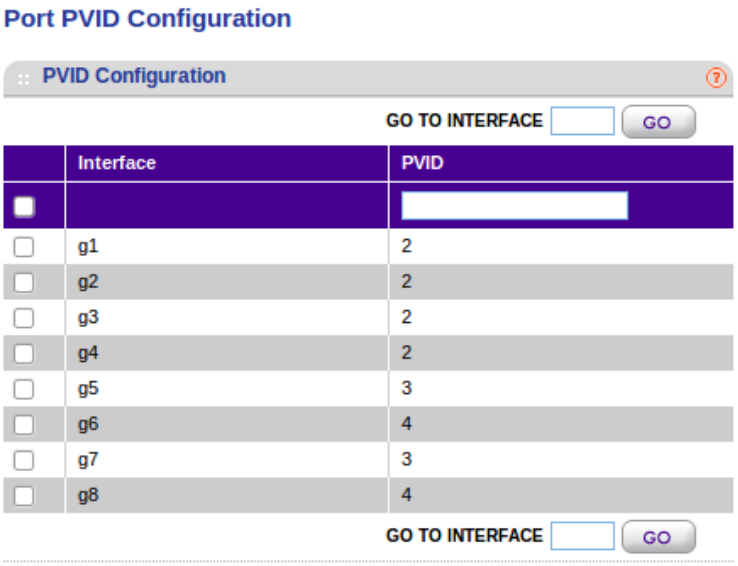

Figura 4.9: Configurazione dei Port Virtual ID sullo switch

4.2.3 VLAN Membership

`VLAN Membership 2` (vedi figura 4.10) regole da applicare al traffico che entra dalle interfacce con `PVID 1 2 3` e `4`. Tra queste abbiamo la `porta 1` e `3`, che sono collegate a due pc che necessitano l'accesso a VMWare ESXi, attraverso l'utilizzo del client di VMWare. La `porta 4` va al router e ad

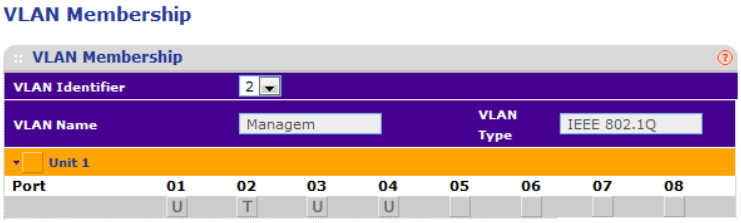

Figura 4.10: Configurazione dell'appartenenza alla VLAN 2

internet, la `porta 2`, che è anche l'unica *tagged*, consegna il traffico verso la macchina server con sopra ESXi. La porta è *tagged* per permettere al traffico che arriva da internet (e quindi dalla `porta 4` cioè *untagged*) di ricevere un tag con `ID 2`.

`VLAN Membership 3` (vedi figura 4.11) regole da applicare al traffico che entra dalle interfacce con PVID 5 e 7. Attraverso queste due porte abbiamo il

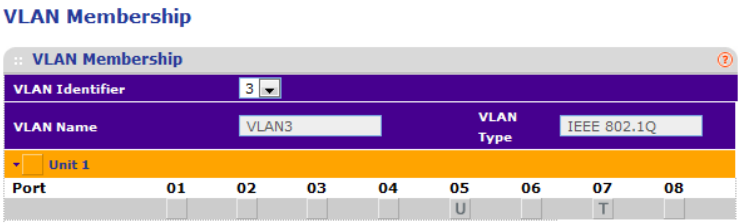

Figura 4.11: Configurazione dell'appartenenza alla VLAN 3

traffico delle macchine virtuali con `ID 3`. La porta 5 è quella del client che effettuerà le richieste, mentre la 7 è la porta con cui si arriva ad ESXi sul vSwitch con VLAN 3.

`VLAN Membership 4` (vedi figura 4.12) regole da applicare al traffico che entra dalle interfacce con PVID 6 e 8. Attraverso queste due porte abbiamo il traffico delle macchine virtuali con `ID 4`. La porta 6 è quella del client che effettuerà le richieste, mentre la 8 è la porta con cui si arriva ad ESXi sul

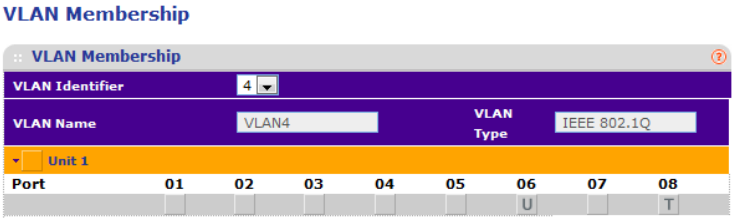

Figura 4.12: Configurazione dell'appartenenza alla VLAN 4

vSwitch con VLAN 4.

4.3 Benchmark e software utilizzato durante i test

Sulle macchine client è stato scelto di installare **Xubuntu 9.10**[2] proprio per la leggerezza di questo sistema operativo. Sulla macchina server invece è stata installata una versione di prova di **ESXi 4.0.0 Releasebuild-171294**[3]. Le macchine virtuali gestite da ESXi dispongono di un disco virtuale da 8 GB, di 512 MB di RAM, con sopra installato il sistema operativo **JeOS 8.04.3 LTS**[4]. Questo sistema operativo è una distribuzione Server Edition di Ubuntu studiata appositamente per il suo utilizzo nell'ambito della virtualizzazione; tra i vantaggi che fornisce:

- Immagine ISO d'installazione di solo 100 MB.
- Meno di 300 MB di spazio occupato appena finito di installare il sistema.
- Utilizzo di un kernel modificato per sfruttare al meglio un ambiente

[2] http://xubuntu.org/
[3] https://www.vmware.com/tryvmware/?p=esxi
[4] https://help.ubuntu.com/8.04/serverguide/C/jeos.html

virtualizzato (le macchine utilizzano un kernel di tipo -virtual Kernel 2.6.24).

- Ottimizzazione per sistemi VMWare ESX, VMWare Server and KVM.
- Solamente 128 MB di RAM minima necessaria per l'esecuzione.
- Nessun ambiente desktop precaricato.

Per quanto riguarda i test, questi ultimi sono stati eseguiti con dei benchmark open source: Sysbench e Apache Benchmark. Le versioni del software utilizzato sono le seguenti:

- Sysbench 0.4.10[5]
- Apache 2.2.8[6] e il suo tool di benchmark "ab"[7] incluso in `apache-utils`
- MySQL 5.0.51a-3ubuntu5.4 (Ubuntu)[8]

4.3.1 Sysbench

Sysbench[9] è un benchmark che permette di generare varie tipologie di carico per mettere sotto sforzo componenti di un sistema come: disco fisso, RAM, rete, ecc... Il test, tra quelli forniti, a cui si è interessati, è quello che ci permette di misurare le prestazioni di un database MySQL. Il test è della tipologia On Line Transaction Processing; effettua una serie di operazioni di lettura e scrittura, nell'ordine di dimensioni che vanno da 2 KB a 16 KB, eseguendo numerose piccole transazioni. Il carico del test, sui client, è stato impostato su 10. 000 operazioni da eseguire per ogni macchina virtuale accesa, con un livello di concorrenza pari a 16 thread su un database di 10. 000. 000 record.

[5] http://sysbench.sourceforge.net/
[6] http://www.apache.org/
[7] http://httpd.apache.org/docs/2.0/programs/ab.html
[8] http://www.mysql.com/?bydis_dis_index=1
[9] http://sysbench.sourceforge.net/

4.3.2 Apache Benchmark

Apache Benchmark[10] ("ab") invece è stato utilizzato per analizzare le prestazioni di un web server Apache, sottoposto a numerose richieste di lettura di pagine html. L'unico difetto riscontrato da questo programma è il fatto di permettere solamente la richiesta di una stessa pagina html, per più volte consecutive. Questo non era ciò che si voleva, dal momento che era impossibile evitare completamente che il sistema sfruttasse meccanismi di caching (disco/sistema), ed una volta che la pagina era nella cache, i test risultavano troppo performanti. Per porvi rimedio sono stati creati degli script che permettessero il lancio concorrente di più istanze di "ab". Su ogni macchina virtuale si è proceduto alla clonazione di 10.000 pagine web, partendo da una pagina web sorgente, con il risultato di avere pagine di uguali dimensioni.
A questo punto è stato possibile fare si che ogni istanza di "ab" richiedesse una singola pagina, differente da quella richiesta dalle altre istanze, per un totale di 10.000 richieste per macchina virtuale, con un grado di concorrenza pari anche questa volta a 16 thread.

[10] http://httpd.apache.org/docs/2.0/programs/ab.html

Capitolo 5

Test

I test, come detto in precedenza, sono stati concepiti per analizzare le prestazioni di un sistema su cui esegue VMWare ESXi 4.0. Cercheremo quindi di capire, come a seconda delle configurazioni, possono cambiare queste prestazioni e nello specifico si cercherà di vedere come reagirà il sistema con l'aumentare delle macchine virtuali e del carico di lavoro. Ogni macchina virtuale presente sul server, infatti, verrà messa sotto sforzo attraverso richieste inviate da parte di Sysbench e Apache Benchmark (che abbiamo descritto poco sopra). Con l'aumentare del carico di lavoro verranno registrate attraverso dei grafici le variazioni dei componenti critici del sistema come la rete, il disco, la RAM e la CPU. Questi grafici serviranno per capire possibili colli di bottiglia delle varie tipologie di test. I test come descritto nel capitolo precedente sono di tre tipologie:

1. Apache Benchmark che accede in lettura a pagine http di 1 KB su web server Apache.

2. Apache Benchmark che accede in lettura a pagine http di 500 KB su web server Apache.

3. Sysbench che accede in lettura, e principalmente in scrittura, attraverso transazioni a dati presenti su un database gestito da un motore MySQL.

Per ogni tipologia si parte da una base inferiore di 8 macchine virtuali, salendo via via, di 8 in 8, fino ad arrivare a 64. Ad ogni cambio di configurazione la macchina server con ESXi è stata spenta per evitare ogni forma di cache. Infatti non si è interessati al caso di funzionamento ottimale, quando i dati sono in RAM o in cache, ma a quello pessimo, dove il sistema è costretto continuamente ad accedere al disco. Per tutti i test effettuati con macchine virtuali con CPU da 8 core, non è stato possibile accenderne contemporaneamente più di 32 per mancanza di RAM disponibile, nonostante la macchina server ne possieda ben 12 GB. ESXi necessita infatti di un quantitativo riservato di RAM della macchina server, dipendente dal numero di CPU virtuali e quantitativo di memoria che vengono assegnati ad una macchina virtuale per la sua gestione (vedi figura 5.1). Nel caso ESXi si trovi impossibilitato ad allocare questa memoria lancia un messaggio di errore in cui avvisa che non è stato possibile aggiungere ulteriori macchine virtuali. Ricordiamo nuovamente che nei test ogni macchina virtuale è stata configurata con 512 MB di RAM.

I test effettuati sono divisi in tre sezioni: Apache Benchmark con pagine da 1 KB, Apache Benchmark con pagine da 500 KB e Sysbench. Per la tesi sono stati effettuati all'incirca un totale di 72 test; 3 tipi di test (Apache 1 KB, 500 KB, Sysbench), per 3 configurazioni sui core (1, 4 e 8 core), per 8 possibili configurazioni sulle macchine virtuali (8, 16, 24, 32, 40, 48, 56, 64). Ognuno di questi test ha generato un minimo di 4 grafici: uno per la CPU, uno per la RAM, uno per la rete ed uno per il disco. Un minimo di 4 grafici, dal momento che un grafico di ESXi non registra più di un ora di esecuzione e quindi per ogni test che durava più di un ora è stato necessario salvare 4 grafici ulteriori per ogni successiva ora di esecuzione (il massimo raggiunto è stato con il test Sysbench con 64 macchine virtuali, durato di 260 minuti, di cui è stato necessario salvare 20 grafici). Dal momento che i grafici ottenuti sono ben 344 si è deciso di non includerli nella tesi, ma di mostrarne alcuni in un appendice. Per tutti coloro che volessero esaminare il contenuto di questi

Memory Assigned (MB)	1 vCPU	2 vCPUs	3 vCPUs	4 vCPUs	5 vCPUs	6 vCPUs	7 vCPUs	8 vCPUs
256	113	159	201	241	293	334	375	416
512	117	165	206	247	303	344	385	426
1024	124	176	217	258	322	363	404	446
2048	138	198	239	281	360	401	443	484
4096	166	242	284	325	437	479	520	561
8192	222	331	373	414	591	633	675	716
16384	335	508	550	592	900	943	986	1028
32768	560	863	906	949	1516	1559	1603	1647
65536	1011	1572	1616	1660	2746	2792	2838	2884
131072	1912	2990	3036	3083	5220	5273	5326	5379
262144	3714	5830	5885	5938	10142	10205	10267	10329

Figura 5.1: Grafico che mostra i vari sprechi di memoria, necessari ad ESXi per avviare una macchina virtuale, in funzione di CPU virtuali e RAM assegnati ad essa [fonte [10]]

grafici è possibile contattarmi per posta elettronica[1] e provvederò alla loro spedizione.
Nelle varie sezioni dei test, per prima cosa, verrà mostrata una tabella riassuntiva contenente i vari consumi realtivi alla CPU, al disco, alla rete e al tempo necessario per eseguire i test. Una volta fatto ciò verranno mostrati i grafici del'andamento reale dei test rispetto ad un tempo ideale. Questo tempo ideale è basato su queste regole:

- per i test con 8 macchine virtuali il tempo ipotetico e quello reale coincidono e lo chiameremo tempo ideale iniziale.
- per i successivi test il tempo ideale è calcolato così:
  `( numero VM / 8 ) * tempo ideale iniziale`

Questo tempo ideale sta ad indicare una crescita lineare dei tempi rispetto a quello con 8 macchine virtuali; sarebbe in poche parole l'andamento auspicabile per un sistema scalabile. Fatto ciò, verranno poi mostrati due grafici

[1] amici@cs.unibo.it

conclusivi, uno con i tempi di esecuzione per i test a 1, 4 e 8 core, e l'altro con lo scostamento in minuti dal tempo ideale.

Ora andiamo ad esaminare i vari casi di studio ricordando che all'interno delle tabelle dei risultati:

- il simbolo - indica un valore talmente piccolo da poter essere considerato trascurabile.
- il simbolo **X** indica che il sistema non è riuscito a portare a compimento il test per il troppo carico.

5.1 Caso numero 1: Apache Benchmark pagine da 1 KB

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	6.300	-	2.000	2 : 30
16	6.250	-	2.200	5 : 00
24	5.685	-	2.400	7 : 30
32	6.750	-	2.200	10 : 00
40	5.100	-	1.800	12 : 30
48	6.550	-	2.000	15 : 00
56	X	X	X	X
64	X	X	X	X

Tabella 5.1: Valori medi per il test Apache Benchmark 1 KB con 1 core

Questo test, tra i 3 che sono stati fatti, è in un certo modo quello meno rilevante, dato l'esigua dimensione delle pagine html da recuperare di solo 1 KB. Nonostante questo si possono già cominciare a delineare alcuni comportamenti ricorrenti nel corso anche degli altri test. Per prima cosa si può

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	5.625	-	2.000	2 : 30
16	6.000	-	2.200	5 : 00
24	7.000	-	2.400	7 : 30
32	3.400	-	2.200	9 : 00
40	8.000	-	1.800	14 : 00
48	X	X	X	X
56	X	X	X	X
64	X	X	X	X

Tabella 5.2: Valori medi per il test Apache Benchmark 1 KB con 4 core

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	6.300	-	1.960	2 : 30
16	7.500	-	2.200	5 : 00
24	7.500	-	2.400	7 : 30
32	8.500	-	2.000	10 : 00

Tabella 5.3: Valori medi per il test Apache Benchmark 1 KB con 8 core

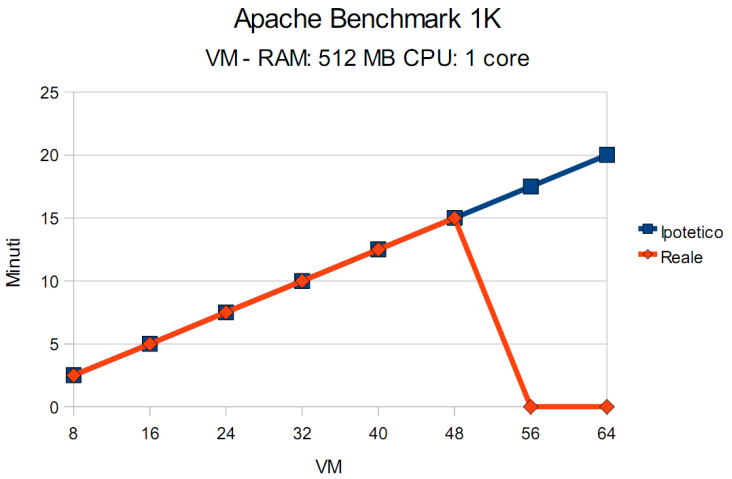

Figura 5.2: Grafico dei tempi ottenuti con i test Apache Benchmark 1 KB - 1 core

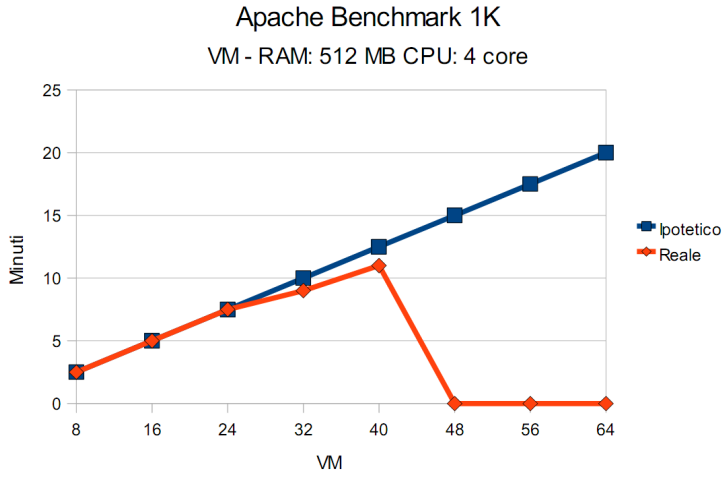

Figura 5.3: Grafico dei tempi ottenuti con i test Apache Benchmark 1 KB - 4 core

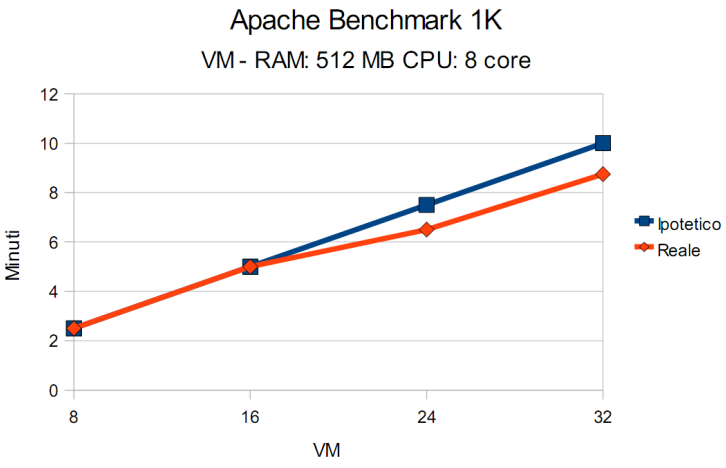

Figura 5.4: Grafico dei tempi ottenuti con i test Apache Benchmark 1 KB - 8 core

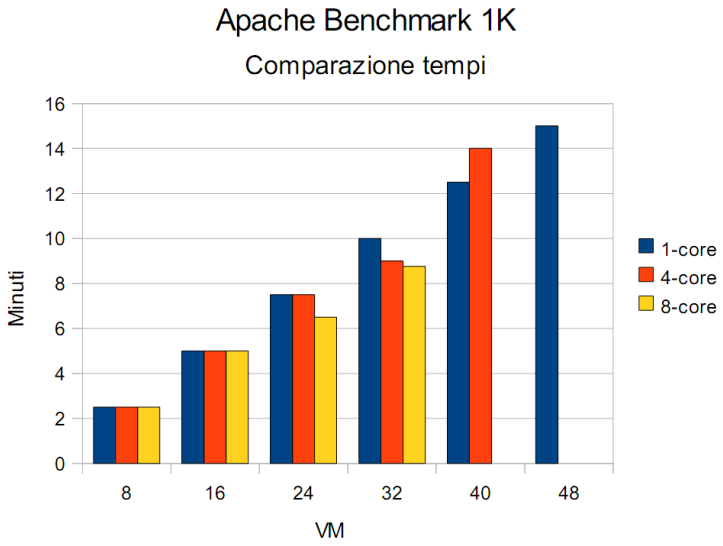

Figura 5.5: Grafico della comparazione dei tempi ottenuti con i test Apache Benchmark 1 KB a 1, 4 e 8 core

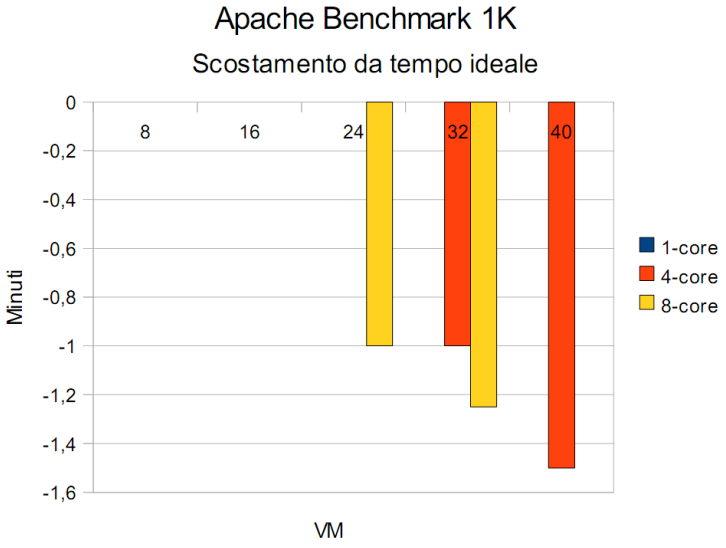

Figura 5.6: Grafico degli scostamenti dal tempo ideale (lineare) ottenuti con i test Apache Benchmark 1 KB a 1, 4 e 8 core

notare che fino a 40 macchine virtuali (32 per il caso con 8 core) i test procedono seguendo la linea ideale con scostamenti troppo piccoli per essere degni di nota. Il vero scalino si ha nel passaggio da 32 a 48 macchine per il test a 4 core e quello da 48 a 56 per quello a 1 core. Infatti, da questi momenti, i client cominciano a fare cadere le richieste a causa di timeout troppo lunghi. Questi due valori sono divenuti la soglia oltre la quale il server non riesce, causa il troppo carico, a soddisfare le richieste dei client. Per confermare questo andamento i test sono stati ripetuti anche con un benchmark diverso, e cioè HTTPERF[2], ma il risultato è rimasto invariato. Possiamo quindi affermare che alla soglia di 48 macchine con 4 core e 56 macchine con 1 core il sistema genera così tanto overhead nel cambiare di contesto tra le macchine e nel gestire le troppe richieste in arrivo, che non riesce più a servirle in maniera corretta e i client si vedono costretti a far cadere le connessioni. Risalta il fatto che la soglia per un sistema con macchine virtuali impostate con 1 core, sia successiva rispetto a quella per 4 core, e c'è un pò di rammarico per il fatto di non aver potuto testare l'andamento delle macchine con 8 core oltre le 32 unità (non raggiungendo quindi la soglia).
Inoltre, all'aumentare delle macchine virtuali, segue un innalzamento progressivo dell'utilizzo della CPU nei casi con 4 e 8 core. Questo è un comportamento che si ripeterà ancora durante i test di Apache Benchmark con pagine da 500 KB e Sysbench, ed è dovuto principalmente all'aumento dell'overhead, dato dalla gestione dello *scheduler* di ESXi per quanto riguarda l'assegnazione di cicli di CPU alle macchine con più core. Le macchine con 4 e 8 core risulteranno in seguito, quelle meno performanti so grossi carichi. La rete, il disco e la RAM per questo tipo di test influiscono molto poco, dato il loro basso utilizzo; è invece la CPU ad essere molto utilizzata e nello specifico risulta essere importante una bassa latenza nei tempi di risposta alle numerose richieste onde evitare timeout che troncano le connessioni.

[2]http://www.hpl.hp.com/research/linux/httperf/

5.2 Caso numero 2: Apache Benchmark pagine da 500 KB

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	2.250	37.500	37.500	27 : 30
16	2.350	37.000	37.000	52 : 30
24	2.250	35.000	35.000	80 : 00
32	2.500	32.500	32.500	117 : 30
40	X	X	X	X
48	X	X	X	X
56	X	X	X	X
64	X	X	X	X

Tabella 5.4: Valori medi per il test Apache Benchmark 500 KB con 1 core

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	2.600	37.500	37.500	27 : 30
16	2.700	38.000	38.000	52 : 00
24	7.500	32.500	32.500	87 : 30
32	8.000	30.000	30.000	118 : 00
40	X	X	X	X
48	X	X	X	X
56	X	X	X	X
64	X	X	X	X

Tabella 5.5: Valori medi per il test Apache Benchmark 500 KB con 4 core

Qesto test è simile a quello precedente ma con pagine da recuperare delle dimensioni di 500 KB. Si può subito notare che in questo caso il collo di bottiglia principale è dato dai dischi. Dato l'elevato numero di letture casuali ai quali i quattro dischi vengono sottoposti, si hanno prestazioni che vanno

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	3.000	37.500	37.500	27 : 30
16	3.500	37.000	37.000	52 : 30
24	12.500	32.500	32.500	87 : 30
32	9.000^3	32.500	32.500	117 : 30

Tabella 5.6: Valori medi per il test Apache Benchmark 500 KB con 8 core

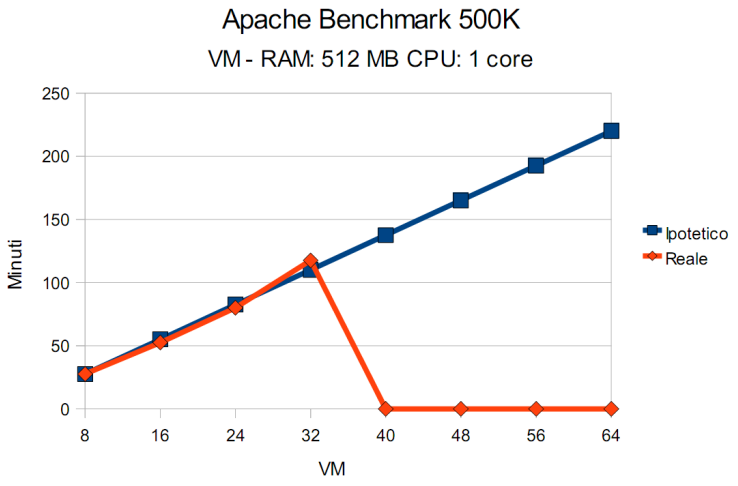

Figura 5.7: Grafico dei tempi ottenuti con i test Apache Benchmark 500 KB - 1 core

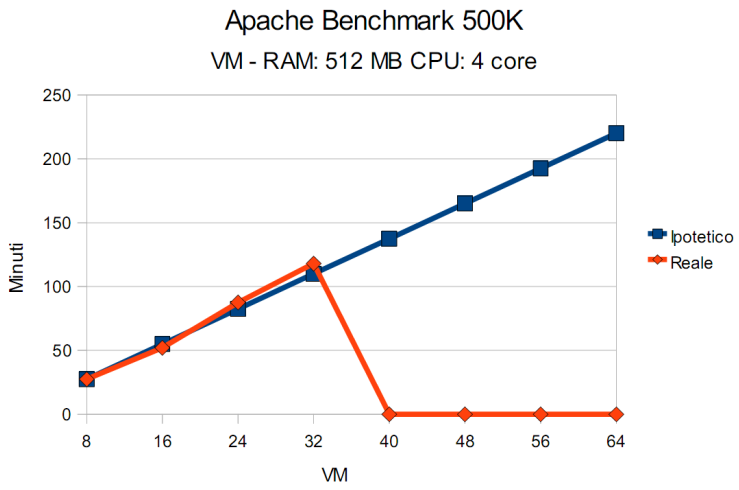

Figura 5.8: Grafico dei tempi ottenuti con i test Apache Benchmark 500 KB - 4 core

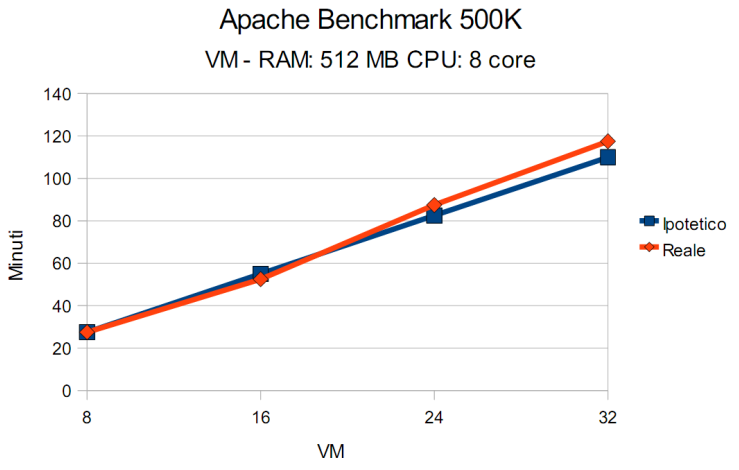

Figura 5.9: Grafico dei tempi ottenuti con i test Apache Benchmark 500 KB - 8 core

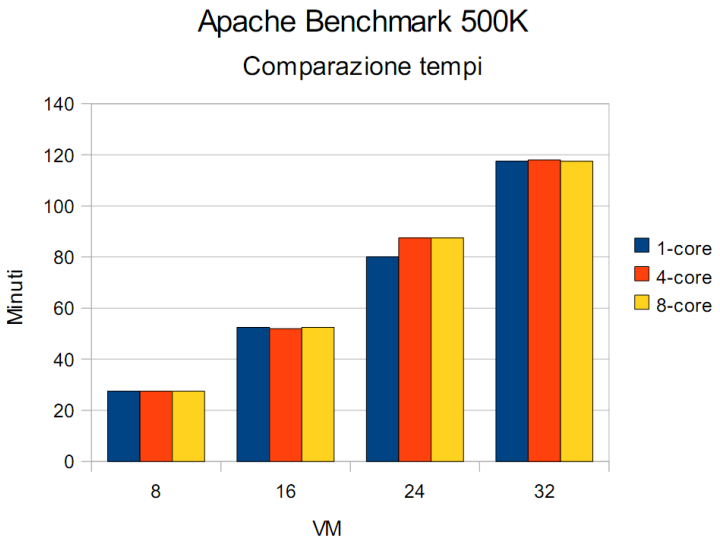

Figura 5.10: Grafico della comparazione dei tempi ottenuti con i test Apache Benchmark 500 KB a 1, 4 e 8 core

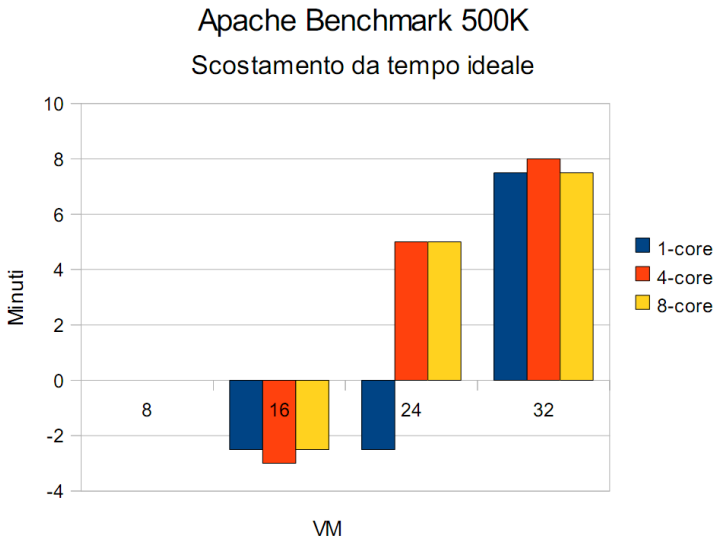

Figura 5.11: Grafico degli scostamenti dal tempo ideale (lineare) ottenuti con i test Apache Benchmark 500 KB a 1, 4 e 8 core

dai 37.500 KBps, nei test con poche macchine, ai 30.000 KBps in quelli con più macchine. I tempi di questo test sono aumentati notevolmente perché si passa da una media di una decina di minuti visti nei test con pagina da 1 KB, a una media di qualche ora. Nonostante l'aumento dei tempi, giustificato anche dal dover fare passare attraverso la rete un grosso quantitativo di dati (nell'ordine dei 160 GB per il test a 32 macchine virtuali), fino a 32 macchine virtuali, il sistema riesce a soddisfare le richieste dei client. L'andamento rispecchia quello della linea ideale con scostamenti che sono un pò più accentuati rispetto al caso precedente, ma giustificati dai lunghi tempi d'esecuzione del test.
Il collo di bottiglia dei dischi ha fatto scendere l'utilizzo della CPU a livelli molto bassi, causa poco lavoro da svolgere, soprattutto nei casi con 8 e 16 macchine. Per i test con 24 e 32 macchine con più core assegnati, l'utilizzo della CPU sale invece di molto. Il motivo è da ricercare non solo nell'algoritmo di *scheduling*, come visto nel test con pagine da 1 KB, ma anche nella saturazione della RAM, che raggiunge livelli di consumo elevatissimi e costringe il sistema ad effettuare *swap* su disco. La soglia di rifiuto delle richieste dei client questa volta è stata raggiunta in anticipo con 40 macchine virtuali sia per 1, 4 o 8 core.

5.3 Caso numero 3: Sysbench

Sysbench genera una serie di query da effettuare su un database, che per la maggior parte generano scritture su disco. L'accesso in scrittura al disco in maniera casuale, unito alle transazioni, rende nuovamente il disco un collo di bottiglia, ma in questo caso la situazione è più evidente rispetto al caso precedente, soprattutto se consideriamo che Sysbench muove dati da 1 KB a 16 KB di dimensioni (questo è confermato anche dal basso utilizzo della rete in confronto a quello che viene fatto nel test con Apache e pagine da 500 KB).
Al crescere delle macchine virtuali il tempo di completamento dei test au-

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	6.200	47.000	4.000	2 : 30
16	6.250	44.500	3.750	7 : 30
24	5.600	35.000	3.100	12 : 30
32	3.000	20.000	2.000	32 : 30
40	2.800	17.000	1.700	48 : 00
48	2.500	14.000	1.300	82 : 00
56	1.600	6.200	600	157 : 00
64	1.500	5.000	400	260 : 00

Tabella 5.7: Valori medi per il test Sysbench con 1 core

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	5.700	40.000	4.000	3 : 45
16	4.500	30.000	3.000	8 : 45
24	4.300	20.000	2.000	18 : 00
32	5.000	20.000	2.000	32 : 00
40	6.500	10.000	2.000	60 : 00
48	X	X	X	X
56	X	X	X	X
64	X	X	X	X

Tabella 5.8: Valori medi per il test Sysbench con 4 core

MV	CPU (MHz)	Disco (KBps)	Rete (KBps)	Tempo (min)
8	6.000	40.000	3.500	6 : 00
16	7.000	40.000	3.500	12 : 30
24	5.000	20.000	2.500	22 : 30
32	6.250	15.000	1.500	33 : 30

Tabella 5.9: Valori medi per il test Sysbench con 8 core

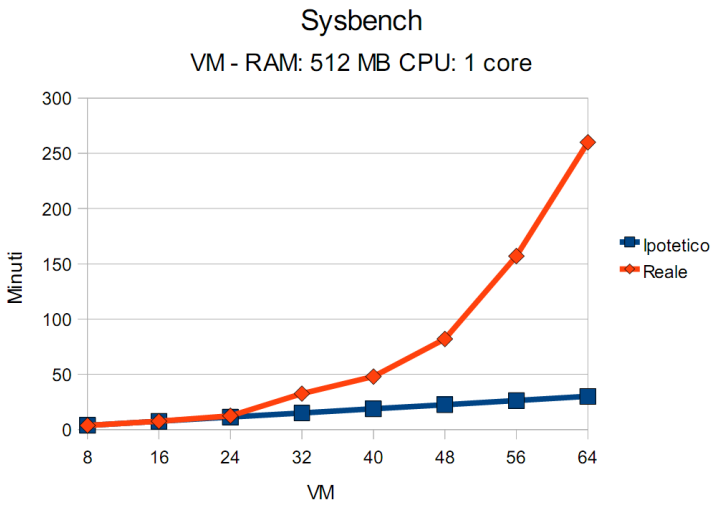

Figura 5.12: Grafico dei tempi ottenuti con i test Sysbench - 1 core

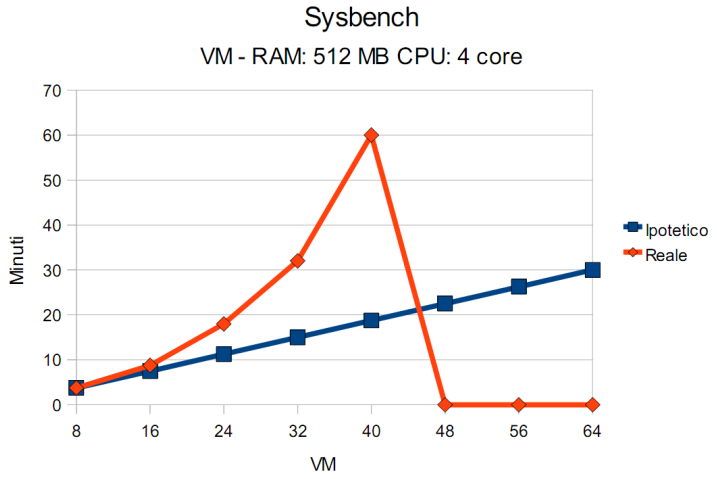

Figura 5.13: Grafico dei tempi ottenuti con i test Sysbench - 4 core

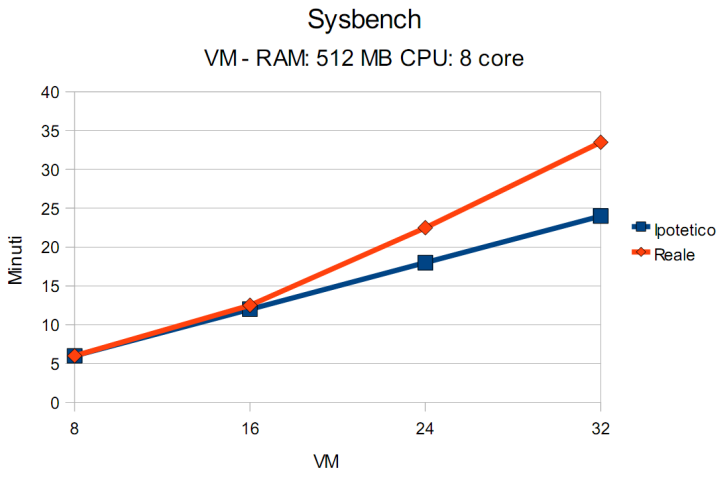

Figura 5.14: Grafico dei tempi ottenuti con i test Sysbench - 8 core

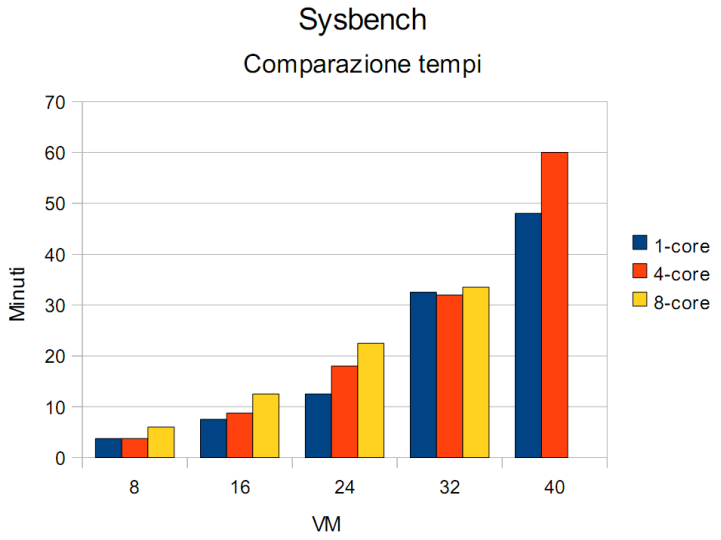

Figura 5.15: Grafico della comparazione dei tempi ottenuti con i test Sysbench a 1, 4 e 8 core

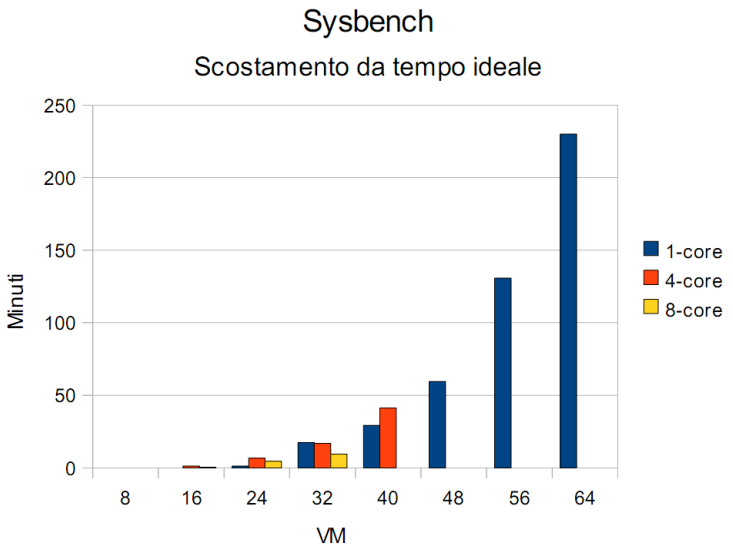

Figura 5.16: Grafico degli scostamenti dal tempo ideale (lineare) ottenuti con i test Sysbench a 1, 4 e 8 core

menta drasticamente in maniera **esponenziale**. Anche la CPU è messa sotto sforzo. Da notare questo comportamento: fino a che si utilizza 1 core l'utilizzo cala gradulamente, seguendo l'andamento calante delle prestazioni del disco. Con 4 e 8 core, invece, l'utilizzo della CPU rimane sullo stesso livello variando di poco verso l'alto o verso il basso. Questo si spiega poiché la riduzione di CPU, che dovrebbe avvenire, viene compensata dal consumo dovuto allo *scheduler* di ESXi, che nei test precedenti per 4 e 8 core, ha sempre portato la CPU a crescere con l'aumentare delle macchine impiegate. Con 8 e 16 macchine, infatti, risultano molto buoni sia per i casi a 1, 4 e 8 core, seguendo l'andamento della linea ideale. Da notare che, nonostante con 4 e 8 core seguano la linea ideale, già cominciano a metterci più tempo a completare rispetto ad 1 core soltanto. Il fatto di metterci più tempo rispetto a 1 core è risultato vero per tutta la durata dei test.
Passando a 24 e 32 macchine i test iniziano a discostarsi dalla linea ideale e comincia un degrado delle prestazioni. Questo degrado, come si può ben vedere dalla tabella, è dovuto principalmente al disco che scende in tutti i

test dai 40.000 KBps ottenuti con 8 e 16 macchine, a 20.000 KBps ottenuti con 24 e 32. I 4 dischi, infatti, passano dall'essere acceduti da 2/4 client contemporaneamente dei test con 8 e 16 macchine virtuali a 8/16 client dei test con 24 e 32. Appena si arriva a 40 macchine virtuali il disco scende attorno a 10.000 KBps per il test con 4 core, e si ha un ulteriore degrado delle prestazioni, che arrivati a 48 macchine, congestionerà completamente il server impedendogli di finire il test. Con solo 1 core per macchina virtuale invece si riesce a terminare i test fino a 64 macchine, anche se con andamento dei tempi esponenziale e raggiungendo un misero 5.000 KBps di trasferimento dati dal disco.

5.4 Analisi dei risultati

Analizzando i vari risultati, ottenuti all'interno di questo capitolo, possiamo affermare che per quanto riguarda la **CPU** e l'assegnazione dei core alle varie macchine virtuali, di un sistema con grossi livelli di carico, che raggiunge spesso livelli di congestionamento, la scelta migliore è quella di impostare 1 solo core. Uno spunto di ricerca potrebbe essere quello di eseguire i test, impostando diversi gruppi di macchine virtuali in configurazioni miste (anche dispari) di core, per capire quanto incide sulle prestazioni la frammentazione dei core; per esempio una macchina virtuale con 7 core virtuali e un altra con 2, su una macchina server con 8 core fisici totali, visto che in questa configurazione i core delle due macchine virtuali non potranno mai eseguire concorrentemente.
L'utilizzo dell'**Hyper-Threading** è consigliato a tutti coloro che possiedono un sistema, che non raggiunge mai la saturazione delle risorse disponibili e dove ci sono programmi che sfruttano il multithreading. Per evitare la saturazione una eventuale soluzione è quella di utilizzare insiemi di risorse, per limitare e distribuire equamente le risorse disponibili; questo per evitare che

il sistema superi normalmente il 70%/80% delle risorse globali, così da averne a disposizione un 30%/20% per coprire eventuali picchi di carico.
Se un sistema invece arriva al congestionamento l'Hyper-Threading abbassa le prestazioni del sistema. Per prima cosa ciò comporta un grosso consumo di RAM che già di per sé riduce le prestazioni; seconda cosa lo *scheduler* di ESXi deve trovare ogni volta dei core allineati da assegnare alle macchine virtuali, seguendo quello che viene chiamato Relaxed Coscheduling Algorithm; e terza cosa, se un sistema è saturo, i processori logici appartenenti ad uno stesso *package* competono fortemente sulla cache L2 e L3 e su tutta una serie di componenti funzionali (come l'ALU) non condivise, rallentandosi a vicenda.
Il principale collo di bottiglia riscontrato rimane comunque il **disco**. La situazione peggiora ancora, in caso che l'attività a cui il disco deve supplire, è quella di un motore di database. Il consiglio è infatti di evitare di virtualizzare motori di database, a meno di un attenta pianificazione delle risorse e comunque accettando tempi, che sarebbero inferiori a quelli di uno stessa situazione non virtualizzata. Il fattore disco spiega come mai nelle grandi aziende vengano utilizzate così tante risorse per pianificare una buona rete SAN in Fibre Channel. Quando si parla dei dischi, sarebbe meglio utilizzare il top di quanto la tecnologia dispone in quel momento, se il budget dell'azienda lo permette. In questa ottica sarebbe molto interessante rielaborare i test sostituendo i dischi SATA con dei dischi SSD, per esaminare se apportano un miglioramento alle prestazioni come ci si aspetta, e se si in quale misura questi incidano.
Per concludere, possiamo dire, che mentre con 8 o 16 macchine virtuali il sistema si comporta egregiamente in tutte le situazioni, da 24 a 32 inizia un leggero declino delle prestazioni soprattutto nel test relativo al database. Il punto di criticità medio del sistema viene raggiunto quando si arriva a **40** macchine virtuali, che è il punto dove il grafico esponenziale dei tempi comincia ad impennarsi vertiginosamente verso l'alto.

Capitolo 6

Conclusioni e sviluppi futuri

Dopo tutto il lavoro svolto all'interno di questa tesi posso affermare con certezza di avere una maggiore consapevolezza di quando avevo iniziato riguardo all'argomento virtualizzazione. Mentre da una parte i test hanno confermato comportamenti che mi aspettavo come l'inefficienza nel virtualizzare motori di database o il fatto che i dischi risultano essere uno dei maggiori colli di bottiglia, dall'altro sono rimasto sorpreso nello scoprire che sotto carichi molto grossi abilitare l'Hyper-Threading risulta in un maggior degrado delle prestazioni senza nessun beneficio aggiuntivo.
Ho poi scoperto quanto sia importante non consumare inavvertitamente tutte le risorse critiche del sistema prestando attenzione invece nel limitare le risorse sul 70%/80% della componente più critica; data, infatti, la natura software di molte delle componenti virtualizzate come il vSwitch, i processori virtuali, la memoria, è possibile effettuare un minuzioso lavoro di configurazione per limitarle a piacimento utilizzando insiemi di risorse. Mentre una corretta configurazione porta l'intero sistema virtualizzato verso la strada del successo, una cattiva configurazione risulterà fallimentare. È fondamentale quindi per una corretta configurazione avere a disposizione la massima disponibilità di dati da consultare, che ci permettano di scegliere la strada più appropriata alle nostre esigenze; questi dati possono essere ottenuti solamente in maniera sperimentale attraverso un accurata fase di testing come

ci ha dimostrato la tesi.

In quest'ottica sarebbe interessante proseguire i test ed ampliare il bacino dei risultati ottenuti prendendo come spunto alcune configurazioni che erano state prese in considerazione ad inizio tesi e di cui ho parlato nell'introduzione e nelle appendici. Confrontando questi nuovi risultati con quelli di questa tesi sarà poi possibile ampliare il discorso e ottenere ulteriori spunti per lavori futuri.

Tutto il discorso affrontato in questa tesi, riguardante la virtualizzazione, non è comunque un discorso isolato e a sé stante. Per comprenderlo maggiormente è necessario legarlo anche ad un altro aspetto molto importante, che è quello del Cloud Computing. Il Cloud Computing in generale è un modello di sfruttamento delle risorse offerte dalle reti di computer, che negli ultimi anni sta cominciando a prendere forma e che sta rivoluzionando il mercato. Il Cloud Computing nasce dal bisogno di grosse aziende come Microsoft, Google, Amazon, Yahoo di trarre profitto dall'inefficienza delle loro enormi reti, quando queste sono in uno stato in cui possiedono tanta potenza di calcolo inutilizzata.

Amazon, per esempio, fino a pochi anni fa, si ritrovava nella situazione di poter sfruttare solamente una piccola parte della potenza di calcolo delle sue macchine, salvo periodici picchi di carico di lavoro, che comunque andavano coperti, causa eventuali disservizi che aziende di questo calibro non possono permettersi. Mentre da una parte si è lavorato attraverso la virtualizzazione per consolidare tra loro l'enorme quantitativo di macchine a disposizione, dall'altro si è pensato di poter vendere tutta la potenza di calcolo in eccesso a privati che la necessitano, facendogliela pagare a seconda dell'utilizzo; un pò come avviene normalmente tra un azienda che fornisce luce, gas e acqua e i fruitori di questi servizi. All'utilizzatore finale poco importa quali macchine all'interno della "nuvola" gli stiano fornendo la potenza necessaria mancante, l'importante è che questa esigenza venga soddisfatta. Si è poi preso spunto, partendo da questo, per fornire nuovi servizi di varia tipologia: Software (SaaS - Software as a Service), Piattaforma (PaaS - Platform as a Service),

Infrastruttura (IaaS - Infrastructure as a Service). Tra questi servizi citiamo GoogleMail (SaaS) e Amazon Elastic Compute Cloud (IaaS).
Per dare un idea della reale potenza di questi servizi, citiamo un famoso caso[1] d'utilizzo, che risale all'ottobre del 2007, quando il Times necessitava di trasformare 4 TB di immagini TIFF di vecchi articoli in PDF. Per portare a compimento questo lavoro sarebbero state necessarie tante settimane di lavoro e una grossa potenza di calcolo. Il Times decise di affidarsi ad Amazon e ai suoi servizi di Cloud Computing: Amazon Elastic Computing Cloud 2 (EC2) e Amazon Simple Storage Service 3 (S3). Sfruttando 6 TB di spazio totale su S3 e ben 100 istanze di macchine virtuali su EC2, è stato possibile generare ben 11 milioni di PDF in poco più di 12 ore, risparmiando notevolmente sia in denaro che in tempo.
Prendendo spunto da un caso così ecclatante e di successo è naturale pensare alle tante possibilità che ancora questo nuovo campo ci può fornire. Soltanto il tempo potrà dirci quanto ancora crescerà l'utilizzo di servizi basati su Cloud Computing e quanto tutto ciò verrà ad influire anche sul mercato *home*, che per ora si ritrova a sfruttarli solo in minima parte.

[1] http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/

Appendice A

iSCSI e Linux

In questa appendice si mostrerà una configurazione basata su iSCSI attraverso l'utilizzo di Linux. Inizialmente si era pensato di utilizzare iSCSI per fornire lo spazio di memorizzazione attraverso la rete, ma poi, come spiegato all'interno della tesi, non è stato possibile un suo utilizzo per mancanza di PC. Si è deciso di lasciare questa parte sia per chi volesse riprendere iSCSI per test futuri, sia perché facente parte del lavoro pratico svolto durante la tesi. Questa parte è prettamente tecnica e scritta utilizzando il gergo di iSCSI (inititator, target, ecc...).

iSCSI target
Inizialmente creiamo dei target che possano essere acceduti poi dagli initiator. Per fare ciò utilizziamo Linux ed un suo pacchetto `iscsitarget`:

```
apt-get install iscsitarget
```

apriamo */etc/default/iscsitarget* e abilitiamo il suo utilizzo mettendo la variabile a true.

```
ISCSITARGET_ENABLE=true
```

A questo punto possiamo andare a modificare il file di configurazione */etc/ietd.conf*. Un esempio dei parametri più importanti da definire sono:

```
IncomingUser nome password
```

```
OutgoingUser altronome altrapassword

Target iqn.<yyyy-mm>.<reverse-domain>:<name>
   IncomingUser nome password
   OutgoingUser altronome altrapassword
   Lun 0 Path=/dev/sdc, Type=fileio
   Lun 1 Sectors=10000, Type=nullio
   Alias Test
   HeaderDigest   None
   DataDigest     None
   MaxConnections 1
```

Il file di configurazione di iSCSI permette di definire target multipli, ognuno dei quali può contenere multiple LUN. Il nome del target deve essere un nome univoco a livello globale un iqn. Il demone `ietd` avvia i target nell'ordine definito. Da notare inoltre che in iSCSI è possibile generare un sistema di autenticazione CHAP (Challenge Handshake Authentication Protocol) della tipologia challenge and response. Per avviare il demone digitiamo:

```
/etc/init.d/iscsitarget start
```

Analizziamo le proprietà:

[IncomingUser <username> <password>] (fuori dal target)
Lo <username> e la <password> utilizzati durante la fase di ricerca (*discovery*) per autenticare gli iSCSI initiator. Se non si viene autenticati non è possibile effettuare la ricerca. É possibile anche specificare più IncomingUser ma se nemmeno uno viene specificato, tutti possono effetuare la ricerca. Lo standard richiede che la <password> sia lunga almeno 12 caratteri.

[OutgoingUser <username> <password>] (fuori dal target)
Lo <username> e la <password> utilizzati durante la fase di ricerca per autenticare l'iSCSI server target agli initiator. Deve essere specificata solamente una volta. Lo standard richiede che la <password> sia lunga almeno 12 caratteri.

Target iqn.<yyyy-mm>.<tld.domain.some.host>[:<identifier>]
Definizione di un target con relativo nome iqn. La specifica EUI-64 non è supportata.

[IncomingUser <username> <password>]
Lo <username> e la <password> utilizzati per autenticare gli initiator iSCSI a questo target. Le informazioni specificate possono essere diverse da quelle globali viste sopra. Quelle sopra specificano se un initiator può effettuare la ricerca mentre quelle sotto specificano se un initiator può connettersi a questo target. È possibile anche specificare più IncomingUser ma se nemmeno uno viene specificato tutti possono connettersi a questo target.

[OutgoingUser <username> <password>]
Lo <username> e la <password> utilizzati per autenticare questo iSCSI target agli initiator. Le informazioni specificate possono essere diverse da quelle globali viste sopra. Lun <lun> Path=<device>, Type=(fileio|blockio) [,ScsiId=<scsi_id>] [,ScsiSN=<scsi_sn>] [,IOMode=(wb|ro)|| Sectors=<size>, Type=nullio Almeno una linea di questo tipo deve essere definita dentro un target. Il valore di <lun> può variare tra 0 e 2î14-1. Ad una LUN può essere assegnato una periferica (<*device*>) che può essere un qualsiasi periferica a blocchi (/dev/sda), partizione (/dev/sda1), logica (LVM) o raid (/dev/md0). Le modalità di accesso alla periferica possono essere di tipo fileio, blockio o nullio e verranno spiegate in seguito.

[Alias <aliasname>]
Semplicemente assegna un alias <aliasname> al target.

[HeaderDigest <CRC32C|None>] (Opzionale)
Se settato a CRC32C gli header dei dati trasferiti saranno protetti da un checksum CRC32C. Il valore di default è “None”.

[DataDigest <CRC32C|None>] (Opzionale)
Se settato a CRC32C i dati trasferiti saranno protetti da un checksum CRC32C.
Il valore di default è "None". **[MaxConnections <value>]** (Opzionale)
Deve essere settato a "1" che è anche il valore di default.

Modalità di accesso fileio, blockio o nullio.

nullio : Definisce una mappatura tra una LUN e un device virtuale (simile a /dev/null). Questà modalità è utile solo per effettuare misurazioni di prestazioni. Tutte le scritture vengono scartate e tutte le letture ritornano dati casuali.

blockio : Questa modalità permette di effettuare block I/O con il device evitando la cache di pagina per tutte le operazioni (cos'è la cache di pagina lo vediamo sotto). Questo è particolarmente utile nei casi in cui non vogliamo utilizzare la RAM di sistema, ma facciamo affidamento su un RAID hardware. È la scelta consigliata anche nel caso di utilizzo da parte di ambienti di virtualizzazione perché si ha una maggiore efficienza nella gestione di trasferimenti di dati non allineati.

fileio : questa modalità permette di sfruttare la cache di pagina per tutte le operazioni che vengono effettuate e va a riempire la memoria del sistema. Può essere utilizzato in due modi chiamati *IOMode*.

IOMode ro : Sola lettura (ro: Read Only).

IOMode wb : Scrittura (wb: Write Back). Questa modalità abilita la cache. Attenzione assicurarsi di avere un gruppo di continuità, oppure in caso di mancanza di corrente si potrebbe avere una perdita di alcuni dati, su cui si stava lavorando utilizzando questa metodologia (i dati scritti nella cache non ancora salvati).

Buffer Cache
Quando si legge da un filesystem vengono generate molte richieste di lettura

e scrittura di blocchi di dati. Tutte le richieste di scrittura e di lettura vengono fornite ai driver sotto forma di strutture *buffer_head*. Queste strutture forniscono tutte le informazioni di cui il driver ha bisogno. L'identificativo della periferica e il numero del blocco da cui leggere. Tutti i device a blocchi infatti vengono visti come un insieme lineare di blocchi delle stesse dimensioni.
Per velocizzare l'accesso fisico ai blocchi, Linux mantiene una cache dei blocchi. Questa cache è condivisa tra tutte le periferiche fisiche a blocchi. Se dati validi sono disponibili questo permette al sistema di guadagnare del tempo evitando di andare a leggere dalla periferica. Questi dati validi possono nel tempo essere scartati dalla cache oppure se acceduti frequentemente rimanerci ancora per molto. Le dimensioni di blocchi supportati sono da 512, 1024, 2048, 4096, 8192 byte.
Quando i blocchi sono nella buffer cache vengono anche inseriti in una lista chiamata Least Recently Used (LRU). Esiste una lista LRU diversa per ogni tipologia di blocco. Le tipologie di blocco conosciute sono:

clean : Nuovi buffer inutilizzati.

locked : Buffer che sono bloccati in attesa di essere scritti.

shared : Buffer condivisi.

dirty : Buffer che contengono nuovi dati validi che devono ancora essere scritti su disco.

unshared : Buffer che sono stati condivisi, ma che ora non lo sono più.

iSCSI initiator

VMWare include di già in ESXi la parte client che fa da initiator per collegarsi a target iSCSI. Nel caso si volesse utilizzare un initiator software è possibile utilizzare Linux ed il pacchetto `open-iscsi`:

```
apt-get install open-iscsi
```

Apriamo il file */etc/iscsi/iscsid.conf* e settiamo l'avvio del nodo in maniera automatica:

```
node.startup = automatic
```

Riavviamo il demone dell'initiator:

```
/etc/init.d/open-iscsi restart
```

Ora proviamo a connetterci al target per una fase di discovery e per scoprire quali target sono disponibili:

```
iscsiadm -m discovery -t st -p 192.168.0.101
iscsiadm -m node
```

A questo punto le informazioni sui target disponibili vengono salvati nei file all'interno della cartella */etc/iscsi/nodes*. Nel nostro caso il file di configuraizone si trova all'interno del file:

```
/etc/iscsi/nodes/iqn.2001-04.com.example:
          lun1/192.168.0.101,3260,1/default
```

È possibile modificare le informazione per collegarsi a quella LUN manualmente, editando il file oppure attraverso l'utilizzo del comando `iscsiadm`. Per esempio se volessimo abilitare l'autenticazione CHAP e definire utente più password con cui collegarci:

```
iscsiadm -m node --targetname "iqn.2001-04.com.example:lun1"
   --portal "192.168.0.101:3260" --op=update
   --name node.session.auth.authmethod --value=CHAP

iscsiadm -m node --targetname "iqn.2001-04.com.example:lun1"
   --portal "192.168.0.101:3260" --op=update
   --name node.session.auth.username --value=someuser

iscsiadm -m node --targetname "iqn.2001-04.com.example:lun1"
   --portal "192.168.0.101:3260" --op=update
   --name node.session.auth.password --value=secret
```

Per effettuare la connessione e la disconnessione:

```
iscsiadm -m node --targetname "iqn.2001-04.com.example:lun1"
   --portal "192.168.0.101:3260" --login
iscsiadm -m node --targetname "iqn.2001-04.com.example:lun1"
   --portal "192.168.0.101:3260" --logout
```

Una volta effettuato l'accesso possiamo trovare la nostra periferica attraverso il comando `fdisk -l` come fosse locale. A questo punto non ci resta che scrivere la tabella della partizioni da usare, partizionarlo con il filesytem scelto e montarlo in locale per un suo utilizzo.

Appendice B

Link Aggregation

In questa appendice verranno ripresi i concetti del Link Aggregation e delle sue modalità di funzionamento. Anche in questo caso Link Aggregation è una configurazione che è stata provata durante la fase pratica della tesi e può essere ripresa (magari insieme ad iSCSI) per provare una configurazione di test abbastanza complessa, previa la disponibilità di un quantitativo sufficiente di PC.
Anche in questo caso l'appendice, come quello precedente, è prettamente pratica e scritta utilizzando termini facenti parte del gergo utilizzato in ambito di configurazioni Link Aggregation.

Metodologie di aggregazione
Riprendiamo i concetti sul Link Aggregation rianalizzando brevemente le varie metodologie di aggregazione esistenti:

mode=0 : Politica Round Robin

mode=1 : Politica Active-backup

mode=2 : Politica XOR

mode=3 : Politica Broadcast

mode=4 : IEEE 802.3ad

mode=5 : Bilanciamento di carico adattivo

Ubuntu e bonding

Supponiamo di volere aggregare tra loro due interfacce di rete `eth0` e `eth1`. Utilizziamo il comando `mii-tool` per ottenere più informazioni sulle interfacce di rete.
Nota: mii sta per media-independent interface.
Il risultato del comando sarà:

```
eth0: negotiated 1000baseT-HD flow-control, link ok
eth1: negotiated 1000baseT-HD flow-control, link ok
```

Questo ci indica che le schede sono correttamente collegate allo switch (link ok) e che funzionano entrambe alla velocità di 1000 Mbit/s (ottenuto attraverso la negoziazione) e che permettono il controllo di flusso del traffico (flow control). Successivamente controlliamo che sia installato il comando `ethtool`, che ci fornisce una visione ancora più dettagliata delle informazioni di ogni scheda di rete (`ethtool` può essere utilizzato non solo per vedere le informazioni sulle schede di rete, ma anche per forzare una determinata velocità o modalità di funzionamento):

```
ethtool eth0 && ethtool eth1


Settings for eth0:
Supported ports: [ TP MII ]
Supported link modes: 10baseT/Half 10baseT/Full
   100baseT/Half 100baseT/Full
   1000baseT/Half 1000baseT/Full
Supports auto-negotiation: Yes
Advertised link modes: 10baseT/Half 10baseT/Full
   100baseT/Half 100baseT/Full
   1000baseT/Half 1000baseT/Full
Advertised auto-negotiation: Yes
Speed: 1000Mb/s
Duplex: Full
Port: MII
```

```
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
Supports Wake-on: pumbg
Wake-on: g
Current message level: 0x00000033 (51)
Link detected: yes

Settings for eth1:
Supported ports: [ TP MII ]
Supported link modes: 10baseT/Half 10baseT/Full
   100baseT/Half 100baseT/Full
   1000baseT/Half 1000baseT/Full
Supports auto-negotiation: Yes
Advertised link modes: 10baseT/Half 10baseT/Full
   100baseT/Half 100baseT/Full
   1000baseT/Half 1000baseT/Full
Advertised auto-negotiation: Yes
Speed: 1000Mb/s
Duplex: Full
Port: MII
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
Supports Wake-on: pumbg
Wake-on: g
Current message level: 0x00000033 (51)
Link detected: yes
```

Poi controlliamo che sia installato `ifenslave` un comando che serve per collegare e scollegare interfacce di rete di tipo *slave* ad un interfaccia virtuale (che viene chiamata comunemente interfaccia di *bonding*) di tipo *master*. Un interfaccia di rete *bonding* è vista dal kernel come una normale interfaccia di rete ma essendo virtuale spedirà il traffico attraverso le interfacce di tipo *slave*

utilizzando varie tecniche a seconda della modalità di trasmissione scelta. Questa tecnica permette una forma semplice di bilanciamento di carico.
In caso che uno dei pacchetti citati sopra non fossero presenti procedere all'installazione attraverso:

```
sudo apt-get install ifenslave ethtool mii-tool
```

Se stiamo utilizzando l'applicazione Network Manager per la gestione della rete, procediamo alla sua eliminazione dal momento che non ci permette di impostare funzionalità avanzate come Link Aggregation:

```
sudo apt-get remove network-manager network-manager-gnome
```

Apriamo il file */etc/network/interfaces* e modifichiamolo come segue:

```
auto lo
iface lo inet loopback


###Adapter bonding for eth0 and eth1
auto bond0
iface bond0 inet static
   address 192.168.0.30
   netmask 255.255.255.0
   gateway 192.168.0.1
   slaves eth0 eth1
   bond-mode 0
   bond-miimon 100
```

Possiamo quindi vedere dall'esempio qui sopra, che è stato scelto di creare un interfaccia virtuale `bond0` con indirizzo statico 192.168.0.30 e che ha come *slave* le schede `eth0` e `eth1` che verranno messe in *bonding* in modalità 0 (bond-mode); inoltre abbiamo scelto di effettuare un controllo ogni 100 millisecondi per vedere se qualcuna di queste schede ha smesso di funzionare correttamente.
Dopo aver fatto ciò assicuriamoci di caricare il modulo del *bonding*:

```
modprobe bonding
```

Per fare in modo che il modulo venga caricato ad ogni avvio del sistema, inseriamo una riga con scritto "bonding" nel file */etc/modules*. A questo punto non ci resta che riavviare la rete:

```
/etc/init.d/networking restart
```

Se tutto è andato a buon fine il comando `ifconfig` dovrebbe mostrare 4 interfacce di rete:

1. lo: localhost
2. bond0: in modalità *master*
3. eth0: in modalità *slave*
4. eth1: in modalità *slave*

```
bond0 Link encap:Ethernet  HWaddr 90:e6:ba:15:56:ef
      indirizzo inet:192.168.0.30
      Bcast:192.168.0.255  Maschera:255.255.255.0
      indirizzo inet6: fe80::92e6:baff:fe15:56ef/64 Scope:Link
      UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
      RX packets:2193 errors:0 dropped:0 overruns:0 frame:0
      TX packets:2278 errors:0 dropped:0 overruns:0 carrier:0
      collisioni:0 txqueuelen:0
      Byte RX:1661076 (1.6 MB)  Byte TX:319642 (319.6 KB)

eth0  Link encap:Ethernet  HWaddr 90:e6:ba:15:56:ef
      UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
      RX packets:33 errors:0 dropped:0 overruns:0 frame:0
      TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
      collisioni:0 txqueuelen:1000
      Byte RX:4092 (4.0 KB)  Byte TX:8081 (8.0 KB)
      Interrupt:36 Indirizzo base:0x6000

eth1  Link encap:Ethernet  HWaddr 90:e6:ba:15:56:ef
      UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
```

```
          RX packets:2160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2214 errors:0 dropped:0 overruns:0 carrier:0
          collisioni:0 txqueuelen:1000
          Byte RX:1656984 (1.6 MB)  Byte TX:311561 (311.5 KB)
          Interrupt:16 Indirizzo base:0x8000

lo        Link encap:Loopback locale
          indirizzo inet:127.0.0.1  Maschera:255.0.0.0
          indirizzo inet6: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisioni:0 txqueuelen:0
          Byte RX:158 (158.0 B)  Byte TX:158 (158.0 B)
```

Interessante notare come il MAC address risultante sia lo stesso per `bond0`, `eth0` e `eth1`.

Test

Proviamo ora ad eseguire:

```
cat /proc/net/bonding/bond0
```

Se tutto è andato a buon fine dovremmo ottenere una situazione simile a questa:

```
Ethernet Channel Bonding Driver: v3.5.0 (November 4, 2008)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 200
Down Delay (ms): 200

Slave Interface: eth0
MII Status: up
```

```
Link Failure Count: 0
Permanent HW addr: 90:e6:ba:15:56:ef

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:25:86:e3:14:94
```

A questo punto utilizzando la metodologia descritta qui sopra, ma cambiando ogni volta la modalità di bonding, sono andato a vericare il corretto funzionamento di alcune di queste:

Test: bond 0
A questo punto in modalità di bonding 0 possiamo testare il funzionamento. Iniziamo uno scaricamento un pò lungo e proviamo a scollegare prima un cavo di rete, poi ricolleghiamolo e poi scolleghiamo l'altro. Se tutto va bene lo scaricamento non dovrebbe interrempersi, questo proprio per un fattore di tolleranza ai guasti. Per quanto riguarda la trasmissione (TX) e la ricezione (RX) dei pacchetti da parte di `eth0` e `eth1` possiamo notare come entrambe spediscano e ricevano lo stesso quantitativo di pacchetti.

Test: bond 1
Successivamente è stata provata la configurazione *active-backup* dove una scheda è attiva (`eth0`) mentre l'altra è in modalità passiva (`eth1`) pronta a prendere il suo posto in caso di guasto. Scollegando il cavo di `eth0` avremo che `eth1` si attiverà e continuerà il lavoro. *Test: bond 4*
In modalità 4 (la IEEE 802.3ad) invece cambia tutto. Questa è una configurazione dinamica e necessita come detto sopra del supporto da parte dello Switch. Qui lo switch utilizza il protocollo LACP per controllare lo stato delle due schede di rete. Per prima cosa è necessario abilitare il LAG (Link Aggregation Group) sullo switch: questo permetterà di creare i famosi gruppi di aggregazione automatici tra schede della stessa velocità e tipologia. Per controllare il tutto eseguiamo il comando:

```
cat /proc/net/bonding/bond0
```

```
Ethernet Channel Bonding Driver: v3.5.0 (November 4, 2008)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 200
Down Delay (ms): 200

802.3ad info
LACP rate: slow
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
   Aggregator ID: 5
   Number of ports: 2
   Actor Key: 17
   Partner Key: 49978
   Partner Mac Address: 00:22:3f:f1:a7:34

Slave Interface: eth0
MII Status: up
Link Failure Count: 0
Permanent HW addr: 90:e6:ba:15:56:ef
Aggregator ID: 5

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:25:86:e3:14:94
Aggregator ID: 5
```

La cosa importante è che entrambe le schede possiedano lo stesso Aggregator ID (in questo caso il 5) e che sotto la voce Active Aggregator Info il numero delle porte utilizzate sia 2. Ora possiamo effettuare una prova iniziando lo scaricamento

di un file. Ovviamente varie richieste TCP vengono bilanciate sulle due schede, ma il traffico che viene generato in trasmissione non è più simmetrico, come nella modalità 0, ma asimmetrico.

Appendice C

NFS e RAID Software

In questa appendice si tratterà invece la parte pratica riguardante la configurazione su Linux di NFS e di RAID software. Questa configurazione era stata presa in considerazione nella fase iniziale quando si era pensato di poter sfruttare una macchina con Linux per creare un architettura di memorizzazione NAS sfruttando NFS e utilizzando RAID software.
È necessario fare una precisazione riguardante il RAID. Fin dal principio era mia intenzione utilizzare uno spazio di memorizzazione che utilizzasse RAID (sia che fosse software o hardware) per avere un caso di studio il più simile ad una reale situazione aziendale. Mentre questo era possibile, su una macchina Linux che facesse da NAS, utilizzando RAID software, questa scelta si è dovuta abbandonare quando si è deciso di passare, per mancanza di PC, all'utilizzo di un unico server con spazio di memorizzazione locale. Infatti i dischi locali vengono gestiti in maniera automatica da ESXi e questo non permette nessuna forma di RAID software ma necessita di un controllor hardware che purtroppo deve essere certificato come compatibile e di cui non ero in possesso.
Alla fine ho deciso di non utilizzare RAID all'interno dei miei test della tesi, ma ho comunque lasciato questa parte di configurazione che era stata intrapresa per chi volesse riprendere i test su NAS.

NFS Lato server
Installiamo i servizi di NFS:

```
apt-get install portmap nfs-kernel-server
```

Per indicare al server quali saranno le cartelle che dovranno essere condivise modifichiamo il file */etc/exports* e per ogni cartella da condividere aggiungiamo una riga, seguendo la seguente sintassi:

```
/cartella_da_condividere ip1(opzioni) ip2(...)...


Esempio:
   /mnt/nfs 192.168.0.1/24(rw,sync)
```

Adesso possiamo rendere il sistema più sicuro specificando quali client possono accedere al PC e quali invece no. Specifichiamo i client, e relativi servizi lato server, a cui vogliamo dare l'accesso modificando il file */etc/hosts.allow* ed aggiungiamo la seguente riga (per specificare più indirizzi IP separarli con la virgola):

```
portmap mountd nfsd statd lockd rquotad : ip_dei_client_ammessi
```

Specifichiamo i client, e relativi servizi lato server, a cui vogliamo negare l'accesso modificando il file */etc/hosts.deny* ed aggiungiamo la seguente riga:

```
portmap mountd nfsd statd lockd rquotad : ALL
```

Con questa riga neghiamo a tutti l'accesso alle risorse condivise via NFS, tranne che ai client specificati nel file */etc/hosts.allow*. Per negare l'accesso ad un preciso client, al posto di ALL, basta specificarne l'indirizzo IP. Per negare l'accesso a tutti i client tranne ad uno in particolare, la nostra riga in */etc/hosts.deny* dovrà essere:

```
portmap mountd nfsd statd lockd rquotad : ALL EXCEPT ip
```

Nei file *hosts.allow* e hosts.deny potete specificare i client tramite indirizzo IP o tramite host e/o nome del dominio (nel caso abbiate un vostro server DNS). A questo punto, per rendere effettive le modifiche, dobbiamo riavviare i demoni:

```
/etc/init.d/portmap restart
/etc/init.d/nfs-kernel-server restart
```

NFS Lato client

Ora passiamo alla configurazione lato client. Installiamo, nel caso non fossero già presenti, i pacchetti necessari:

```
apt-get install portmap nfs-common
```

Il client è pronto per montare la condivisione, ma possiamo rendere anch'esso più sicuro specificando delle politiche di accesso, come precedentemente fatto per il server. Modifichiamo il file */etc/hosts.deny* ed inseriamo la seguente riga:

```
portmap : ALL
```

Adesso modifichiamo il file */etc/hosts.allow* ed inseriamo la seguente riga:

```
portmap : ip
```

Creiamo la cartella che sarà il nostro punto di mount:

```
mkdir /mnt/punto_di_mount
```

Montiamo la condivisione con il comando:

```
mount -t nfs ip_server:/../../cartella_condivisa
      /mnt/punto_di_mount
```

Volendo, possiamo montare la condivisione direttamente nel file */etc/fstab* per averla disponibile automaticamente all'avvio. **RAID Software**
Per quanto riguarda il RAID software su Linux è necessario installare il pacchetto `mdadm`.
Nota: `mdadm` sta per Multi Device Admin.

```
apt-get install mdadm
```

Creiamo 5 partizioni con `fdisk`. Mettiamo il caso di avere un disco di prova `/dev/sdb`.

```
fdisk /dev/sdb
```

A questo punto dobbiamo creare una partizione estesa sdb1 per tutto lo spazio che vogliamo utilizzare di sdb e poi andiamo a creare 5 nuove partizioni logiche all'interno di quella partizione estesa, stando bene accorti nel crearle tutte e 5 delle stesse dimensioni.
Queste saranno solitamente create a partire da sdb5 fino a sdb9. Utilizziamo ancora *fdisk* per cambiare l'etichetta del filesytem utilizzato (opzione `-t`) alle 5 partizioni assegnadogli quello con codice **"fd"** e cioè autorilevamento di RAID. A questo punto con l'opzione `w` andiamo a scrivere su disco la tabella delle partizioni.
Per vedere ciò che abbiamo fatto fino ad ora:

```
fdisk -l
```

Per verificare che il sistema abbia riconosciuto la nuova partizione digitiamo il comando:

```
partprobe
```

A questo punto utilizziamo il comando `mdadm` per creare un RAID5 con 4 dischi attivi e un disco di *spare*.

```
mdadm --create /dev/md0 --level=5 --spare-devices=1
      --raid-devices=4 /dev/sdb5 /dev/sdb6 /dev/sdb7
      /dev/sdb8 /dev/sdb9
```

Dove abbiamo che:

create : Indichiamo il device RAID da creare.

level : Indichiamo la tipologia di RAID che vogliamo utilizzare.

spare-devices : Il numero di dischi di *spare* utilizzati.

raid-devices : Il numero dei dischi che prendono parte attivamente al RAID.

Per vedere se tutto è andato a buon fine lanciamo:

```
cat /proc/mdstat
```

Se vogliamo verificare che il RAID è stato settato correttamente possiamo provare a mandare giù un disco tramite il comando:

```
mdadm /dev/md0 -f /dev/sdbX
```

E nuovamente verificare con:

```
cat /proc/mdstat
```

Se il RAID funziona correttamente, a questo punto possiamo creare dei Logical Volume. Installiamo il pacchetto `lvm2`:

```
apt-get install lvm2
```

Poi continuiamo prima con la creazione di una partizione logica sopra il nostro RAID:

```
pvcreate /dev/md0
```

Aggiungiamo un Logical Group:

```
vgcreate vg0 /dev/md0
```

Creiamo all'interno del logical group "vg0" un paio di Logical Volume:

```
lvcreate -L 30G -n lvLinux vg0
lvcreate -L 30G -n lvWindows vg0
```

A questo punto non ci resta che creare il filesystem:

```
mkfs.ext4 /dev/vg0/lvLinux
mkdosfs -F 32 /dev/vg0/lvWindows
```

Non resta che aggiungere le entry in */etc/fstab* e montarli.

Appendice D

Grafici

Ecco un esempio di alcuni grafici ottenuti darante la fase dei test. Per chi volesse consultare tutti i grafici, può richiedermeli scrivendo a `amici@cs.unibo.it`.

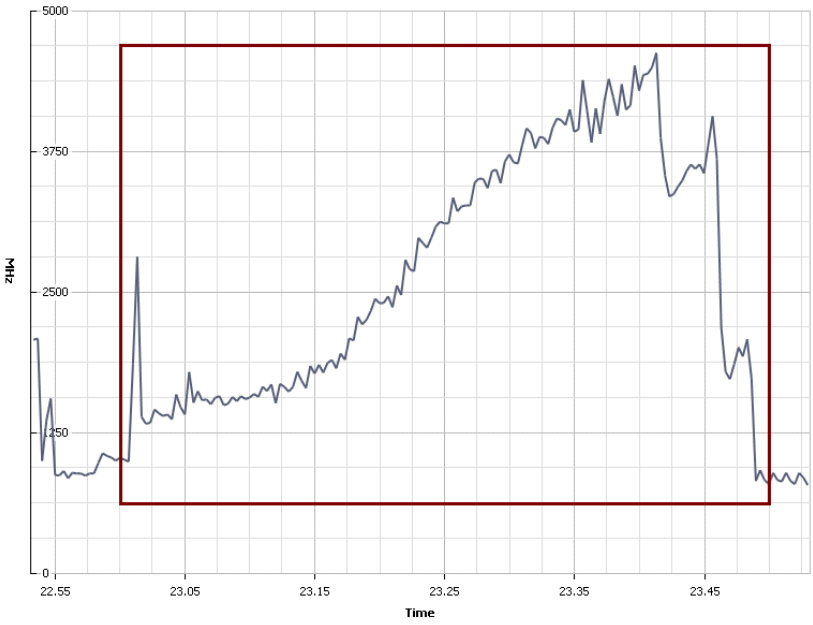

Figura D.1: sysbench-1core-40VM-CPU

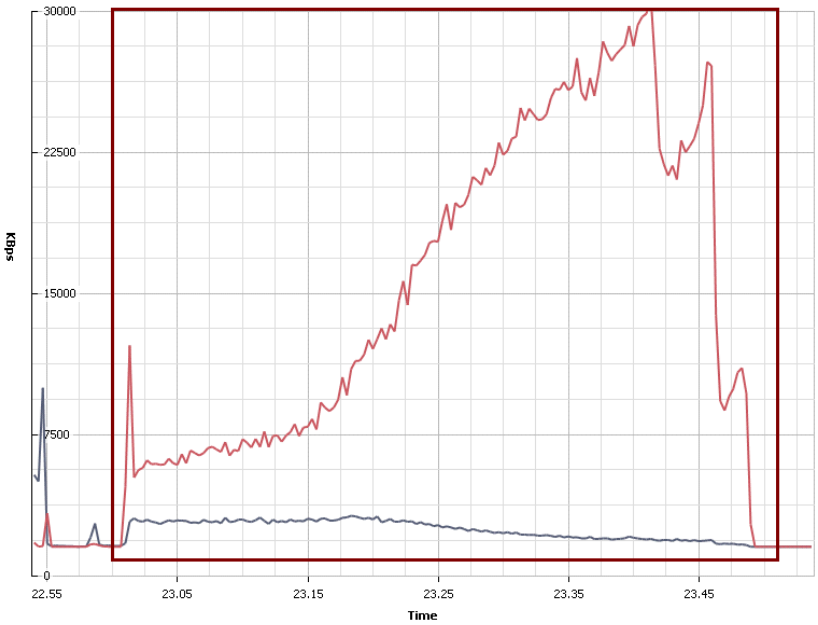

Figura D.2: sysbench-1core-40VM-disco

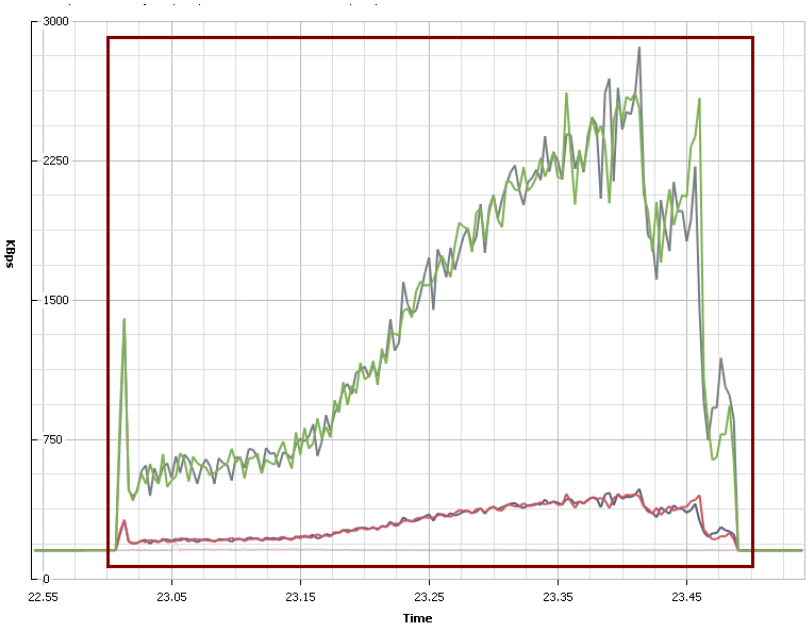

Figura D.3: sysbench-1core-40VM-rete

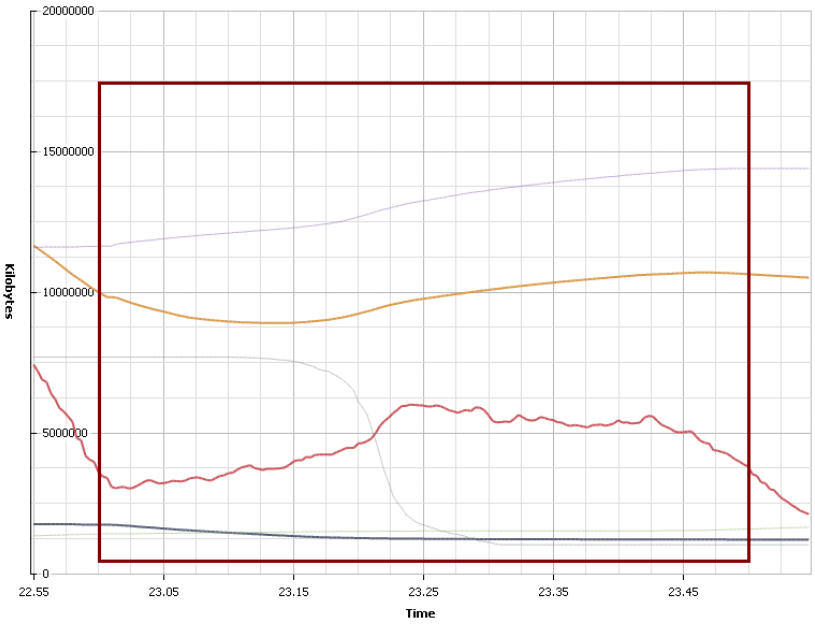

Figura D.4: sysbench-1core-40VM-RAM

Bibliografia

[1] [Ada06] Adaptect, "Measuring the True Performance of Today's SATA and SAS RAID Controllers", 2006, http://whitepapers.zdnet.com/abstract.aspx?docid=311004, 7 dicembre 2006

[2] [AdaAge06] Keith Adams, Ole Agesen, "A comparison of Software and Hardware Tecniques for x86 Virtualization", 2006, http://www.vmware.com/pdf/asplos235_adams.pdf, 8 agosto 2006

[3] [Cli09] Ken Cline, "The Great vSwitch Debate", 2009, http://kensvirtualreality.wordpress.com/2009/03/29/the-great-vswitch-debate-part-1/, 29 marzo 2009

[4] [DeM06] Ivan De Marino, "VLAN - Virtual Local Area Network",2006, http://downloads.detronizator.org/projects/tesina_vlan/html/tesina_vlan.html#x1-30000A.2.1, 4 febbraio 2006

[5] [Ent07] Enterasys, "Configuring VLANs", 2007, http://secure.enterasys.com/support/manuals/hardware/VLAN.pdf, 8 gennaio 2009

[6] [Fra07] Howard Frazier e al., "IEEE 802.3ad Link Aggregation (LAG)", http://www.ieee802.org/3/hssg/public/apr07/frazier_01_0407.pdf, 20 aprile 2007

[7] [How07] HowToForge Practical Guide, "How to Set up Network Bonding in Ubuntu 6.10", 2007, http://www.howtoforge.com/network_bonding_ubuntu_6.10, 13 febbraio 2007

[8] [HP03] HP, "Assessing and Comparing Serial Attached SCSI and Serial ATA Hard Disk Drives and SAS interface", 2003, ftp://ftp.compaq.com/pub/

products/servers/proliantstorage/drives-enclosures/sata-vs-sas.pdf, 21 ottobre 2003

[9] [LamGai07] Garry Lamasa, Silvano Gai, "Fibre Channel over Ethernet in the Data Center: an introduction", 2007, FibreChanneloverEthernetintheDataCenter:anintroduction, 18 marzo 2008

[10] [Low09] Scott Lowe, "Mastering VMWare vSphere 4", Wiley Publishing Inc., Indianapolis, Indiana, 2009

[11] [Low10] Scott Lowe, "Calculate IOPS in a storage array", 2010, http://www.zdnetasia.com/calculate-iops-in-a-storage-array-62061792.htm, 11 marzo 2010

[12] [LSPD10] Mark Lippitt, Erik Smith, Erik Paine, Mark Anthony De Castro, "Fibre Channel over Ethernet (FCOE)", 2010, http://www.emc.com/collateral/hardware/technical-documentation/h6290-fibre-channel-over-ethernet-techbook.pdf, 7 maggio 2010

[13] [Mah06] Mallik Mahalingam, "I/O Architectures for Virtualization", 2006, http://download3.vmware.com/vmworld/2006/tac0080.pdf, 7 novembre 2006

[14] [Nov06] Novalis, "VLAN, queste sconosciute", 2006, http://novalis.mib.infn.it/site/media/tech/vlan.html, 4 agosto 2006

[15] [Pre02] W. Curtis Preston, "Using SANs and NAS", O'Reilly, USA, 2002

[16] [Raj06] Krishna Raj Raja, "VI3 Networking Scenarios and Troubleshooting", 2006, http://download3.vmware.com/vmworld/2006/tac9689-b.pdf, 8 novembre 2006

[17] [TLM06] Jon Tate, Fabiano Lucchese, Richard Moore, "Introduction to Storage Area Networks", IBM Redbooks, Fourth Edition, USA, 2003

[18] [Tsa02] Mel Tsai, "An Introduction to iSCSI", 2002, http://www.mtsai.net/documents/tsai-iSCSI.pdf, 13 novembre 2002

[19] [Ubu09] Ubuntu Forum, "Link Aggregation", 2009, https://help.ubuntu.com/community/LinkAggregation, 29 novembre 2009

[20] [VMW03] VMWare, "NIC Teaming IEEE 802.3ad", 2003, http://www.vmware.com/pdf/esx2_NIC_Teaming.pdf, 11 agosto 2003

[21] [VMW04a] VMWare, "Hyper-Threading Support in VMWare ESX Server 2", 2004, http://www.vmware.com/pdf/esx21_hyperthreading.pdf, 8 aprile 2004

[22] [VMW04b] VMWare, "Best Practices Using VMware Virtual SMP", 2004, http://www.vmware.com/pdf/vsmp_best_practices.pdf, 29 giugno 2005

[23] [VMW06a] VMWare, "Virtualization Overview",2006, http://www.nitro.ca/images/pdf/virtualization.pdf, 19 maggio 2006

[24] [VMW06b] VMWare, "Multipathing", 2006, http://pubs.vmware.com/vi301/san_cfg/wwhelp/wwhimpl/common/html/wwhelp.htm?context=san_cfg&file=esx_san_cfg_manage.8.16.html, 2009

[25] [VMW07a] VMWare, "Understending Full Virtualization, Paravirtualization, and Hardware Assist", 2007, http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf, 18 ottobre 2007

[26] [VMW07b] VMWare, "VMWare Virtual Networking Concepts", 2007, http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf, 18 luglio 2007

[27] [VMW08] VMWare, "Building the Virtualized Enterprise with VMWare Infrastructure", 2008, http://www.vmware.com/pdf/vmware_infrastructure_wp.pdf, 23 aprile 2008

[28] [VMW09a] VMWare, "VMWare vSphere 4", 2009, http://www.vmware.com/files/it/pdf/vsphere_datasheet_it.pdf, 7 aprile 2009

[29] [VMW09b] VMWare, "VMWare Converter", 2009, http://www.vmware.com/files/pdf/09Q1_VM_CONVERTER_DS_EN.pdf, 14 febbraio 2009

[30] [VMW09c] VMWare, "VMWare ESX and ESXi", 2009, http://www.vmware.com/files/pdf/esx_datasheet.pdf, 20 ottobre 2009

[31] [VMW09d] VMWare, "VMWare vStorage VMFS", 2009, http://www.vmware.com/files/pdf/VMware-vStorage-VMFS-DS-EN.pdf, 16 ottobre 2009

[32] [VMW09e] VMWare, "VMWare vCenter Server 4", 2009, http://www.vmware.com/files/pdf/vcenter_server_datasheet.pdf, 16 ottobre 2009

[33] [VMW09f] VMWare, "VMWare VMotion", 2009, http://www.vmware.com/pdf/vmotion_datasheet.pdf, 16 ottobre 2009

[34] [VMW09g] VMWare, "VMWare Storage VMotion", 2009, http://www.vmware.com/files/pdf/storage_vmotion_datasheet.pdf, 16 ottobre 2009

[35] [VMW09h] VMWare, "VMWare Distributed Resource Scheduler (DRS)", 2009, http://www.vmware.com/files/pdf/drs_datasheet.pdf, 19 ottobre 2009

[36] [VMW09i] VMWare, "VMWare High Availability", 2009, http://www.vmware.com/files/pdf/ha_datasheet.pdf, 19 ottobre 2009

[37] [VMW09l] VMWare, "VMWare Fault Tolerance", 2009, http://www.vmware.com/files/pdf/fault_tolerance_datasheet.pdf, 19 ottobre 2009

[38] [VMW10] VMWare, "Fibre Channel SAN Configuration Guide", 2010, http://www.vmware.com/pdf/vsphere4/r40_u1/vsp_40_u1_san_cfg.pdf, 19 maggio 2010

Ringraziamenti

Vorrei iniziare i ringraziamenti partendo dalla mia ragazza Elisa per l'aiuto e l'amore che ha saputo darmi durante questo lungo percorso, anche nei momenti che credevo di non farcela ed alla sua famiglia, a Paolo, Dolores e Laura (per non parlare di Charlie), che mi ha nutrito nel corpo (che buona la pizza!) e nello spirito con tanti consigli.
Un ringraziamento fraterno ad Ale una delle persone più speciali che conosco.
Un ringraziamento alla società per cui lavoro, il CISCOOP, ed in particolare il gruppo 1 che ha dovuto sopportarmi e supportarmi per un anno intero: Daniele Forcellini (grazie per la pazienza), Fausto Mularoni, Elena Canova, Valeria Zanni e Simona Giulianelli.
Un grazie anche a tutti gli amici che mi sono stati vicini e sono sempre stati comprensivi: Denny, Nick, Borga, Lucz, Bollo, Ila, Elly, Bitz, Pelli, Davi, Marty, Vanny, Gle, Luca. Un grazie speciale a Brigz e alla nostre chiaccherate infinite e alla Cri che mi ha fatto conoscere lati di me che non credevo possibili: vi voglio bene.
Un altro ringraziamento speciale a Massimiliano Casali, Barbara Massari e alla piccola Giulia: grazie per avermi dedicato il vostro tempo, il regalo più speciale che una persona possa ricevere.
Un ringraziamento in generale a tutti quelli che conosco e che credono in me e che per questioni di memoria non sono stati citati: voi sapete chi siete. Un ringraziamento simbolico al mio PC che ha dovuto aspettare per più di un anno per poter vedere l'ombra di un videogioco: RAID 0 arrivo!
Infine un ringraziamento di cuore a mia sorella, ed in particolar modo a mio babbo e mia mamma, che in fondo non hanno mai smesso di crederci: spero di avervi resi orgogliosi.