UNIVERSITA' DEGLI STUDI DEL SANNIO

Facoltà di Ingegneria

Corso di laurea in

INGEGNERIA INFORMATICA

A.A. 2008/2009

Tesi di laurea

Analisi delle prestazioni di sistemi cluster in ambienti paravirtualizzati

RELATORE

Ch.mo Prof.

Umberto Villano

CANDIDATO

Flavio Pace

195001062

Ringraziamenti

Credevo che la stesura dei ringraziamenti fosse la parte più semplice della tesi, invece ho dovuto ricredermi perché le persone che mi sono state vicine in questi anni sono molte e da ognuna di loro ho appreso qualcosa.

In assoluto le singole persone che più di ogni altre hanno meritato che questo lavoro fosse loro dedicato sono certamente i miei genitori, per aver aspettato e sopportato così a lungo. Ad essi vanno tutta la mia stima, il mio rispetto e la mia riconoscenza, sperando che questa prima meta raggiunta gli renda orgogliosi di me.

A mio fratello, il cubatore della famiglia, il quale ha scelto una strada diversa rispetto alla mia e che non potrà essere presente alla discussione della tesi.

A Fabrizio ed ai miei nonni, anche se non potranno mai leggere queste righe mi guardano e mi proteggono dall'alto.

A Lei, perché ormai vessato dal suo sorriso, è diventata una parte di me.

Ai miei amici, grazie a loro ho appreso tutto ciò che non si trova su manuali o libri e che mi hanno supportato e "sopportato" in questi lunghi anni, sono davvero fortunato ad avervi!

Ai ragazzi del laboratorio di Ingegneria, grazie ai quali riuscivo a trovare sempre un consiglio o un sorriso, anche nei giorni più cupi.

Inoltre ringrazio il Prof. Umberto Villano per la pazienza e la massima disponibilità dimostrata e per avermi introdotto nell'affascinante mondo dei sistemi cluster, e il Dott. Antonio Cuomo, che come un fratello maggiore mi ha consigliato ed indirizzato.

Indice

Introduzione

Il futuro della tecnologia ha sempre le sue radici nel passato. Sebbene oggigiorno il termine virtualizzazione sembri essere un hot topic nell'ambito degli ambienti IT, in special modo nei sistemi mainframe, esso ha un passato davvero molto lungo. Una tecnologia che sta trasformando il settore IT di oggi, e probabilmente avrà un ruolo fondamentale nelle infrastrutture dei data center del domani.

Il presente lavoro di tesi si inserisce nell'ambito di una progetto di ricerca su un framework attuo alla valutazione delle performance dei cluster virtuali (1) finalizzato alla misurazione di alcune caratteristiche di rete come il *throughput* e *l'utilizzo della CPU* dei vari processi sender/receiver verso un certo numero di Macchine Virtuali (VM). Come tecnica di virtualizzazione è stata usata la Paravirtualizzazione, che come vedremo in seguito è quella che permette prestazioni molto simili a quelle native, realizzata usando l'hypervisor open-source Xen, all'interno della distribuzione Rocks.

La prima forma di virtualizzazione nacque negli anni '60 nei laboratori IBM Cambridge Scientific Center con lo sviluppo del Sistema Operativo CP-40 (2), il quale aveva come compito primario quello di ottimizzare l'uso del mainframe System/360 dividendolo a livello logico in un numero di macchine virtuali (quattordici) così da permettere il multitask e l'esecuzione parallela di più sistemi operativi.

Venne rapidamente rimpiazzato dal CP-67, la seconda versione dell'hypervisor IBM, il quale dava ad ogni user del mainframe un *Conversational Monitor System* (CMS) il quale era essenzialmente un Sistema Operativo single-user.

Nonostante il concetto della virtualizzazione negli anni '80 venne messo da parte a causa dello sviluppo dei mini-computer (minori capacità di calcolo), oggi assistiamo ad un suo risveglio grazie alle esigenze di sicurezza, isolamento e testing degli applicativi software.

Lo sviluppo di hardware sempre più performanti fa si che la virtualizzazione non venga usata solo a livello business (mainframe, data center, sistemi cloud) ma anche negli ambienti desktop dove far coesistere contemporaneamente più sistemi operativi spesso è una necessità.

In generale, la virtualizzazione permette di realizzare ambienti d'esecuzione personalizzabili e configurabili dinamicamente, grazie al concetto stesso di virtualizzazione che ci indica un disaccoppiamento tra l'ambiente di esecuzione e le risorse fisiche. Un ambiente di esecuzione siffatto viene chiamato "macchina virtuale". Più macchine virtuali possono coesistere contemporaneamente su una stessa risorsa, senza pregiudicare in alcun modo il loro isolamento, e combinandole tra di loro è possibile dar vita a più cluster virtuali.

Il concetto di cluster viene introdotto nel '67 da Gene Amdahl della IBM, il quale pubblicò un articolo in cui si discuteva dell'aumento delle prestazioni grazie all'esecuzione in parallelo di un'operazione. Solo negli anni '80, con l'adozione di reti ad alta velocità e la diffusione dei PC si ebbero i primi sistemi di calcolo ad elevate prestazioni formati dall'interconnessione di molte macchine a basso costo.

Tali sistemi, chiamati cluster, sono formati da hardware molto diffuso e relativamente economico che ha permesso la diffusione di tale approccio soprattutto negli ambienti HPC (*High Performance Computing*).

La complessità sempre crescente, e spesso la necessità di avere più ambienti di esecuzione indipendenti tra loro, hanno portato ad introdurre la virtualizzazione nei sistemi cluster. Le tecnologie di virtualizzazioni possono essere molto di aiuto in quanto permettono di avere più cluster virtuali configurati per una specifica applicazione sullo stesso cluster fisico.

Il tool da me sviluppato nel lavoro di tesi permette in automatico di calcolare il throughput massimo delle varie macchine virtuali in esecuzione, mettendolo in relazione con l'uso della CPU virtuale ed esegue il plot dei risultati. Infine viene effettuato il profiling per avere una panoramica sulle funzioni maggiormente interessate dai tool di benchmarking e che introducono l'overhead delle performance di rete.

La tesi è strutturata come segue. Nel capitolo 1 viene introdotto il concetto di virtualizzazione ed i vari approcci usati con uno sguardo più approfondito all'architettura IA-32 ed ai nuovi processori Intel e AMD. Il capitolo 2 riguarda interamente Xen e la sua architettura, spiegando nel dettaglio l'approccio usato da Xen per la virtualizzazione, la paravirtualizzazione. Nel capitolo 3 vi è una panoramica sull'ambiente di esecuzione dei test dove si spiega il concetto fondamentale di cluster, le varie tipologie ed una introduzione a Rocks , il sistema operativo utilizzato. Il capitolo 4 tratterà la metodologia adottata, una panoramica dei vari tool utilizzati per il benchmarking delle prestazioni, ed i risultati avuti. Nel capitolo 5 verrà utilizzato

Xenoprof, un profiling toolkit sviluppato appositamente per gli ambienti virtuali di Xen, per analizzare l'esecuzione di un tool di misurazione delle prestazioni di rete nei cluster virtualizzati, alla ricerca della causa a monte di un' anomalia riscontrata.

CAPITOLO 1

1. Virtualizzazione

Con il termine virtualizzazione si vuole indicare la logica suddivisione delle risorse fisiche di un computer in multipli ambienti di esecuzione. In termini pratici, la virtualizzazione fornisce la capacità di lanciare applicazioni, sistemi operativi, o servizi di sistema in un ambiente logico che può differire completamente dal livello fisico sottostante. Tale livello di astrazione ci dà la possibilità di far eseguire applicativi in sistemi fisici diversi da quelli per cui sono stati progettati superando il problema della portabilità.

La virtualizzazione può essere applicata a tre livelli:

- **Linguaggi ad alto livello**
- **Sistema Operativo**
- **Hardware.**

L'esempio più popolare dell' approccio alla virtualizzazione usato dai linguaggi di alto livello è quello del linguaggio di programmazione Java (3) in cui il codice sorgente viene prima compilato e trasformato in *byte-code* e poi lanciato in esecuzione sulle Java Virtual Machine le quali eseguono il codice e riportano output, come mostrato nella figura sottostante il codice prodotto è indipendente dal sistema operativo in cui viene eseguito.

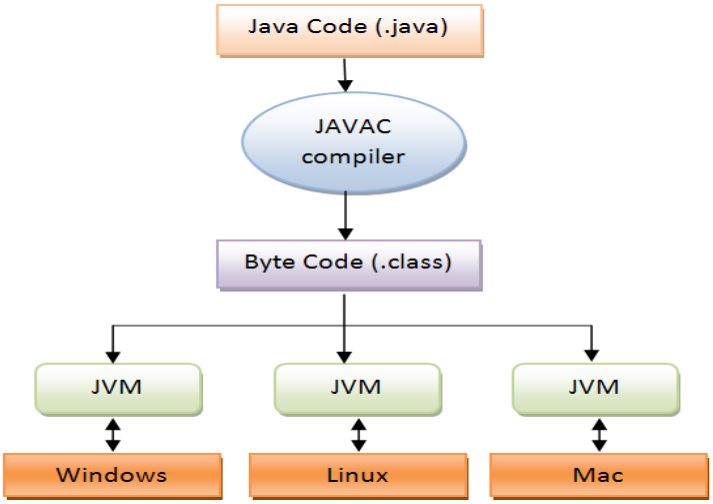


Figura 1.1 - Esempio Virtualizzazione in Java

I moderni sistemi operativi implementano già un semplificato sistema di virtualizzazione, come può essere l'illusione che hanno i processi in esecuzione di avere l'uso esclusivo della CPU oppure l'astrazione della memoria virtuale che da loro l'illusione di avere a disposizione tutta la memoria fisica.

La virtualizzazione a livello hardware introduce il concetto di macchina virtuale: l'hardware di un singolo sistema viene astratto a livello logico in più ambienti di esecuzione (macchine virtuali), creando l'illusione che ciascun sistema operativo, che gira in questo ambiente, disponga di un proprio sistema fisico.

Come definito da Popek and Goldberg, la macchina virtuale è " un efficiente, duplicato isolato della macchina fisica reale " (4) per tale motivo questa tecnica viene largamente utilizzata dai sistemi operativi per prevenire bug o l'esecuzione di codice malevolo.

Per capire meglio le tecniche di virtualizzazione negli ambienti x86 verrà fatto un piccolo background sulla terminologia dei componenti usati. L'hypervisor (in giallo nella figura sottostante) è responsabile dell'hosting e della gestione di tutte le *macchine virtuali* che a loro volta girano sulle VMM (virtual machine monitor).

L'h*ypervisor* è direttamente a contatto con l'hardware reale delle macchine, svolgendo funzionalità molto varie e dipendenti sia dall'architettura che dal tipo di virtualizzazione che si sceglie.

Ogni VMM (virtual machine monitor) che gira nell'hypervisor implementa l'astrazione hardware per le *virtual machine* ed è responsabile della gestione del sistema operativo ospite che gira all'interno delle virtual machine. Ogni VMM ha il compito di partizionare e condividere la CPU, la memoria ed i dispositivi I/O per il corretto funzionamento del sistema.

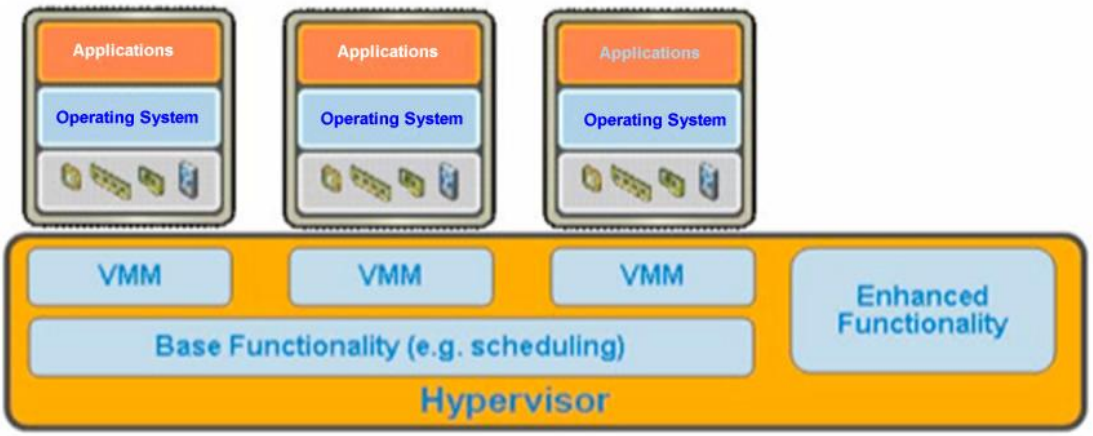


Figura 1.2 - Schema riassuntivo dei sistemi di virtualizzazione negli ambienti x86

1.1. Benefici della Virtualizzazione

Sono molteplici i motivi che oggigiorno portano le aziende ad usare la virtualizzazione nelle loro infrastrutture computazionali, sia da un punto di vista finanziario che computazionale. In questo paragrafo verranno elencati alcuni di essi.

- Da uno studio effettuato da IDC (International Data Corporation) è emerso che vi è un basso utilizzo dell'infrastruttura, tipicamente intorno al 10%-15% delle capacità complessive di elaborazione (5) . Per tale motivo con la virtualizzazione si riesce a sfruttare al massimo la potenza computazionale aggregando in un solo server fisico più server virtuali , senza perdere il livello di sicurezza poiché tali server virtuali sono completamente isolati tra loro.

- Diminuiscono i costi dell'infrastruttura fisica. La maggioranza delle infrastrutture di elaborazione deve rimanere sempre operativa, il che fa sì che i costi di consumo, raffreddamento e utenze non siano rapportati ai livelli di utilizzo. Per tale motivo aggregando più server in uno si ha un notevole risparmio sui costi dell'infrastruttura.
- Diminuzione dei costi di gestione e manutenzione dei server e del relativo personale, con la virtualizzazione si può diminuire il numero dei server fisici abbassando di conseguenza i costi di manutenzione e gestione degli stessi.
- Semplificazione del ruolo del System Admin grazie alla centralizzazione ed al monitoraggio dei vari sistemi operativi su una singola macchina fisica.
- Failover e disaster recovery. In questo caso la virtualizzazione può aiutare in maniera significativa il lavoro degli admin poichè i server virtuali possono essere agevolmente migrati da una macchina fisica ad un'altra e quindi in caso di guasto di un server i sui servizi possono essere tranquillamente trasferiti su un'altra macchina senza l'interruzione del servizio.
- Debugging dei Sistemi Operativi e delle applicazioni a run-time, incrementando i livelli di sicurezza e di isolamento tra i vari sistemi.

1.2. Problemi con l'architettura IA-32

La più comune architettura di CPU usata nei moderni sistemi operativi è IA-32 o x86-compatible. All'inizio con il chipset 80285, la famiglia x86 prevedeva due modi di indirizzamento della memoria: *real mode* e *protected-mode*.

La prima CPU creata con la virtualizzazione in mente fu la 80386, infatti uno degli obiettivi dell'epoca era quello di far girare multiple applicazioni DOS . Difatti 80386 includeva una terza modalità di indirizzamento chiamato virtual 8086 mode, il quale permetteva ad un sistema operativo di creare un ambiente di esecuzione 8086 isolato dove far girare i vecchi programmi.

Le istruzioni eseguite in real-mode limitate ad un singolo megabyte di memoria diventarono molto velocemente obsolete, così come quelle in virtual-mode bloccate per operazioni a 16-bit anch'esse diventarono obsolete con l'avvento dei sistemi operativi a 32-bit. L'esecuzione in protected-mode salvò l'architettura x86 in quanto prevedevano numerose innovazioni per il supporto al multitasking. Tra queste implementava la segmentazione dei processi, la quale non permetteva più di scrivere al di fuori del loro spazio di indirizzamento.

La famiglia x86 usa quattro livelli, o anelli, di privilegi per l'esecuzione delle istruzioni in protected-mode numerati da 0 a 3. La memoria di sistema è divisa in segmenti, ed a ogni segmento è assegnato un particolare livello. Il processore usa tale sistema di livelli per determinare se una tale applicazione può o non può accedere ad un indirizzo di memoria. Il livello 0 è considerato quello con il può alto grado di privilegi, ed ha il controllo totale del processore , mentre il livello 3 è quello con il più basso livello di privilegi.

I moderni sistemi operativi come Windows, Linux e molte varianti UNIX, sebbene riconoscano tutti e quattro i livello di esecuzione ne utilizzano solo due, il livello 0 e quello 3. Il livello 0 è comunemente chiamato Supervisor-Mode ed include il kernel del sistema operativo, mentre il livello 3 chiamato User-Mode è il livello in cui girano le applicazioni. Se un programma di un user che gira nel livello 3 tenta di accedere alla memoria al di fuori del suo segmento, un interrupt hardware ferma l'esecuzione del codice aumentando così i meccanismi di sicurezza.

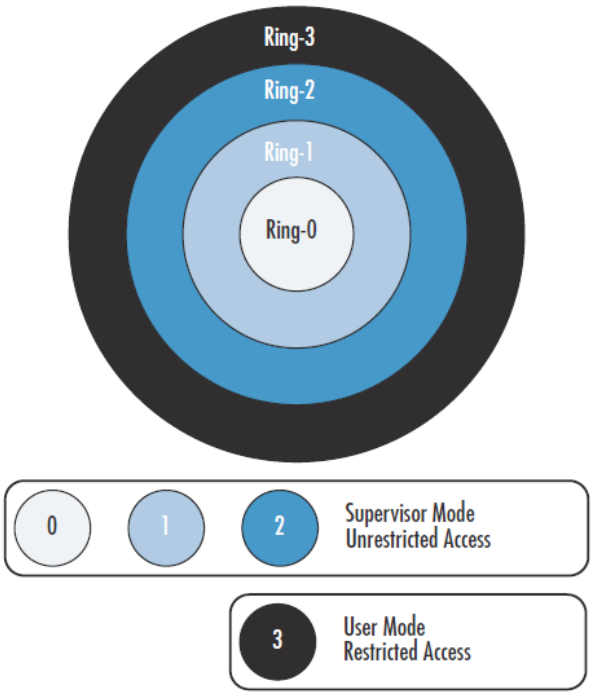


Figura 1.3 - Livelli di privilegi architettura x86

1.3. I requisiti di Popek e Goldberg

Popek e Goldberg definiscono un set di condizioni tali che l'architettura di un processore supporti la virtualizzazione, essi generalizzano le condizioni che il software deve soddisfare per provvedere alla corretta astrazioni delle macchine virtuali (VMM) . Queste condizioni, o proprietà, sono:

1. **Equivalenza**, un programma in esecuzione su una VMM deve avere lo stesso comportamento dimostrato durante l'esecuzione diretta sullo strato hardware sottostante. Tale proprietà a volte è anche chiamata *Fedeltà.*
2. **Controllo delle risorse,** la VMM deve avere il completo controllo delle risorse hardware virtualizzate usate dai sistemi operativi ospiti.
3. **Efficienza,** tale proprietà permette l'esecuzioni della maggior parte delle istruzioni senza l'intervento delle VMM.

In accordo con Popek e Goldberg il problema che gli sviluppatori delle VMM devono risolvere è quello della creazione di una macchina virtuale che soddisfa le precedenti condizioni. Tale problema viene risolto modificando l'Instruction Set Architecture (ISA) delle macchine fisiche dividendolo in tre gruppi di istruzioni:

- **Privileged instruction,** sono quelle che vengono interrotte se il processore è in *user mode* e non interrotte se sono in *supervisor mode.*
- **Control sensitive instruction,** sono quelle istruzioni che tentano di modificare la configurazione delle risorse del sistema, come per esempio aggiornare la memoria fisica o virtuale , comunicazione con i vari device , o la manipolazione dei registri globali del processore.
- **Behavior sensitive instruction,** sono tutte quelle istruzioni che si comportano in modo differente in base alla configurazione delle risorse.

Tale analisi deriva dallo studio del modello delle "architetture di terza generazione" (IBM 360, Honeywell 6000, DEC PDP-10) che è comunque abbastanza generale da essere esteso a macchine moderne. Questo modello include un processore che lavora sia in *system* che in *user mode*, che ha accesso in modo lineare alla memoria indirizzabile. Si presuppone che una parte delle istruzioni vengono eseguite solo quando si è in *system mode* e che la memoria è accessibile alla rilocazione dei registri.

1.3.1. L'architettura dei nuovi processori

Sia Intel che AMD hanno incorporato nei loro processori un set di istruzioni per il supporto alla virtualizzazione. AMD chiama la sua tecnologia AMD-V (alias Pacifica) (6), mentre Intel introduce Virtualization Technology (VT-x) (7).

VT-x estende IA-32 con due nuovi modi di esecuzione della CPU:

- ***VMX root operation* .**
- ***VMX non-root operation*.**

VMX root operation è destinata all'uso della *virtual machine monitor* (VMM) e le sue caratteristiche sono simili a quelle della tecnologia IA-32, mentre VMX non-root operation fornisce un ambiente alternativo a IA-32 controllato da VMM e designato per il supporto alle *virtual machine* (VM) . Entrambi i due metodi supportano tutti e quattro i livelli di privilegi descritti nel paragrafo precedente, permettendo al guest software di girare nel livello in cui si aspetta, e fornendo alla VMM la flessibilità di usare multipli livelli di privilegi.

VT-x fornisce anche due nuove transizioni: una transizione da VMX root operation a VMX non-root operation chiamata *VM entry* ed una da VMX non-root operation a VMX root operation chiamata *VM exit.* VM entry e VM exit sono gestite da una nuova struttura chiamata virtual machine control structure (VMCS). La VMCS include la *guest-state area* e la *host-state area* , ognuna delle quali contiene campi corrispondenti a differenti stati del processore. Con la VM entry carichiamo lo stato del processore dalla guest-state area, mentre con la VM exit salviamo lo stato del processore nella guest-state area e carichiamo lo stato del processore appena liberato dalla host-state area.

Prima dell'introduzione dell'architettura Intel VT-x le istruzioni della VM venivano intercettate e riscritte dalla VMM e provocavano nella VM un'uscita inaspettata, adesso con questo nuove operazioni la VM non si accorge della presenza della VMM e non ci sarà nessuna uscita inaspettata delle istruzioni privilegiate.

I nuovi processori AMD Pacifica, oltre alle caratteristiche dei processori Intel VT-x , modificano l'accesso alla memoria da parte della MMU e DMA, dato dal fatto che i processori AMD contengono al proprio interno la MMU (Intel non ha la MMU sul chip) e quindi non può essere ignorata così facilmente come i processori Intel.

I processori AMD-V usano due modi per gestire la memoria virtuale:

- ***Shadow Page Tables***, permette all'hypervisor di stoppare le istruzioni ogni volta che i sistemi operativi guest tentano di modificare la loro tabella delle pagine;
- ***Nested Page Tables ,*** aggiunge un altro livello di traduzione della memoria virtuale.

In aggiunta a queste due nuove caratteristiche vi è stata l'implementazione di una interfaccia *Device Exclusion Vector*, la quale maschera lo spazio di memoria dove il device ha il permesso di scrivere, in questo modo un device può scrivere solo nello specifico spazio di memoria guest a lui assegnato.

1.4. Tipi di virtualizzazione

Esistono diverse forme di virtualizzazione nel moderno mondo del I*nformation Technology.* Quello più comune è quello della **virtualizzazione lato server,** che è ciò che pensano la maggior parte delle persone quando si fa riferimento al termine " virtualizzazione ". Esistono, in aggiunta alla virtualizzazione server, altri tipi di virtualizzazione che al posto di partizionare le risorse computazionali in multiple entità fanno esattamente il contrario, cioè aggregano molteplici entità come un'unica entità virtuale nascondendo all'utilizzatore finale le vere risorse hardware.

Molte aziende hanno sviluppato software che sfruttano quest'ultimo approccio alla virtualizzazione, come per esempio la **virtualizzazione della memoria, della rete e delle applicazioni**.

1.4.1. Virtualizzazione lato server

Esistono diverse tecniche di virtualizzazione lato server e le differenze tra loro a volte possono essere molto sottili, ma sono comunque importanti nella scelta della migliore soluzione possibile. I più comuni approcci alla virtualizzazione sono:

- ✓ **Full Virtualization** (binary rewriting), tale metodologia, molto popolare grazie a VMWare (4), usa il metodo della riscrittura dei binari, ciò avviene scansionando il flusso delle istruzioni che provongono dalle macchine virtuali, e identificate le istruzioni privileggiate, vengono riscritte per puntare alle loro versioni emulate. Questo approccio non è da considerarsi tra i più permorfanti soprattutto se vi è un intenso uso dell I/O in quanto le performace di solito sono tra 80-97%

rispetto alla macchina host, con la possibilità di peggiorare ulteriormente se se il flusso contiene molte istruzioni privileggiate.

Le istruzioni che provengono dalle applicazioni vengono direttamente eseguite dal processore.

Sia l'uso del binary rewritting sia quello della diretta esecuzioni delle istruzioni delle applicazioni fanno in modo che il SO virtualizzato sia completamente astratto e per tale motivo non ha nessuna modifica del proprio kernel.

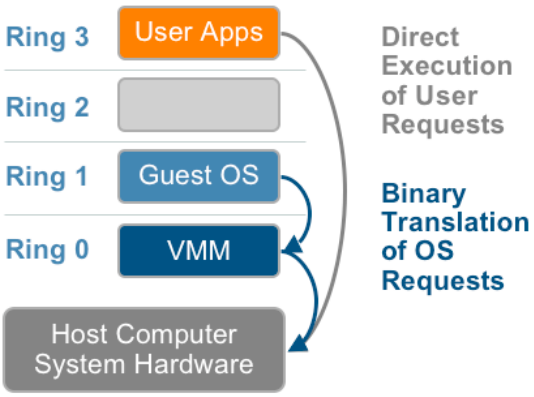


Figura 1.4 - L'approccio Full Virtualization sui sistemi x86

- ✓ **Paravirtualizzazione**, tramite questa tecnica di virtualizzazione si aggiunge un livello software tra la macchina fisica e le macchine virtuali, in tale livello software vi è l'hypervisor il quale gira nel livello di privilegi 0 e che ha il compito di gestire e schedulare le istruzioni che arrivano dai sistemi operativi virtualizzati, i quali vengono posti ad un livello più alto rispetto all'hypervisor , al livello 1.

  Tale tecnica comporta la modifica degli OS kernel per rimpiazzare le istruzioni privilegiate, che non possono essere eseguite direttamente dalla macchina host, con delle hypercalls che sono concettualmente simili alle system call.

  L'approccio paravirtualizzato è simile a quello del binary rewriting, eccetto il fatto che la riscrittura delle istruzioni privileggiate viene fatta a *compile-time* e non a *run-time* come accade nella full virtualization e questo comporta un tempo di overhead quasi nullo.

Senza l'aiuto dei nuovi processori Intel VT e AMD-V la portabilità di molti sistemi operativi non sarebbe stata possibile tramite tale approccio.

Tale tecnica di virtualizzazione è resa popolare grazie a Xen (8), un progetto *opensource* che verrà spiegato nel dettaglio nel prossimo capitolo.

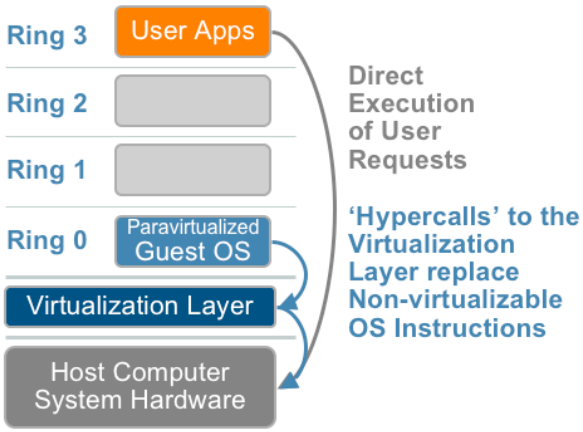


Figura 1.5 - Approccio Paravirtualizzato su sistemi x86

- ✓ **Hardware-Assisted Virtualization,** molto simile alla tecnica paravirtualizzata e a quella full virtualization. Tale tecnica usa un hypervisor che gira ad un livello più basso del livello 0 (dove abitualmente è situato nelle altre due tecniche di virtualizzazione) grazie ad una innovazione dei nuovi processori Intel-VT ed AMD-V alias Pacifica i quali introducono un livello -1 nel quale viene aggiunto un modo privilegiato di esecuzione extra delle istruzioni (nella figura Root Mode Privilege Levels).

  In questo modo i sistemi operativi girano nel livello 0, dove essi si aspettano di girare e tutte le istruzioni che fanno diretto accesso all'hardware vengono intercettate dall'hypervisor, il quale converte le istruzioni e schedula l'accesso all'hardware dei sistemi operativi ospiti .

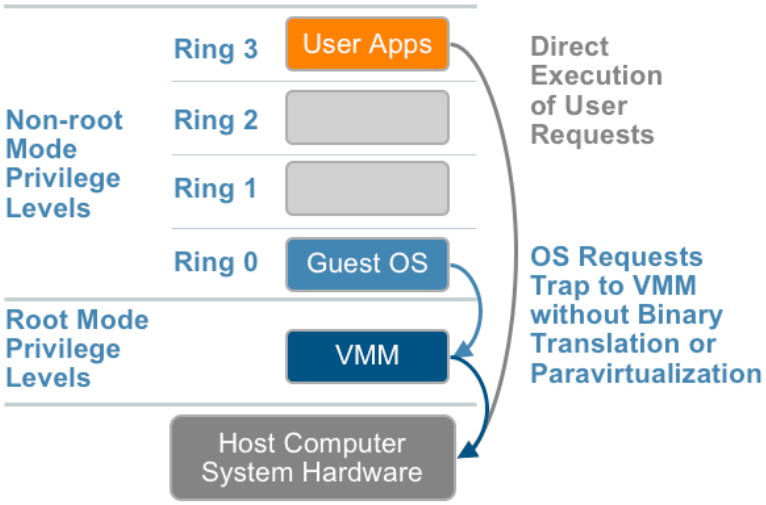


Figura 1.6 – Approccio Hardware per la virtualizzazione su sistemi x86

- ✓ **Kernel-level virtualization,** è la tecnica di virtualizzazione dove lo stesso kernel provvede alla creazione di multiple istanze isolate in *user-space,* al posto di una sola. Questa tecnica non ha bisogno di un hypervisor , ma fa girare separate versioni del kernel nelle rispettive macchine virtuali associate come se fossero dei processi in user-space sullo stesso host fisico. Un esempio di tale implementazione è User-Mode Linux (UML) (9), supportato nel mainline del kernel (10) dalla versione 2.6.20 in poi, il quale permette la gestione delle macchine virtuali direttamente dalla linea di comando senza il bisogno di un software amministrativo aggiuntivo. Il vantaggio di tale tecnica di virtualizzazione è dato dal fatto che richiede il minor spreco di risorse in quanto i sistemi operativi ospiti spesso condividono gli stessi programmi in user-space, le stesse librerie ed anche lo stesso stack di memoria.

  Tale tecnica è consigliata per chi cerca una estrema scalabilità nella concorrenza dei sistemi operativi ospiti, anche se tali sistemi operativi ospiti possono essere visti più come dei contenitori di processi in user-space che come un vero e proprio sistema operativo indipendente.

Tabella 1

Tabella Riepilogativa

Tipo	Descrizione	Vantaggi	Svantaggi
Full Virtualization	L'hypervisor fornisce una completa macchina virtuale, identica alla macchina fisica sottostante e permette al SO guest di girare senza alcuna modifica	Flessibile, non ha bisogno di modifiche di nessun genere al SO guest. Può mandare in esecuzioni SO di diversi produttori	Il SO non sa di essere virtualizzato, per tale motivo vi è dell'overhead, soprattutto se vi è un largo uso di I/O
Paravirtualizzazione	L'hypervisor crea una virtual machine molto simile all'hardware sottostante, per cui il kernel dei SO guest devono essere modificati per poter comunicare con l'hypervisor	E' una tecnica che assicura velocità di esecuzioni quasi prossime al SO nativo (0.5%-3.0% overhead range)	I SO che girano nelle macchine virtuali devono essere modificati
Hardware-Assisted Virtualization	L'hypevisor gira ad un livello -1 e in questo modo può bloccare e gestire tutte le istruzioni di accesso all'hardware fisico e permette ai SO guest di girare senza	Permette a vari SO di girare senza alcuna modifica al proprio kernel. E' considerato il metodo può semplice per ottenere le migliori performance	Richiede l'esplicito supporto nella CPU host. Usando SO senza alcuna modifica comporta un largo uso di traps e quindi cpu overheads

	modifiche		
Kernel-Level Virtualization	Il SO manda in esecuzioni diverse istanze del proprio kernel in user-space, ogni istanza è isolata rispetto alle altre	E' la più veloce tecnica di virtualizzazione, i SO virtualizzati hanno le stesse risorse del SO nativo.	E' difficile avere un completo isolamento tra i vari container, inoltre non è possibile eseguire altri SO oltre a quello che funge da host

1.4.2. Virtualizzazione della memoria

Esistono due approcci a tale virtualizzazione, il primo approccio usa la virtualizzazione come aggregazione di più dispositivi fisici in un unico dispositivo virtuale per l'immagazzinamento di grandi quantità di dati. Mentre il secondo approccio viene usato dalle macchine virtuali per la gestione e la condivisione della memoria fisica.

Le case produttrici di dispositivi di memoria offrono alte performance nelle soluzioni di immagazzinamento dei dati, infatti grazie al sistema RAID, multiple entità di immagazzinamento vengono viste come un'unica entità dal sistema operativo. Questa è considerata una tecnica di virtualizzazione in quanto tutti i driver dei singoli dispositivi fisici interagiscono tra di loro come se fossero un unico driver logico.

Un ulteriore passo in avanti in tale tecnica di virtualizzazione è dato dai dispositivi SAN (storage area network), i quali senza alcuna modifica nei sistemi operativi danno la possibilità tramite l'infrastruttura di rete di condividere la memoria fra multipli server, ed ognuno dei server ha l'illusione di avere l'uso esclusivo della risorsa.

Mentre la condivisione e gestione delle memoria fisica tra le varie macchine virtuali funziona in maniera molto simile alla memoria virtuale dei moderni Sistemi Operativi, attraverso la quale le applicazioni vedono

uno spazio di indirizzi contiguo che non corrisponde necessariamente alla memoria fisica sottostante. In questo caso il sistema operativo traduce i numeri di pagine virtuali in numeri di pagine fisiche sfruttando la tabella delle pagine. Oltre a tale tabella, per ottimizzare ulteriormente le prestazioni, vi è un ulteriore aiuto hardware chiamato MMU (Memory Management Unit) e TLB (Translation Lookaside Buffer).

Affinchè le macchine virtuali possano utilizzare tale sistema di virtualizzazione vi è bisogno di un ulteriore livello di virtualizzazione della memoria: è quindi necessario virtualizzare la MMU. In questo contesto vengono distinti i seguenti tipi di indirizzi:

- **Indirizzi macchina**, Indirizzi reali, fisici.
- **Indirizzi pseudo-fisici,** Indirizzi appartenenti alla memoria del sistema virtualizzato, a cui appaiono come indirizzi fisici.
- **Indirizzi di memoria virtuale,** Indirizzi di memoria virtuale per le applicazioni.

Il sistema operativo ospite cambia gli indirizzi virtuali in indirizzi pseudo-fisici , quest'ultimi vengono letti dell'hypervisor il quale è responsabile della reale traduzione dell'indirizzo pseudo-fisico in quello fisico della macchina.

1.4.3. Virtualizzazione della rete

Tali tecniche di virtualizzazione vengono usate da molto tempo, ma solo recentemente sono state classificate come un metodo di virtualizzazione. Le più importanti forme di virtualizzazione sono:

- **Virtual LAN (VLAN) ,** Omologate nello standard IEEE 802.1Q , le VLAN sono un metodo di creazione di multiple reti logiche che condividono lo stesso hardware fisico. Ogni rete logica è isolata dalle altre, a meno che non sia configurata in altro modo, e sono molti utili nei moderni hardware usato per instradare i pacchetti (switch , router ecc..).

- **Virtual IP (VIP),** è un IP non connesso ad uno specifico computer o interfaccia di rete, il VIP è usualmente usato da un device di rete il quale ha il compito di instradare il pacchetto ricevuto in base alle esigenze, comunemente usato come load-balancing o come ridondanza dei pacchetti.
- **Virtual private network (VPN),** inteso come la creazione di un tunnel di rete dove poter comunicare in tutta sicurezza al di sopra della rete pubblica. La VPN è usata molto spesso nel web, in quanto è l'unica forma di comunicazione sicura tra due macchine che usano un protocollo di rete pubblico, per esempio Internet.

1.4.4. Virtualizzazione delle applicazioni

La virtualizzazione delle applicazioni, o virtualizzazione software, è la nuova arrivata nella sempre più crescente famiglia della virtualizzazione. Tale tecnica rompe il paradigma ed il legame tra le applicazioni, il sistema operativo e l'hardware sottostante, usando dei pacchetti software virtuali dove poter immagazzinare dati e applicazione al posto delle solite procedure di installazione. In questo modo tali applicazioni possono agevolmente essere disattivate, resettate, o addirittura migrate da un sistema operativo ad un altro.

Di seguito elencate alcuni dei benefici di tale tecnica:

- Eliminare i conflitti con altri applicazioni ;
- Lanciare più esecuzioni dello stessa applicazione senza avere nessun conflitto ;

1.5. Riepilogo

Questo capitolo offre una introduzione alla virtualizzazione nella sua eccezione più generale, discute delle molte forme in cui è usata, e i differenti approcci. Spiega il perché ha incrementato la sua popolarità fino a diventare un hot topic ed offre una overview sui principali vantaggi di tale tecnica. Abbiamo iniziato parlando dei primi tentativi avuti negli anni '60 per poi passare alle forme sempre più raffinate di oggi, in cui lo strato hardware viene completamente sovrastato da un livello logico (hypervisor) che permette l'esecuzione di più macchine virtuali (VM) contemporaneamente. Ogni macchina virtuale ha il suo ambiente di esecuzione isolato, garantendo sicurezza ed efficienza allo stesso tempo.

Si è parlato dei problemi dell'architettura IA-32 e delle soluzione offerte da IBM ed AMD. Infine si è fatta una panoramica sulle varie implementazioni della virtualizzazione, passando da quello più comune conosciuta come " server virtualization " a quelle meno conosciute: virtualizzazione di rete, quella di memoria ed infine la virtualizzazione delle applicazioni.

CAPITOLO 2

2. Xen e la sua architettura

Grazie alle sue origini nell'open source ed alla folta comunità che ancora oggi contribuisce in maniera molto attiva allo sviluppo del software, Xen si pone ad essere uno dei più popolari ed usati monitor di macchine virtuali (VMM). Tuttavia Xen è molto di più che un altro semplice free VMM, infatti grazie a XenSource oggi Xen è utilizzato anche per le soluzioni business dei data center.

Xen hypervisor offre un potente, sicuro ed efficiente mezzo per la virtualizzazione di x86, x86_64, IA64, ARM, ed altre architetture di CPU. Supporta una vasta gamma di Sistemi Operativi tra i quali : Windows, Linux, Solaris, e varie versione di BSD.

2.1. Le origini

L'Inghilterra ha una lunga e proficua tradizione di creatività ed innovazioni tecnologiche nel campo dei calcolatori elettronici. Le origini di Xen sono dovute ad un progetto dell'Università di Cambridge conosciuto come XenoServers project (11). Lo scopo del progetto è quello di sviluppare una potente e flessibile architettura per i computer distribuiti in rete, l'elemento chiave per fare tutto ciò era la possibilità di abilitare una singola macchina a far girare differenti, o multiple, istanze di sistemi operativi contemporaneamente in un ambiente isolato e protetto.

Xen, quindi, era la VMM che originariamente era sviluppata per il supporto al progetto di XenoServers , ma con lo sforzo sempre maggiore della ricerca nel campo della virtualizzazione della CPU fisica, della memoria, dei dischi e delle reti , Xen rapidamente ebbe una vita tutta sua, indipendente dai successi e dai requisiti di XenoServers (12).

Oggi Xen è usato in migliaia di ambienti commerciali e di ricerca, ed è sviluppato oltre che da una folta community open-source anche da molte aziende del settore IT, tra cui : IBM, Intel, Hp, Novell ecc..

La gestione dello sviluppo di Xen avviene da parte di XenSource,la quale si occupa anche della parte commerciale del progetto

2.2. La filosofia di Xen

Prima di addentrarci nei dettagli implementativi, è opportuno capire i principi e le linee guida che hanno ispirato i progettisti di Xen. Capire questo vuol dire avere un quadro più ampio sull'architettura di Xen e sul perché di alcune decisioni implementative.

2.2.1. Separazione della Politica e del Meccanismo

Una delle idee chiave della filosofia di Xen è la separazione tra Politiche e Meccanismi. L'hypervisor di Xen implementa i meccanismi, mentre lascia le politiche al Domain-0.

Un meccanismo determina come fare qualcosa, una Politica come deve essere fatto (13).

Un esempio concreto di tale separazione è dato dal meccanismo di gestione dei device, infatti Xen non supporta nessun device nativamente, ma prevede un meccanismo con il quale il Sistema Operativo ospite può avere accesso diretto al device fisico usando i driver già esistente.

Anche quando il driver già esistente non è l'ideale per gestire la risorsa, in quanto non è stato creato con la virtualizzazione in mente e quindi non può gestire più richieste contemporaneamente da parte di più sistemi, Xen anche in questo caso prevede solo un meccanismo.

In questo modo i Sistemi Operativi guest devono provvedere alla gestione degli accessi, devono quindi cooperare tra loro e l'hypervisor.

2.2.2. Less is More

In contrasto con lo sviluppo degli altri software, ogni nuova release di Xen tenta di fare di meno delle precedenti. La ragione di ciò è che Xen gira al più alto livello di privilegi e per questo un bug nel kernel può compromettere tutto il sistema , comprese le macchine virtuali.

La community di Xen che sviluppa il software è relativamente piccola, perciò lo sviluppo è concentrato sugli aspetti unici dell'hypervisor piuttosto che su funzionalità già implementate in altri progetti.

Ecco perché i device vengono gestiti dai Sistemi Operativi guest, poiché se vi è già un driver per Linux per tale hardware, svilupparlo anche per Xen vorrebbe dire sono un'inutile spreco di tempo.

Per mantenere la flessibilità, Xen non forza meccanismi particolare per le comunicazioni fra domini, ma mette a disposizione dei Sistemi Operativi ospiti un semplice meccanismo, come la memoria condivisa, e permette ai vari sistemi di accedervi per scambiarsi informazioni.

Le prime versioni di Xen avevano maggiori funzionalità integrate nell'hypervisor che sono state successivamente rimosse. Un esempio è costituito dal multiplexing di rete, presente nella prima versione. La maggior parte dei Sistemi Operativi possiede funzionalità molto flessibili per il *bridging* ed il *tunneling* delle interfacce di rete. Usando queste funzionalità è stato possibile spostare il multiplexing di rete dall'hypervisor al Sistema Operativi privilegiato senza doverlo implementare ex novo.

Un ulteriore vantaggio insito nello sfruttare le funzionalità già disponibili nel Sistema Operativo privilegiato risiede nella facilità di amministrazione. Per esempio, un tool di rete come *iptables* è caratterizzato, oltre che dalla difficoltà di realizzazione, dal tempo d'apprendimento necessario per un uso proficuo. Utilizzando lo stesso tool in Xen si facilita il riuso delle esperienze già maturate.

2.3. Overview Architettura

L'architettura di Xen 3.0 con può essere vista come una suddivisione in tre livelli fondamentali: Hardware, Virtual Machine Monitor (VMM) ed i Domini.

La figura sottostante mostra l'architettura di Xen che ospita quattro VM (Dominio 0, VM1, VM2, VM3) e la Virtual Machine Monitor (VMM), la quale è responsabile dell'astrazione dell'hardware sottostante e ne gestisce l'accesso per le differenti macchine virtuali.

La tecnica di virtualizzazione utilizzata da Xen è la paravirtualizzazione (il prefisso "para" significa somiglianza, vicinanza ; quindi il termine "paravirtualizzazione" vuole indicare il concetto di "somiglianza alla virtualizzazione"), questa tecnica, così come spiegato nel capitolo precedente, crea un livello logico simile ma non identico all'hardware sottostante.

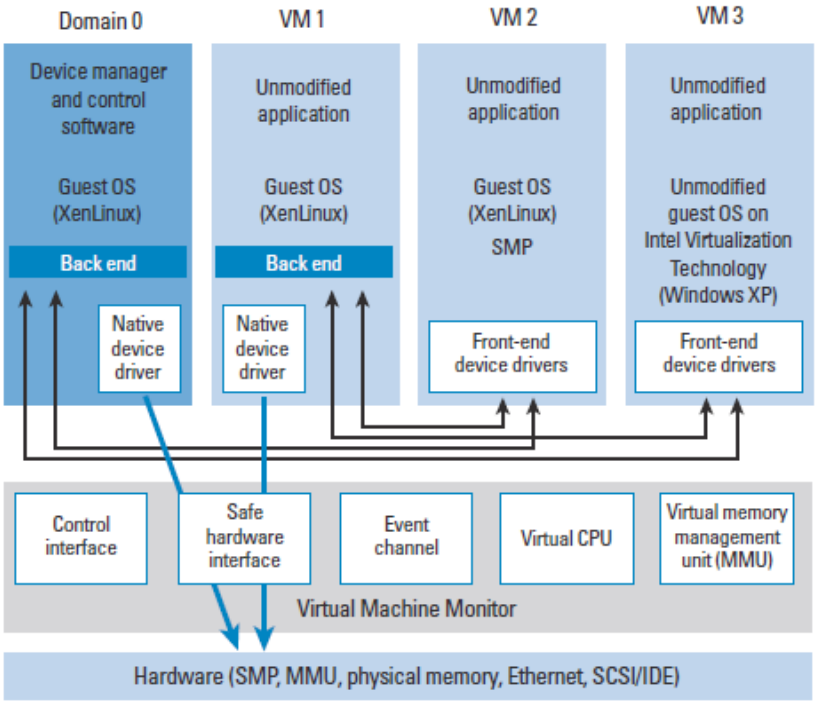


Figura 2.1 – Overview dell'approccio paravirtualizzato di Xen

La figura mostra il ruolo speciale della VM chiamata Domain-0 (DOM0), la quale ha accesso all'interfaccia di controllo della VMM ed è responsabile della creazione, distruzione e della gestione delle altre VM presenti (chiamate anche Domain-U o DOMU).

I software di gestione e di controllo girano nella Domain-0. L'amministratore può creare una macchina virtuale con privilegi speciali, nella figura VM 1, la quale può accedere direttamente all'hardware fisico

attraverso una speciale API fornita da Xen. Le altre VM possono accedere all'hardware sottostante solo grazie alla mediazione che effettua il Domain-0 tramite una sua API.

In questo esempio il Sistema Operativo della VM 1 e VM 2 è modificato per girare su Xen ed è anche " Xen-aware driver " nel senso che è a conoscenza delle API da usare per comunicare con la Domain-0 . Grazie a questo approccio si raggiungono risultati molto vicini all'esecuzione nativa dei Sistemi Operativi.

Sistemi Operativi non modificati sono supportati grazie allo sviluppo dei nuovi processori Intel e AMD, ne parleremo meglio nella sezione " Xen e l'approccio alla virtualizzazione di Intel ".

Gli sviluppatori di Xen nel prossimo futuro hanno pianificato l'inclusione del supporto alle macchine virtuali con processori simmetrici (SMP), Sistemi Operativi a 64-bit, AGP (Accelerated Graphics Port), ed il supporto ACPI (Advanced Configuration Power Interface).

Due importanti software di gestione in Xen 3.0 sono *xend* e *xm.* Il demone Xen, xend, è un programma scritto in Python ed è il responsabile dell'avvio e della gestione delle macchine virtuali. Oltre alla creazione e gestione, xend , è responsabile anche del setup della infrastruttura di rete delle macchine virtuali, offre una console server e gestisce i log di Xen.

Xm invece è la command-line interface di Xen, grazie alla quale possiamo interagire con il demone xend.

2.3.1. Il ruolo dei Domini

Come specificato nel paragrafo precedente Xen ha una struttura di sistema unica, composta dall'hardware sottostante, VMM (o hypervisor), una regione di controllo, e le stesse macchine virtuali, come mostrato nella figura successiva. L'hypervisor è il responsabile dell'astrazione del livello hardware e contiene sia API di Gestione che quella della Virtualizzazione Hardware, al suo interno vi è anche una interfaccia di controllo che permette alle macchine ospiti di comunicare direttamente o indirettamente con lo strato hardware sottostante.

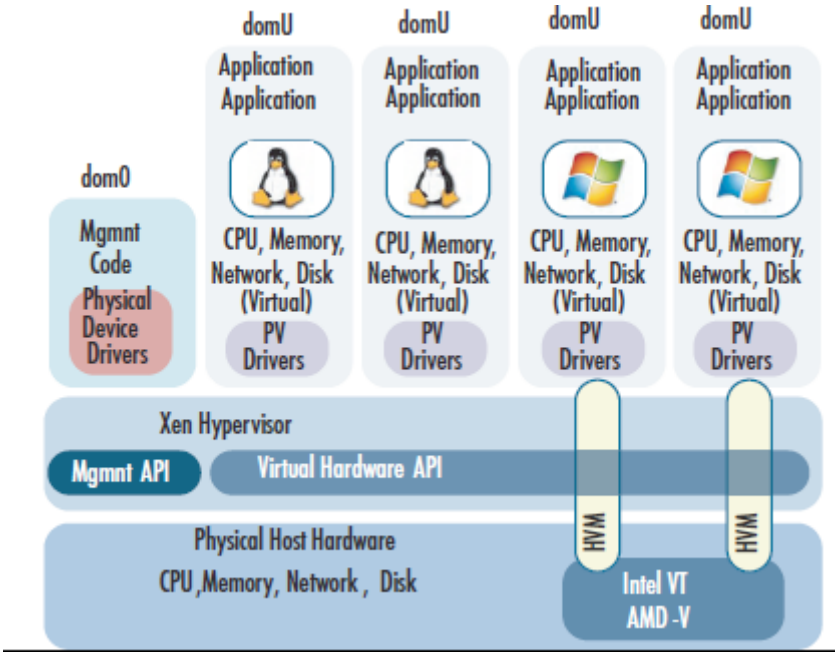


Figura 2.2 – Esempio di uso dei domini da parte di Xen

Ad interagire con l'API di gestione (Management API) è la regione di controllo la quale accede all'hardware e contiene del codice per la gestione dell'user mode.

Alla partenza, il sistema inizializza un dominio speciale che permette l'uso dell'interfaccia di controllo, conosciuto come Domain-0 (dom0). Questo dominio iniziale ospita la gestione software dell'user mode, la quale è responsabile dell'amministrazione dell'ambiente Xen , attraverso i tools della command-line oppure tramite la console XenSource Administration.

Il Domain-0 è anche responsabile della partenza e dell'interruzione dei domini meno privilegiati, domini ospiti (guest domain), chiamati anche Domain-U (domU). Gestisce tramite l'interfaccia di controllo, lo scheduling della CPU delle domU, l'allocazione della memoria, e l'accesso ai device fisici come per esempio l'accesso al disco dati o all'interfaccia di rete.

Il dominio di controllo, dom0, funziona come un normale Sistema Operativo Linux. Possiamo eseguire applicazioni in user mode, e gestire il sistema Xen, come anche installare dei driver per il supporto all'hardware. In questo modo qualsiasi driver già scritto e sviluppano per Linux può essere usato anche nell'ambiente di esecuzione Xen, e dà alle aziende IT un grossa flessibilità nella scelta dell'hardware da usare nelle proprie infrastrutture computazionali.

Nota:

Bisogna minimizzare il numero dei processi all'interno del dom0, in quanto eseguito nel livello più alto di privilegi potrebbe creare instabilità e compromettere non solo la sua esecuzione, ma anche quella di tutte le altre macchine virtuali. In aggiunta la dom0 condivide gli stessi device fisici delle domU, per cui meno si utilizzano nella dom0 e più disponibilità c'è per le macchine virtuali (domU).

L'API dell'hardware virtuale (Virtual Hardware API) include una interfaccia di controllo che gestisce l'esposizione dei device fisici, sia la creazione che la cancellazione, attraverso il seguente device I/O virtuale:

- Virtual Network Interfaces (VIFs)
- Virtual Firefall and Routers (VRFs)
- Virtual Block Devices (VBDs)

Ogni I/O device virtuale ha un Access Control List (ACL) associato ad esso. Questo ACL contiene le informazione sui domU che hanno accesso alla risorsa, le restrizioni ed il tipo di accesso, a titolo puramente esplicativo: read-only, write, e così via.

La Virtual Hardaware API è anche responsabile della gestione del trasferimento delle interazioni tra Xen e le sovrastanti domU. Lo fa attraverso l'uso delle hypercall, chiamate sincrone tra i domani e Xen, e gli eventi (meccanismi asincronici per mandare notifiche tra Xen ed i domini).

L'interfaccia dell'hypercall permette ai Sistemi Operativi ospiti di eseguire operazioni privilegiate grazie all'esecuzione di un trap nell'hypervisor. Tutto questo avviene in maniera molto simile alle system calls che i Sistemi Operativi usano per dialogare con l'hardware sottostante, eccetto del fatto che le hypercall avvengono a livello 1 mentre le system calls a livello 0.

La comunicazione tra Xen ed i domini tramite il meccanismo degli eventi può essere paragonato allo stesso modo in cui i Sistemi Operativi comunicano con i device attraverso gli interrupts. L'interfaccia di controllo prende vantaggio da questo metodo di comunicazione ed usa gli eventi anche per mandare altri segnali

importanti alle macchine ospiti, come per esempio la richiesta di terminazione. Ogni evento è contrassegnato da un univoco flag, come se fosse un particolare tipo di occorrenza. Tale occorrenza può essere usata per far sapere al dominio che una sua richiesta è stata completata o per notificare al dominio il cambiamento di stato nel suo ambiente operativo. Sebbene il responsabile dell'aggiornamento della coda degli eventi in attesa di ogni dominio è l'hypervisor, è responsabilità del dominio stesso di rispondere agli eventi di notifica, del rinvio della gestione degli eventi, ed il resettare una coda.

Tale uso dei meccanismi degli eventi fa in modo che il tutto sia molto leggero per la prospettiva dell'hypervisor e si soddisfa l'obiettivo discusso in precedenza, ciò quello di condividere l'onore della gestione con i Sistemi Operativi stessi.

Per capire meglio l'architettura di Xen vedremo in dettaglio alcuni dei sui aspetti, incluso l'astrazione della macchina virtuale e delle interfacce. Rifletteremo su come Xen virtualizza i seguenti elementi:

- CPU
- Gestione della memoria
- I/O

2.4. Virtualizzazione della CPU

Come mostrato nel capitolo precedente, la virtualizzazione della CPU nella architetture x86 presenta molte sfide, specialmente per i Sistemi Operativi che danno per scontato il fatto che girano direttamente al di sopra dell'hardware nel livello-0. Mentre l'architettura x86 prevede quattro livelli di privilegi, in realtà i Sistemi Operativi tipici usano solo due dei quattro livelli: livello-0 e livello-3. Con al tecnica della F*ull Virtualization*, esempio tipico della virtualizzazione nei sistemi x86, il Sistema Operativo coesiste con le applicazioni nel livello-3. Per proteggere se stesso, sarebbe opportuno farlo girare in uno spazio di indirizzamento unico e facendo passare indirettamente il controllo per e dalle applicazione via VMM.

Questo porta inevitabilmente all'appesantimento del sistema e alla riduzione delle prestazioni.

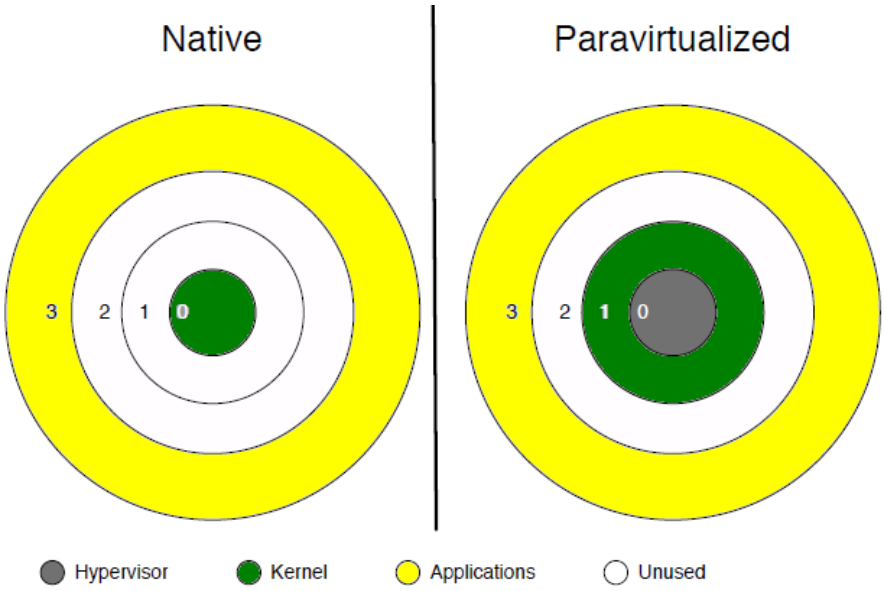


Figura 2.3 - Uso dei livelli di privilegi nei sistemi nativi e paravirtualizzati x86

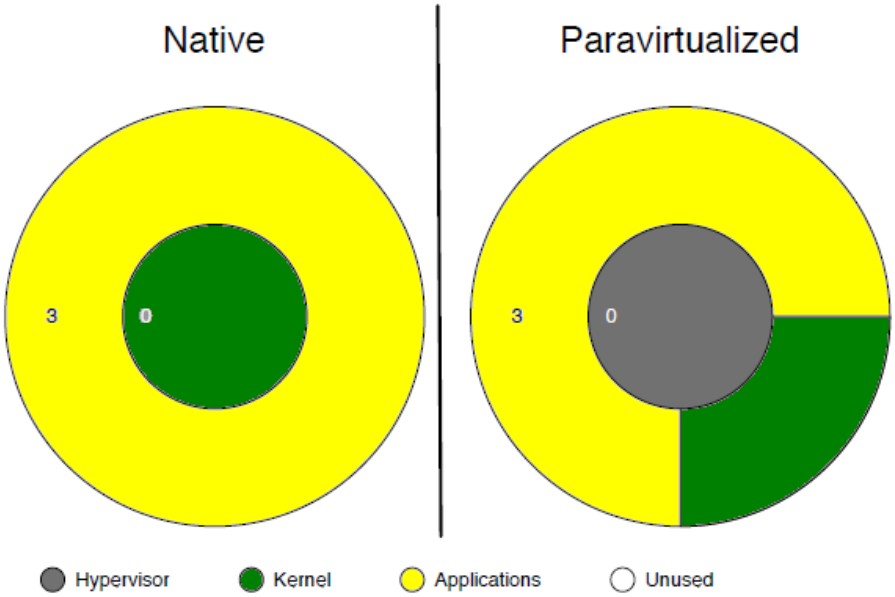


Figura 2.4 - Uso dei livelli di privilegi nei sistemi nativi e paravirtualizzati x86_64

Un approccio efficiente per risolvere questo problema è usare i livelli di privilegi lasciati vuoti, livello-1 e livello-2. Questi due livelli non vengono più usati dai Sistemi Operativi x86 sin dall'epoca di OS/2 dell'IBM (1991). Ogni Sistema Operativo che è conforme a tali specifiche (uso del livello-0 e livello-3) può, tecnicamente, essere portato su Xen. Il Sistema Operativo deve essere modificato in modo da girare nel

livello-1, in questo modo è isolato e protetto dalle altre domU ed allo stesso tempo non può eseguire le istruzioni privilegiate del livello-0 , ma offre un migliore isolamento dalle applicazioni in User Mode.

Xen paravirtualizza il Sistema Operativo ospite della domU incaricando se stesso della convalida e dell'esecuzioni delle istruzioni. Se un domU cerca di eseguire una istruzione privilegiata, questa viene bloccata e respinta dall'hypervisor. La dom0, al contrario della domU, ha un interfaccia di API che permette a tale dominio di creare e lanciare gli altri domini, questa interfaccia permette alla dom0 di avere accesso ad operazioni di controllo di alto livello come per esempio lo scheduling della CPU.

Nei nuovi processori x86_64, come per esempio IA64, i livelli di privilegi sono soltanto due.

Eccezioni

Xen ha un unico modo per gestire le eccezioni. Usando una hypercall speciale, *set_trap_table*, ogni dominio mantiene la propria unica e dedicata tabella delle eccezioni. Le eccezioni, come i fault di memoria oppure i trap di sistema, sono indirizzate usando tale tabella delle eccezioni che si trova un una parte ben riservata dello stack frame. Questo stack frame è identico a quello che si trova nella piattaforma hardware x86 sottostante, l'unico cambiamento riguarda il gestore dei page-fault. In condizioni normali questo gestore legge l'indirizzo *faulting* da un registro privilegiato. Siccome ciò non è consentito con la paravirtualizzazione, l'indirizzo viene scritto dall'hypervisor in uno stack-frame esteso. Quando si verifica un'eccezione, il gestore di Xen effettua una copia dello stack-frame dell'eccezione sullo stack del sistema operativo ospite e passa il controllo al gestore appropriato registrato nella tabella.

Di conseguenza, le eccezioni sono gestite in due maniere.

La prima si occupa delle system call usate nei Sistemi Operativi ospiti e si avvale dell'uso di un gestore delle eccezioni accelerato che è registrato da ogni Sistema Operativo ospite ed il processore vi si può accedere direttamente. Queste system call non hanno bisogno di essere eseguite nel livello-0 per cui hanno la stessa velocità di quelle native, per cui Xen non ha bisogno della validazione.

La seconda è per i page fault, una difficile e quasi impossibile eccezione da affrontare senza l'aiuto dell'hypervisor. La tecnica usata per le chiamate di sistema non può essere fatta all'infuori del livello-0,

quindi solo Xen ha i permessi per utilizzarla, per cui l'eccezione viene eseguita da Xen e poi memorizzato il valore del registro per il recupero da parte del Sistema Operativo ospite.

CPU SCHEDULING

La schedulazione dei processi nei sistemi virtualizzati è un aspetto molto critico. Per far in modo che lo scheduling avvenga con performance vicine a quelle native, lo schema della schedulazione deve essere ottimizzato e non deve sprecare cicli di CPU. Questi tipi di schemi sono denominati work-conserving, cioè essi non consentono che la CPU sia inutilizzata finché c'è la capacità sufficiente per eseguire le istruzioni e ci sono istruzioni da eseguire. Tale schema assegnerà le istruzioni intrappolate dall'hypervisor delle macchine virtuali e poi le passerà alla CPU. Se il carico di lavoro non è congestionato, questo schema funzionerà come una coda FIFO (first in – first out), altrimenti se la coda dei processi del processore è congestionata, le istruzioni verranno messe in coda ed eseguite in base alla priorità e al peso.

Uno dei CPU sheduler che sono oggi disponibili in Xen è basato sulla schema del Borrowed Virtual Time (BVT). Questo è un algoritmo ibrido che implementa sia il work-conserving che il meccanismo della low-latency, chiamato anche domain wake-up, il quale da priorità ai Sistemi Operativi che si sono svegliati a causa di una notificazione di un evento. Quest'ultimo è importante nell'hypervisor per minimizzare gli effetti della virtualizzazione nel sottosistemi di Sistemi Operativi creati per girare in maniera tempestiva.

Xen offre due schemi di schedulazione, Simple Earliset Deadline First (sEDF) e Credit scheduler, e da la possibilità di implementare il proprio scheduler mettendo a disposizione delle API.

Lo schema Credit è ottimizzato per SMP (symmetric multiporcessing) ed è la migliore scelta, infatti BVT è uno scheduler Credit.

Grazie ai parametri che possiamo passare all'API di gestione dello scheduler Credit possiamo fare in modo di assegnare ad una certa macchina virtuale solo un sottoinsieme di CPU fisiche.

Per esempio immaginiamo che una certa applicazione giri su una domU e che possa utilizzare solo due dei quattro processori fisici disponibili, quindi solo CPU-2 e CPU-3. Anche se la CPU-0 e CPU-1 hanno dei cicli liberi e quindi non compiono lavoro, tale applicazione non potrà sfruttarne l'uso.

TEMPO

All'interno delle infrastrutture virtuali, il cronometraggio del tempo di CPU diventa un problema molto serio e può diventare fonte di confusione per i Sistemi Operativi ospiti. Essi hanno bisogno di conoscere sia il tempo reale che quello virtuale. Il tempo reale, riferito al tempo di wall-clock, è gestito dall'hypervisor; mentre , il tempo virtuale diventa ancora più importante per i Sistemi Oeprativi guest per lo scheduling delle operazioni time-sensitive . Ci sono diversi concetti da prendere in considerazione quando si parla di tempo in Xen. I seguenti sono una lista di tali concetti e una breve descrizione:

- **System time**, un contatore a 64-bit tiene il conto dei nanosecondi passati dall'accensione della macchina. Quindi il system time rappresenta il tempo trascorso dalla creazione o dall'esecuzione del dominio.
- **Wall-clock-time,** questo è il tempo dalla prospettiva del dom0, ed è mantenuto dal dom0. Il tempo è rappresentato in secondi e nanosecondi contando dal 1 gennaio 1970. Quale riferimento chiave del tempo complessivo, è importante che sia il più accurato possibile. Un modo affinchè lo sia è quello di utilizzare il client Network Time Protocol (NTP) all'interno del dom0.
- **Domain virtual time ,** questo tempo è simile al system time, ma aumenta solo quando il dominio è schedulato. Tale tempo viene stoppato quando il dominio viene de-schedulato. Il Virtual Time è importante perché la condivisione della CPU che un dominio riceve è basata sulla velocità con cui si incrementa tale tempo virtuale.
- **Cycle counter time,** questo tempo è usato per fornire un riferimento preciso del tempo. Il cycle counter time è usato per estrapolare gli altri tempi, o il tempo di riferimento. E' importante che questo tempo sia sincronizzato tra tutte le CPU fisiche.

Xen mantiene una stampa del tempo sia del *wall-clock-time* che del *system time* ed esporta tali valori tramite la memoria condivisa tra Xen e gli altri domini. Xen fornisce il cycle counter time quando tutte le stampe dei tempi sono calcolate, questo permette ad ogni macchina ospite di estrapolare accuratamente il system time ed il wall-clock-time.

La stampa del tempo contiene al suo interno anche il numero della versione, tale numero viene incrementato due volte per ogni aggiornamento. La prima volta prima che viene incrementato prima di aggiornare il valore, la seconda subito dopo averlo aggiornato, in questo modo un dominio che legge il valore e compara i due valori può capire immediatamente se il valore è correttamente aggiornato oppure se deve ritornare a leggere il valore dopo che è stata eseguita correttamente la procedura di aggiornamento.

Oltre al tempo reale e a quello virtuale, Xen utilizza due timer per aiutare i domini a mantenere correttamente aggiornato il tempo. Il primo è un timer periodico, che ogni 10 ms manda un evento timer a tutti i domini (svegli). Questo evento timer viene gestito dai Sistemi Operativi ospiti quando sono inattivi, così da non compromettere l'esecuzione di altri processi.

Il secondo è una *hypercall* chiamata *set_timer_op* usata dalle macchine ospiti per implementare il concetto del timeout.

2.4.1. Xen e l'approccio alla virtualizzazione di Intel

Intel Virtualization Technology (IVT) è una implementazione nei nuovi processori Intel che aggiunge un nuovo set di istruzioni che può essere usato dalla VMM per la creazione ed il supporto alle VM. In termini di livelli di privilegi della architettura dei processori vuol dire posizionare la VMM un livello sotto al livello 0, permettendo così alle VM di posizionarsi al livello 0.

L'immagine sottostante raffigura un semplice stato del processore IVT. Il sistema inizializza la VMM eseguendo l'istruzione VMXON. La VMM può poi gestire ogni VM lanciando l'istruzione VM Entry . Lo stato del processore della VMM è automaticamente salvato e caricato lo stato della VM.

In certe condizioni, la VM può rilasciare il controllo alla VMM, tali condizioni sono chiamate VM exit e possono essere causate da diversi fattori, compresi gli interrupt esterni, page fault, e da altre istruzioni privilegiate eseguite dalla VM.

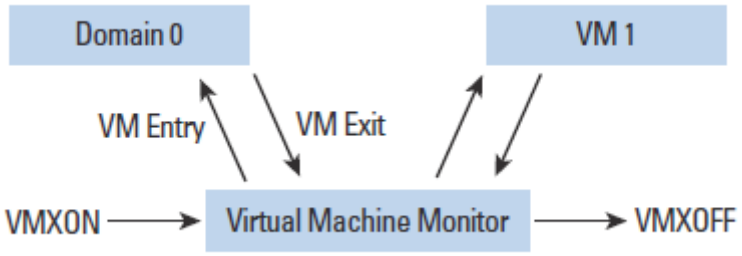


Figura 2.5 – Esempio di utilizzo delle nuove istruzioni dei processori Intel VT-x

Il supporto alla virtualizzazione a livello di processore permettere di far girare le VM con un Sistema Operativo senza alcuna modifica, in questo caso Xen ha la capacità di implementare la *full virtualization* grazie al supporto della IVT. In questo caso Xen provvede alla creazione di un BIOS virtuale e dei device anche'essi virtuali .

2.5. Virtualizzazione della memoria

La virtualizzazione della memoria è molto probabilmente il compito più difficile che deve assolvere la VMM. Xen è responsabile della gestione e dell'allocazione della memoria fisica per i domini, e garantisce il corretto uso della *paginazione* e della *segmentazione*. Poiché molti domini condividono la stessa memoria, bisogna fare molta attenzione all'isolamento. L'hypervisor deve essere sicuro che due domini non privilegiati, o domU, non facciamo accesso alle stessa memoria, ogni pagina o directory di tabelle aggiornata deve essere validate dall'hypervisor così da far in modo che ogni dominio possa manipolare solo le proprie tabelle.

Anche per la segmentazione accede la stessa cosa, l'hypervisor controlla che i segmenti dei domini non si sovrappongano oppure che non siano validi. Per capire bene come Xen gestisce la memoria, di seguito

parleremo in maniera approfondita: **l'allocazione delle memoria, traduzione degli indirizzi virtuali, segmentazione e tipi di tabelle delle pagine.**

Allocazione della memoria

Xen, in aggiunta alla sua porzione di RAM per i propri usi, riserva una piccola porzione anche per ogni sistema virtuale. Questa allocazione è indipendente dalla piattaforma in cui gira, come mostrato in figura.

NOTA:

> Per l'architettura x86 la massima quantità di memoria utilizzabile da Xen è di 4 GB, 8 Gb per l'architettura x64.

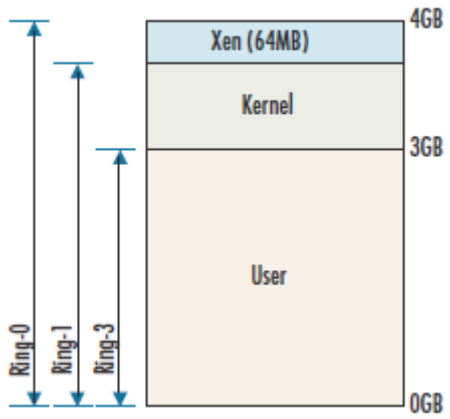


Figura 2.6 – Uso dello stack di memoria nell'architettura x86

Nell'architettura 32-bit x86, la segmentazione è usata per proteggere l'hypervisor di Xen dal kernel.

Per quanto riguarda l'allocazione della memoria fisica per le domU, ogni dominio ha un massimo ed un uso corrente della memoria, infatti Xen ha implementato un sistema di driver che in automatico gestiscono la massima allocazione possibile per ogni Sistema Operativo virtualizzato. Questo permette di impiegare l'area di memoria inutilizzata da altre risorse del sistema.

A causa di questo costante cambiamento di allocazione della memoria, combinato con la creazione e distruzione delle macchine virtuali, essa è allocata e liberata in base alla granularità del livello delle pagine.

Per questo motivo Xen non garantisce che i domini riceveranno un allocazione contigua della memoria fisica in uso, e questo crea molti problemi ai Sistemi Operativi compatibili con l'architettura x86.

Xen per evitare il blocco di tali sistemi implementa l'uso della *memoria pseudo-fisica* , la quale ha il compito di nascondere la segmentazione della memoria fisica reale e mostrare ai sistemi virtuali una segmento di memoria continua. La memoria della macchina è formata da page frame da 4KB, e vengono numerate in maniera lineare senza tener conto della loro locazione fisica.

Proprio grazie a queste metodo ogni dominio pensa di avere la propria porzione di memoria contigua e che inizi sempre con il frame 0. Questo risultato è ottenuto grazie all'uso di due tabelle:

- la prima tabella può essere letta dall'hypervisor e da tutti i domini e mappa i page frame sulla memoria pseudo-fisica;
- la seconda tabella viene fornita privatamente ad ogni dominio e traduce dagli indirizzi pseudo-fisici agli indirizzi fisici della memoria;

Usando entrambi le tabelle è possibile implementare l'astrazione delle memoria di cui ha bisogno la macchine virtuale per poter far girare il proprio Sistema Operativo.

2.5.1. Tabella delle pagine e segmentazione

Nell'architettura x86, la memoria è divisa in tre tipi di indirizzi :

- **Logico,** è quell'indirizzo che viene usato al livello delle applicazioni e che può essere anche relazionato direttamente alle memoria fisica cosi come non lo può essere.
- **Lineare,** è usato da molte architetture non-x86 per indirizzare la memoria, un indirizzo lineare è quella parte della memoria acceduta partendo da 0. Tutti i successivi byte puntano al prossimo numero sequenziale (0,1,2 e così via) fino alla fine delle memoria. Mentre l'architettura x86 usa uno spazio di indirizzamento segmentato, diviso in segmenti da 64KB, ed il registro di segmento punta sempre all'ultimo segmento acceduto.

- **Fisico,** è l'indirizzo che rappresenta direttamente i bit della memoria fisica

L'indirizzo fisico può differire da quello logico, e quando accade la MMU (memory management unit) traduce l'indirizzo logico in quello fisico

La CPU usa due unità per trasformare l'indirizzo logico in quello fisico, la prima chiama *segmented unit* e la seconda *paging unit.*

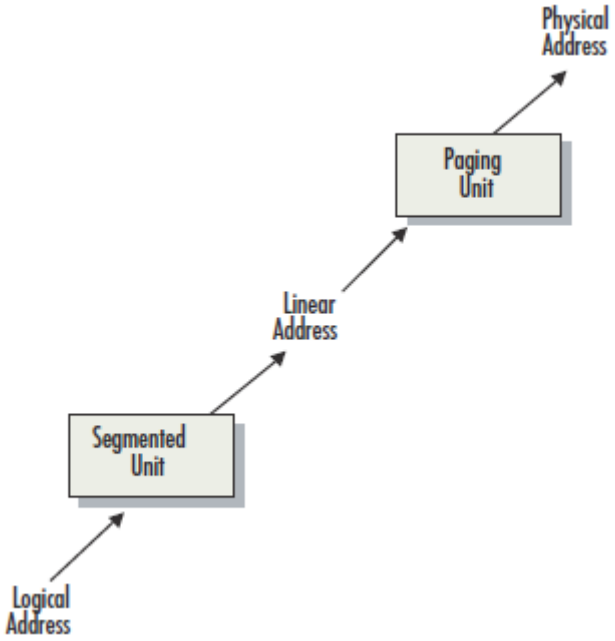


Figura 2.7 – Schema di esempio traduzione da un indirizzo logico ad uno fisico

L'idea alla base dell' unità di controllo del modello di segmentazione è che la memoria è gestita usando un set di segmenti. In sostanza ogni segmento ha il suo spazio di indirizzamento , e tale segmento è formato da un *base address* che contiene l'indirizzo fisico di qualche locuzione di memoria, e *lenght value* che specifica la lunghezza di tale segmento. Ogni segmento ha un campo di 16-bit chiamato *segment identifier* o *segment selector* . Ogni archietettua x86 ha alcuni registri programmabili chiamati *segment register ,* i quali memorizzano questi segment selector. Questi registri sono *cs* (code segment), *ds* (data segment), e *ss* (stack segment). Ogni segment identifier identifica un segmento rappresentato da un 64-bit (8 byte) *segment descriptor.* Questo segment descriptor è memorizzato sia in una tabella globale dei descrittori (GDT) che in una tabella locale dei descrittori (LDT).

In tutto questo processo l'hypervisor ha solo il ruolo di validare l'aggiornamento della tabella delle pagine(utilizzata per tradurre l'indirizzi lineare in quello fisico), così da assicurare sicurezza ed isolamento ad ogni Sistema Operativo ospite.

Poiché il coinvolgimento di Xen è solo per gestire le operazioni di scrittura , le macchine ospiti possono mappare e accedere ad ognuno delle sue frame page senza l'intervento dell'hypervisor, indipendentemente dal tipo di page frame.

2.5.2. Traduzione dall'indirizzo virtuale a quello fisico

La gestione delle memoria è direttamente influenzata dalla tabella delle pagine e dal Translation Lookaside Buffer (TLB). La tabella delle pagine è la struttura di dati che permette al computer di mappare l'indirizzo virtuale con quello fisico. Il TLB è una cache usata dalla CPU per migliorare la traduzione degli indirizzi virtuali, ha un numero fissato di entry che contengono una parte dell'indirizzo della tabella delle pagine che traducono l'indirizzo virtuale in quello fisico.

Il TLB è tipicamente una memoria, la quale viene acceduta dall'indirizzo virtuale tramite un tag e ritorna l'indirizzo fisico corrispondente. Se nel TLB non viene trovato riscontro a quello specifico tag allora la traduzione viene dirottata alla tabella delle pagine, spesso risultando molto più onerosa sia in termini di risorse che di tempo.

Associare ad ogni entry del TLB un indirizzo di memoria è una ottima soluzione per la coesistenza tra l'hypervisor ed i Sistemi Operativi ospiti, poiché non c'è bisogno di svuotare ogni volta il TLB quando vi è il trasferimento dell'esecuzione.

Molti processori RISC come SPARC, Alpha, e MIPS hanno un software di gestione del TLB. Purtroppo l'architettura x86 non ha tale possibilità e delega all'hardware la gestione del TLB, infatti i TLB miss sono inviati al processore in automatico dall'hardware. Per cercare di ottimizzare le performance, tutte le traduzioni delle pagine valide per il corrente indirizzo di memoria dovrebbero essere accessibili nella tabella

delle pagine dall'hardware. Poiché nell'architettura x86 il TLB non può essere configurato dal processore, e che nel caso di switch dei processi il TLB deve essere completamente aggiornato per far posto ai nuovi indirizzi fisici, gli sviluppatori di Xen hanno pesato a due possibili soluzioni: la prima prevede che è il Sistema Operativo ospite il responsabile dell'allocazione e gestione dell'hardware che fa accesso alle tabella delle pagine con il minino coinvolgimento di Xen; la seconda prevede che il kernel di Xen sia presente nei primi 64MB di ogni spazio di indirizzamento, evitando così lo svuotamento del TLB all'entrata e all'uscita dell'hypervisor.

Ogni volta che un Sistema Operativo ospite richiedi una nuova tabella delle pagine , magari a causa di un nuovo processo, è lui stesso a preoccuparsi delle creazione e gestione utilizzando il proprio spazio di indirizzamento come se lo avesse fatto Xen. In questo modo il sistema ospite rinuncia ai diritti privilegiati di scrittura ed ogni cambiamento deve essere prima validato dall'hypervisor (14).

Infine Xen supporta anche l'uso delle *shadow page table* (tabella delle pagine ombra). Quando utilizza questa tecnica, Xen crea un replica identica della tabella delle pagine inizializzata ed allocata dal Sistema Operativo ospite. In questo modo il sistema ospite legge e scrive sulla versione residente sulla sua memoria, mentre Xen copia tutti i cambiamenti in real time nella shadow page table. Se il dominio commette un fault quando aggiorna le proprie tabelle delle pagine, Xen manda un segnale di notifica consigliandolo di gestire il page fault o, in maniera più seria, termina il dominio che ha generato il fault. In tutti e due le maniera l'isolamento tra i due domini è mantenuto.

Questa tecnica è molto utili soprattutto nei Sistemi Operativi dove la CPU o la memoria non può essere paravirtualizzata, come per esempio Microsoft Windows

2.6. Virtualizzazione I/O

La virtualizzazione dei device I/O è molto simile a quella della virtualizzazione della CPU o della memoria, segue la stessa scuola di pensiero del less is more (paragrafo 2.2.2). Ossia la VMM può emulare l'hardware

sottostante oppure può lavorare assieme ai driver dei domini per cercare le migliori prestazioni di I/O, sicurezza e affidabilità.

Con l'approccio full virtualization, la VMM non deve solo gestire l'hardware sottostante ma deve anche essere compatibile con il software dei device driver usati, oltre che emulare anche gli interrupts, la condivisione delle risorse e l'instradamento dei dati. Questi compiti spesso portano ad un sostanziale overhead sulla VMM a scapito delle performance.

La filosofia di Xen fa in modo di delegare il supporto all'hardware ai domini ospiti, di solito al dom0, oppure anche agli altri domU. Questo per rendere il più semplice possibile il kernel di Xen e quindi migliori risultati in termini di affidabilità e prestazioni.

In Xen i driver dei device di solito di suddividono in quattro componenti principali:

- **Real driver**
- **The bottom half of the split driver** oppure **Backend Drivers**
- **The shared ring buffer**
- **The top half of the split driver** oppure **Frontend Drivers**

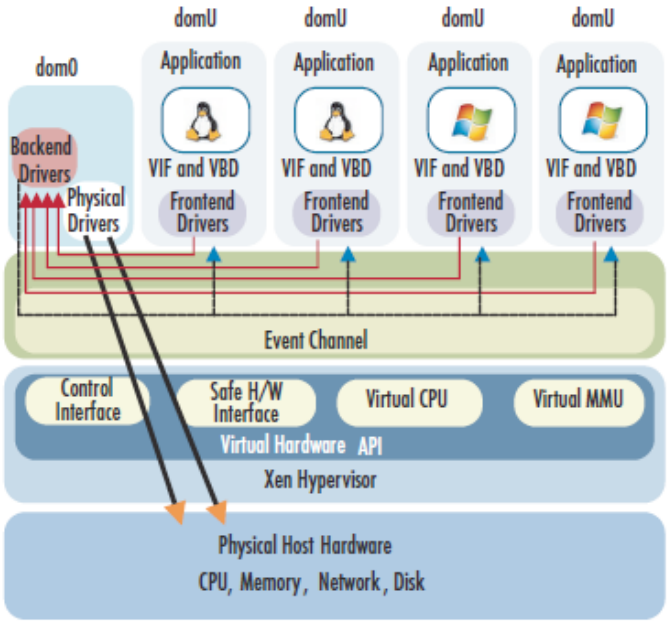


Figura 2.8 – L'architettura di Xen nella gestione dei driver dei dispositivi

Il real driver è il driver vero e proprio presente nel Sistema Operativo, e che grazie alla buon progettazione del kernel non deve essere modificato per portalo nell'architettura Xen.

Il backend driver (la metà inferiore del driver splittato) ha due funzioni principali: quello di gestire il multiplexing, permette l'accesso alla risorsa da parte di più domini, e quello di offrire una interfaccia generica dell'hardware.

Il frontend driver si comporta nei domU come se fosse il vero e proprio driver della macchina guest, ma in realtà non comunica direttamente con l'hardware sottostante ma con il backend driver. C'è bisogno di un metodo per esportare tale meccanismo anche agli altri Sistemi Operativi. Questo è tipicamente fatto usando la shared ring buffer (anelli di buffer nel segmento delle memoria condivisa). Il frontend del driver inizializza una pagina di memoria con una struttura ad anello ed esporta, tramite il meccanismo della memoria condivisa, l'informazione a Xen rendendola pubblica a tutti gli altri domini.

Il frontend (la parte superiore del driver), che gira nei domini non privilegiati, è tipicamente molto semplice. Ha bisogno di ispezionare la memoria condivisa per cercare la pagina immessa dal backend. Molti driver di Xen usano il meccanismo del buffer ad anello per la comunicazione dei dati attraverso i vari domini.

Una ruolo fondamentale nella comunicazione dei device è svolto da Xenstore (ne parleremo più approfonditamente di seguito) il quale è responsabile dell'inizializzazione delle parte di memoria condivisa tra i vari domini per lo scambio dei dati.

2.6.1. Anelli dei Device I/O

Con il ruolo che ha l'hypervisor di offrire un livello di astrazione che protegge ed isola i domini dal I/O dei device, è importante trovare un meccanismo che permetta di trasferire i dati in maniera veloce e con poco overhead. Tale specifiche vengono soddisfatte dal I/O rings, il quale ha code di entrata ed uscita. Queste code non contengono nessun dato, ma sono dei puntatori al buffer I/O allocato dal Sistema Operativo ospite referenziato dai descrittori I/O. Il concetto è molto simile a quello del produttore-consumatore (15).

Quando un dominio vuole utilizzare un device I/O , mette una richiesta nella coda delle richieste(Request Cycle nel disegno) chiamata *request producer* . Questa richiesta avanza o aggiorna il puntatore alla richiesta che è condiviso tra tutti i domini e l'hypervisor.

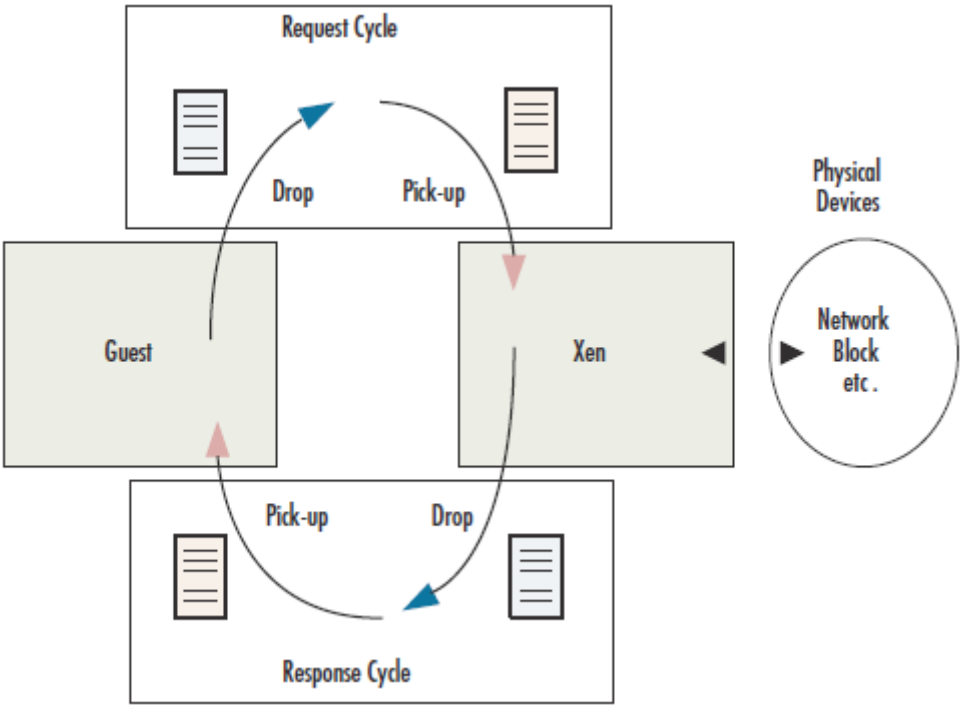


Figura 2.9 – Esempio d'uso degli anelli di I/O nel contesto di Xen

Una volta arrivata una richiesta dal produttore (request producer) nella coda delle richiesta, Xen rimuove questa richiesta dalla coda e crea una *request consumer* , associandola ad un puntatore interno a se stesso. Dopo che Xen ha comunicato con l'hardware la risposta viene messa nella coda delle risposte (Response Cycle nella figura) *response producer* aggiornando o avanzando un puntatore a tale risposta. A questo punto il dominio che ha fatto la richiesta legge la risposta, ed avanza o aggiorna il proprio puntatore alla risposta chiamato *response consumer*.

Tale meccanismo è asincrono, per cui l'ordine di arrivo e smistamento delle richieste non è importante. Ci sono dei vantaggi nell'uso del meccanismo in maniera asincrona. Il primo è quello di permettere a Xen il riordino delle richieste di I/O, in modo da avvantaggiare o migliorare le prestazioni dei singoli domini.

La seconda è data dal fatto che non c'è nessuna notifica formale fatta dal domino ospite a Xen quando formula una richiesta di accesso al driver fisico, queste permette di inserire più richieste nella coda delle richieste ed avvertire Xen una sola volta tramite hypercall. Così facendo si da la possibilità a Xen di prendere e servire le richieste in maniera molto più efficiente decidendo quale servire per prima, tale approccio è molto utile nella gestione del traffico di rete.

Questa metodologia usata nella gestione delle richieste di I/O ha un ulteriore vantaggio nei sistemi virtualizzati, infatti un dominio che ha effettuato delle richieste può diventare "dormiente", quindi senza consumare cicli del processore inutilmente, e risvegliarsi solo quando Xen comunica l'avvenuta risposta della sua richiesta da parte del driver fisico.

In qualche caso specifico il dominio ospite può mettere delle barriere sulla riordinazione delle richieste effettuato da Xen, in quanto ha bisogno che tali richieste vengano fatte in batch (serialmente) per motivi di integrità dei dati, come la scrittura anticipata dei log.

2.6.2. Network I/O

Per facilitare la virtualizzazione della rete I/O, Xen offre un astrazione del router-firewall virtuale (VFR). Il VRF ha una o più interfacce virtuali (VIFs), le quali fanno da tramite tra il VRF ed il dominio ospite ed agiscono come delle vere e proprie interfacce fisiche, avendo due anelli di I/O, uno per la trasmissione ed un altro per la ricezione. Il ruolo del VFR è quello di controllare il flusso dei dati tra le varie VIF, in maniera molto simile alle regole di un firewall.

I due anelli I/O gestiscono il carico delle rete per e verso il Sistema Operativo ospite. Il sistema ospite mette la richiesta nell'anello di trasmissione, Xen prende una copia dell'header e del descrittore di questa richiesta la valida usando le regole del VRF, e poi se tutta la procedura è andata a buon fine alloca il frame di memoria nella memoria condivisa ed esegue l'effettivo trasferimento dall'interfaccia fisica alla VIF. Il driver di back end è il responsabile delle gestione dei pacchetti, ed il suo compito può essere suddiviso in tre sezioni fondamentali:

- **Validation,** garantire che un tentativo di generare traffico invalido sia bloccato, questo può essere fatto analizzando il MAC e IP sorgente del pacchetto
- **Scheduler,** usando un semplice scheduler round-robin (15) il backend può gestire la coda dei pacchetti in trasmissione tra i vari domini
- **Logging end accounting,** il driver può configurare le regole di logging dei pacchetti in transizione tra le varie VIF, per esempio ad ogni trasmissione di SYN o FIN

Per ricevere i dati , i Sistemi Operativi ospiti usano i page frame liberi come buffer temporanei, come se non fossero in un sistema virtualizzato. Xen agisce da demultiplexer per i dati in arrivo, e li smista ai vari VIF ricontrollando di nuovo le regole del VRF cambiando i buffer di pagina con i page frame delle memoria condivisa.

Tabella 2

TABELLA RIEPILOGATIVA

Gestione delle Memoria	
Segmentazione	Non può inserire descrittori di segmenti privilegiati e non Può coincidere con l'estremità superiore della linea dello spazio Di indirizzamento.
Paginazione	I Sistemi Operativi ospiti hanno accesso diretto alla tabella delle Pagine hardware, ma l'aggiornamento viene validato dall'hypervisor.
CPU	
Protezione	I Sistemi Operativi ospiti girano ad un livello di permessi Superiore rispetto a quello di Xen, non può girare al livello 0.

Eccezioni	Il Sistema Operativo ospite devi inizializzare una tabella per La gestione delle eccezioni.
System Call	Il Sistema Operativo ospite devi inizializzare un gestore per le System call, permettendo così la comunicazione diretta tra una Applicazione ed il Sistema Operativo. Qualche System Call può Essere gestita direttamente dal dominio ospite senza L'intervento di Xen.
Interrupt	Gli interrupt hardware sono sostituiti con dei meccanismi di eventi di notifica.
Tempo	Il Sistema Operativo ospite deve essere a conoscenza sia del tempo reale, quello di wall-clock che quello virtuale
Device I/O	
Rete e Dischi	I device virtuali sono molto semplici da raggiungere. I dati vengono trasferiti grazie a due anelli asincroni di I/O, e gli interrupt di comunicazione vengono gestiti come eventi di notifica.

2.7. Networking in Xen

Xen crea, di default, sette coppie di interfacce ethernet virtuali che si possono utilizzare nel dom0. Ogni coppia è pensabile come due interfacce di rete connesse tramite un cavo virtuale crossato interno. Tali interfacce sono chiamate `vethX` e `vifX`. In particolare `veth0` è connessa a `vif0.0`, `vethl` con `vif0.l`, e così via fino a `veth7`, connessa con `vif0.7`. Queste interfacce possono essere utilizzate

configurando IP e MAC sul lato `vethX` e connettendo `vifX` a un bridge. Le modalità con cui si utilizzano queste interfacce varia a seconda del tipo di networking utilizzato sul dom0. Dalla parte dei vari domU Xen crea coppie di interfacce virtuali `vify.x` ed `ethx`, dove y corrisponde all'id del dominio, mentre x numera le interfacce presenti sul domU. Le `ethx` risiedono sui domU e sono dette dispositivi di frontend , mentre le `vif` risiedono tutte sul dom0, e sono chiamate dispositivi di backend. Lo schema utilizzato per le interfacce virtuali è quello illustrato in Figura.

Quando la macchina domU viene spenta , l'interfaccia ethernet virtuale corrispondente viene eliminata.

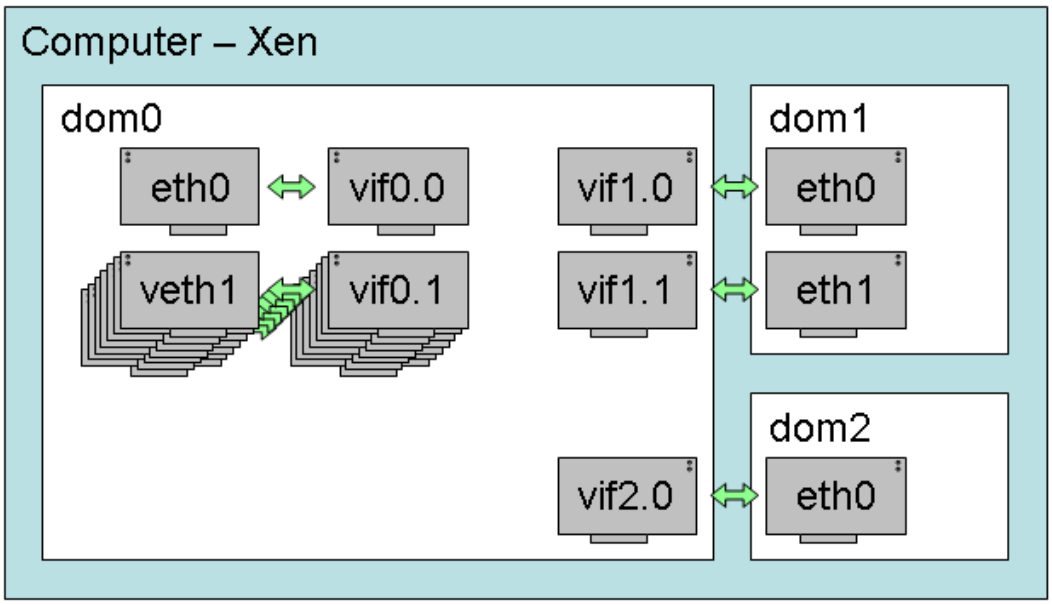


Figura 2.10 – Il collegamento tra le interfacce fisiche e virtuale di Xen

L'implementazione del networking in Xen può avvenire secondo tre configurazioni possibili:

- Bridged Networking
- Routed Networking
- NAT Networking

2.7.1. Bridged Networking

La configurazione di rete di default che usa Xen è quella del bridge, la quale permette ai singoli domini di apparire sulla rete come host indipendenti tra loro. L'instradamento dei pacchetti avviene grazie alla suite di iptables, tramite le tre chains: PREROUTING, FORWARD e POSTROUTING. In questo modo possiamo controllare i pacchetti che saranno instrada da e verso i nostri domini, in poche parole il bridge funge da switch virtuale ed instrada i pacchetti che arrivano sull'interfaccia fisica verso i domini corrispondenti.

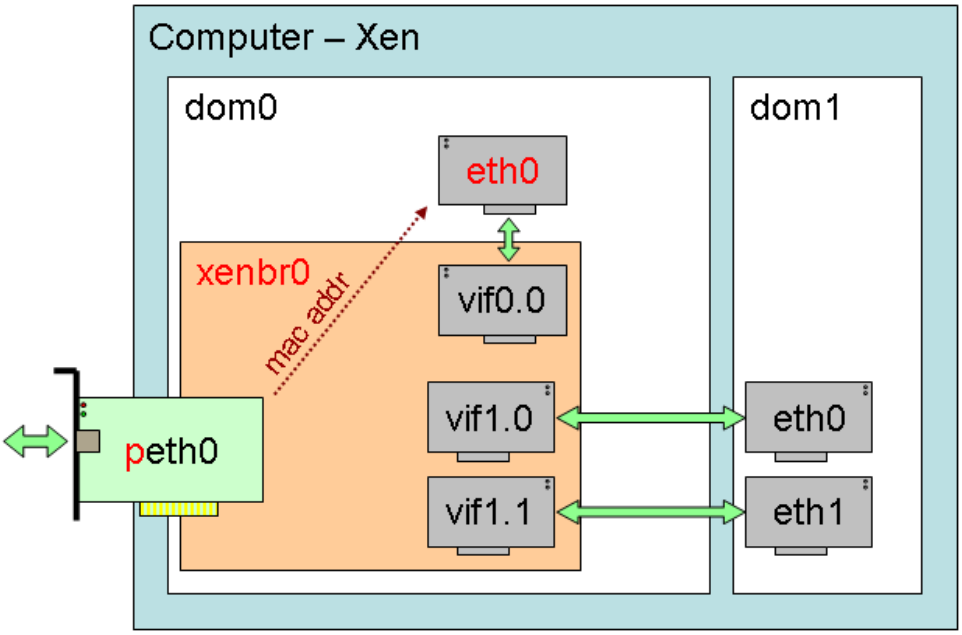


Figura 2.11 – Schema delle interfacce in modalità Bridge

I pacchetti arrivano sull'interfaccia fisica, il driver reale del dom0 gestisce l'arrivo e gli invia a sua volta al peth0.peth0 in quale è legato al bridge. Il tutto avviene a livello ethernet, nessun indirizzo IP è assegnato al bridge o al peth0. Solo ora i pacchetti vengono distribuiti sulle varie interfacce virtuali connesse alle proprio domU , ci sono un certo numero di vifX.Y collegate al bridge e l'unico modo per sapere dove instradare i pacchetti è dato dall'indirizzo MAC contenuto all'interno del pacchetto. Dalle varie vifX.Y i pacchetti confluiscono nelle interfacce interne dei domini ospiti.

Tutto questo meccanismo avviene grazie a xend, il quale appena parte esegue i seguenti passi per mettere in piedi tale meccanismo:

1. crea un nuovo bridge chiamato `xenbr0` ;
2. disattiva l'interfaccia reale `eth0`;
3. riassegna IP ed indirizzo MAC della `eth0` sono alla interfaccia virtuale `veth0` (virtual ethernet 0);
4. l'interfaccia reale `eth0` viene rinominata come `peth0` (physical ethernet 0);
5. l'interfaccia virtuale `veth0` è rinominata come `eth0` ;
6. le interfacce `peth0` e `vif0.0` sono agganciate al bridge `xenbr0`;
7. bridge, `peth0`, `eth0` e `vif0.0` vengono riattivate.

Si nota che il dom0 ha una interfaccia virtuale (`eth0`) differente da quella fisica (`peth0`). Questo è utile per motivi di flessibilità: è possibile impostare un firewall per proteggere il dominio privilegiato (che ha l'importante compito di amministrare tutti gli altri domini) senza interferire con il traffico diretto agli altri domini.

2.7.2. Routed Networking

Tramite questa tipologia creiamo un link point-to-point tra il dom0 ed ogni domU, una regola di routing è aggiunta nel dom0 per ogni macchina virtuale ospite, in questo modo la domU deve avere un indirizzo IP statico. Il DHCP non può essere utilizzato, in quanto la richiesta broadcast del dhcp non può arrivare correttamente al domU in quanto ancora non è stata scritta la regola di routing verso quella macchina virtuale.

In questo caso xend effettua le seguenti operazioni:

1. copia l'indirizzo IP da `eth0` a `vif<#id>.0`;

2. attiva l'interfaccia di backend `vif<#id>.0`;

3. aggiunge una riga alla tabella di routing per l'indirizzo IP del domU

(specificato nel suo file di configurazione), che punta all'interfaccia virtuale `vif<#id>.0`.

2.8. XenStore

XenStore è il sistema di immagazzinamento dei dati condiviso tra i domini ospiti. Sebbene sia una parte importante del sistema Xen, non c'è nessuna hypercall associata con esso ma viene acceduto grazie alle pagine delle memoria condivisa e gli event channel . XenStore ha una struttura gerarchica, gestita dalla dom0 ed ha il ruolo fondamentale di gestire le notifiche, prendere o riportare le informazioni sullo stato del sistema dai Sistemi Opeativi ospiti.

Tutte le informazioni sono immagazzinate in un database, situato in */var/lib/xenstored/tdb* .

2.8.1. L'interfaccia di XenStore

L'hypervisor non è a conoscenza dell'esistenza di XenStore. Esso è gestito da dom0 , e poi essere acceduto da molti altri driver dei device. La struttura base per immagazzinare le informazioni consiste in due anelli, uno per ogni direzione. Le richieste di aggiornamento o quelle di informazione sul corrente stato sono posizionare nel primo anello. Le risposte e le notifiche asincrone dei cambiamenti sono inserite nel secondo anello. Il primo anello è scritto dai vari domU e letto dal dom0; l'altro anello è scritto dal dom0 e letto dalle varie domU.

XenStore è composto da directory, ed ogni directory contiene altre directory o delle chiavi. Ogni chiave ha un valore associato, può anche essere visto come un array associativo. La struttura è molto simile a quella di un file system, anche se l'uso è in qualche modo differente in quanto è stato progettato per trasmettere

piccole quantità di informazioni tra i domini . Per esempio la locazione di un device virtuale è memorizzata in XenStore.

Come molti file system , XenStore supporta anche il modello transazionale per I/O. Il quale consiste nel raggruppare più richieste in una sola transazione, assicurando che essa sarà eseguita atopicamente.

2.8.2. Lettura e scrittura di una chiave

Lo scopo primario di XenStore è quello di memorizzare una coppia di *key-value* in una locazione di facile accessibilità da parte dei domini e dei tools.

Ci sono tre metodi per accedere a XenStore. Il primo attraverso i tolls in phyton che Xen mette a disposizione via command-line, questo è il miglior modo di accedervi dato che è molto semplice per gli amministratore scrivere piccoli shell script per l'amministrazione e gestione dei domini.

CODICE da Wiki Xen (16)

```
#!/bin/sh

function dumpkey() {
   local param=${1}
   local key
   local result
   result=$(xenstore-list ${param})
   if [ "${result}" != "" ] ; then
      for key in ${result} ; do dumpkey ${param}/${key} ; done
     else
      echo -n ${param}'='
      xenstore-read ${param}
   fi
}

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

Il secondo è ad un livello di programmazione più basso e si tratta del linguaggio C, attraverso le sue API riusciamo a comunicare attraverso il Sistema Operativo direttamente ai device di XenStore.

CODICE da The definite guide to Xen Hypervisor (17)

```
/ Read a response from the response ring /

 int xenstore_read_response ( char message , int length)
 {
      int i ;
      for ( i=xenstore->rsp_cons ; length > 0 ; i++,length --)
      {

            XENSTORE RING IDX data ;
            do
            {

                  data = xens tore->r s p prod - i ;
                  mb( ) ;
            } while ( data == 0) ;

             / Copy the byte /
             int ring_index = MASK_XENSTORE_IDX( i ) ;
             *message = xenstore->rsp [ring_index] ;
             message++;
       }
 xenstore->rsp_cons = i ;
 return 0 ;
 }
```

L'ultimo metodo è quello di accedervi direttamente tramite l'interfaccia del kernel-space. Questo approccio è quello più usato nelle ultime versioni di Xen, in quanto il kernel mette a disposizione delle interfacce molto semplici per interagire con XenStore.

2.9. Hypercall

Poiché il kernel dei Sistemi Operativi ospiti è costretto a girare ad un livello di privilegi più alto, il livello 1, ed in tale livello non si possono eseguire tutte le istruzioni, si è cercato un modo di bypassare tale restrizione usando le hypercall.

Tale problema non è nuovo. I programmi in *user space*, che girano nel livello 3, spesso hanno bisogno di leggere input dalla tastiera, mandare i dati attraverso la rete, tutte operazioni che hanno bisogno di interazioni con l'hardware e quindi un livello di privilegi più alto. La soluzione a tali problemi è stato risolto grazie all'uso delle *system call*, un meccanismo con il quale un programma in user space riesce a comunicare con il kernel. I passi fondamentali per l'implementazione di una system call sono 5 :

1. Si effettua il salvataggio dello stato del processo nei registri o all'interno dello stack.
2. Si emette uno specifico interrupt, o si invoca una speciale istruzione system call.
3. Si salta all'interno del gestore delle interruzioni del kernel, come risultato dell' interrupt.
4. Si processa la system call al livello di privilegi del kernel.
5. Si scende ad un livello di privilegi più basso ed il controllo ritorna al processo.

Figura 2.12 – Passaggi fondamentali di una chiama ad una System Call

Nella architetture x86 è di uso comune usare interrupt 80h per le system call, sebbene nelle CPU più moderne esistono le istruzioni SYSENTER/SYSEXIT o SYSCALL/SYSRET per l'invocazione veloce delle system call.

Lo stesso meccanismo viene utilizzato da Xen. Le Hypercall vengono generate dal kernel ospite nello stesso modo con cui le system call vengono generate da un programma utente, l'unica differenza è che viene usato l'interrupt 82h al posto del 80h.

Questo metodo di invocamento è ormai deprecato dalla versione di Xen 3.0 , al suo posto viene mappata una pagina di memoria all'interno dello spazio di indirizzamento del dominio ospite quando esso viene

inizializzato. In questo modo l'hypercall viene chiamata indirettamente tramite la pagina delle hypercall (*hypercall page*). L'hypercall viene emessa chiamando un indirizzo all'interno di questo indirizzo.

CODICE: una macro usata per chiamare una hypercall (17)

```
#define _hypercall1(type , name , a1 )
({
     Long __res , __ign1;
     asm volatile (
                  "callhypercall_page+("STR(__HYPERVISOR##name)"32)"
                  : "=a" (__res) , "=b" (__ign1)
                  : "1" ((long)(a1))
                  : "memory" );
     (type) __res;
})
```

Come mostrato dal codice , l'hypercall viene chiamata usando il numero di offset 32 dall'inizio della pagina delle hypercall. Poiché la grandezza delle pagine nell'architettura x84 è di 4KB possiamo avere un massimo di 128 hypercall, ma ad oggi solo 37 hypercall sono state definite ed altre 7 sono riservate nel futuro al particolare tipo di architettura usato.

Per un raffronto con i sistemi Unix-like, la versione / di UNIX supporta 64 system call , mentre una distribuzione moderna di FreeBSD arriva fino a 450 system call. Può sembrare strano che Xen supporta meno hypercall di UNIX, ma questo è in accordo con la filosofia di Xen del less is more dove l'hypervisor non fa nulla che sia assolutamente necessario.

2.9.1. Fast System Call

Come visto in precedenza, il livello extra di in direzione di Xen introduce inevitabilmente dell'overhead nell'invocazione delle system call. Piuttosto che il rilascio di un interrupt ed la successiva gestione da parte del gestore degli interrupt da parte del kernel, l'interrupt viene dal gestore degli interrupt di Xen e poi passato al kernel tramite un evento. Questo aggiunge due context switch (dall'applicazione a Xen e da Xen al kernel), il quale porta ad una ulteriore rallentamento del sistema in quanto l'architettura x86 non è particolarmente veloce nel context switching.

Per affrontare questo problema, Xen implementa una interfaccia per velocizzare la chiamata alle system call. Il Sistema Operativo ospite usa l'interrupt 80h, in questo modo bypassa il gestore di Xen, in questo modo è possibile passare dal livello 3 al livelli 1 senza interagire con l'hypervisor.

2.10. Riepilogo

In questo capitolo si è parlato in maniera più approfondita dell'architettura di Xen e del suo approccio alla virtualizzazione, la paravirtualizzazione. Un approccio che grazie all'hardware sempre più evoluto e specifico, rende Xen il sistema più performante e quello con il più alto grado di compatibilità con i sistemi operativi, compresa la famiglia di Microsoft Windows.

La chiave del suo successo è sintetizzata nel suo motto, less is more . Grazie all'efficienza raggiunta nella virtualizzazione della CPU, della memoria, e del I/O, Xen offre un ambiente di sviluppo intelligente e facilmente gestibile sia da parte dello sviluppatore che dei sys administrator. L'efficiente uso dell'anello di I/O, assieme alle hypercall, permette di avere un tempo di overhead molto basso e una gestione ottimale dell'accesso alle risorse in comune.

CAPITOLO 3

3. Sistemi Cluster

A seguito della drastica riduzione del rapporto costi-prestazioni delle apparecchiature hardware, una parte significativa delle risorse informatiche di un'azienda risiede sulle scrivanie. Grazie alle sempre maggiori innovazioni in termini di velocità delle interfacce di rete e di potenza di calcolo dei processori desktop, rendono i sistemi cluster un appetibile strumento per la riduzione dei costi nell'ambito del calcolo parallelo. I cluster costruiti con prodotti e con hardware di uso comune stanno giocando un ruolo fondamentale nella ridefinizione del concetto dei supercomputer.

Il trend nel settore del calcolo parallelo è quello di passare da piattaforme specializzate a sistemi di uso generale molto meno costosi, formati da componenti molto meno performanti assemblati per workstation a singoli o multi processori o PC. Questo approccio ha una serie di vantaggi, tra cui il mettere in piedi sistemi di calcolo dato un certo budget, che è facilmente espandibili e si presta a innumerevoli classi di applicazione e carichi di lavoro.

In questo capitolo verranno dapprima discussi i principi generali relativi ai cluster, successivamente saranno esposte le problematiche principali con cui ci si confronta nei sistemi cluster e le diverse soluzioni che il mercato opensource offre per il cluster management. In questo contesto si inserisce Rocks, la distribuzione linux per cluster presa come riferimento per il presente lavoro di tesi. Ne verranno delineate le caratteristiche e gli aspetti principali. Successivamente verranno illustrati i principi fondamentali dell'attività di benchmarking in ambito cluster e i gli strumenti utilizzati nel presente lavoro di tesi.

3.1. Che cosa è un cluster ?

Un cluster è l'unione di risorse hardware e software cooperanti per uno scopo comune. Gestire un sistema è qualcosa di più complesso della gestione della singola risorsa hardware o software. Ad esempio in un cluster si hanno un numero variabile di macchine chiamate *compute node*, ovvero macchine di calcolo, e diversi frontend per gestirli. La tipica applicazione di un cluster è l'HPC (High performance computing), ovvero il calcolo ad elevate prestazioni, applicazione dell'informatica richiedente un'elevata affidabilità hardware e software, nonché meccanismi semplificati per la gestione dell'intero sistema.

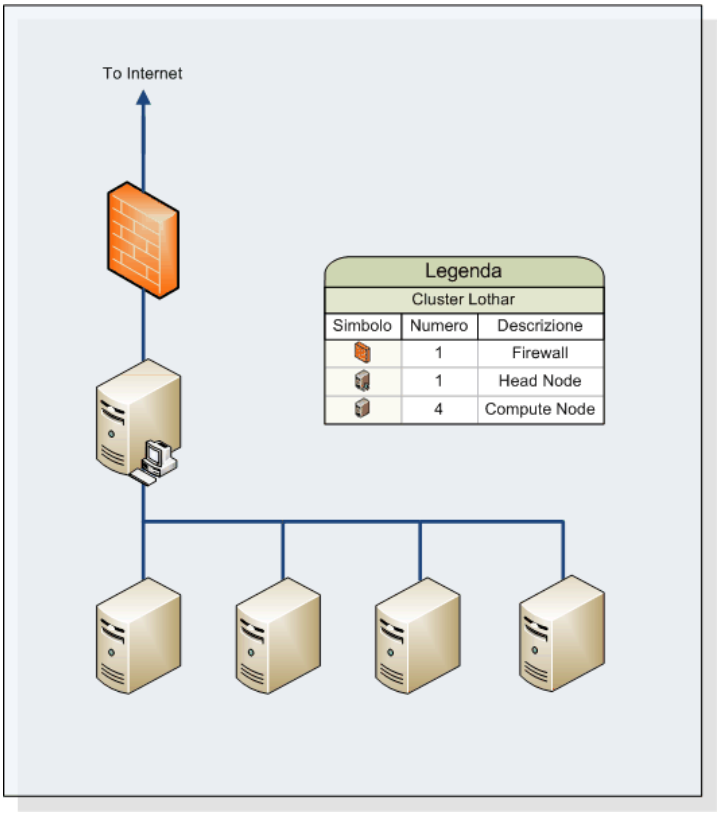


Figura 3.1 – Schema Generale del cluster Beowulf

Lo sviluppo dei sistemi cluster avviene grazie alla domanda sempre crescente di risorse di calcolo, le quali portano, in generale, alla crescita esponenziale della capacità elaborativa dei computer. Nonostante l'impressionante incremento della capacità computazionale, le richieste che determinate applicazioni, specie in campi come la chimica o la fisica, effettuano in termini di potenza di calcolo, di memoria e di

spazio disco, sono ormai divenute troppo elevate per ottenere risposte soddisfacenti da un singolo computer. Per ovviare a questo problema, negli ultimi anni si è assistito al rapido diffondersi di strumenti di calcolo basati sull'utilizzo di diversi computer connessi in rete. Intuitivamente, si potrebbe pensare che l'enorme disponibilità di calcolo generata da cluster o macchine parallele specializzate con centinaia di nodi di elaborazione sia sufficiente, invece non è così .

Di fronte al crescente numero di applicazioni che necessitano grandi quantità di risorse computazionali, diventa sempre più impellente recuperare cicli macchina potenzialmente disponibili per l'elaborazione. Questa tendenza ha portato alla diffusione di sistemi che consentono di allocare i programmi di calcolo su grandi insiemi di computer, soprattutto quando questi risultano essere scarsamente utilizzati. La maturazione di questo processo ha portato allo sviluppo del Grid Computing, che consente in modo diverso l'esecuzione di applicazioni di vario tipo su una griglia di computer eterogenei, la quale può comprendere elaboratori paralleli altamente performanti, workstation ed anche singoli PC.

A partire dai primi anni '90 le tecnologie di rete si sono ampiamente diffuse, soprattutto in ambito accademico. Questo ha favorito all'interno di istituzioni accademiche, ditte ed istituti di ricerca, una graduale sostituzione degli ambienti computazionali basati sui grandi mainframe, che avevano caratterizzato gli anni '70, con stazioni di lavoro interconnesse ad alta velocità .

Con l'avvento di questa distribuzione delle risorse computazionali è stato coniato il termine *Cluster*. Il cluster risponde all'esigenza di utenti ed amministratori di sistema di assemblare insieme più macchine per garantire prestazioni, capacità, disponibilità, scalabilità ad un buon rapporto prezzo/prestazioni ed allo stesso tempo ridurre il più possibile il costo di gestione di tale sistema distribuito.

Di cluster si parla sin dalla "preistoria" dell'Informatica (sistemi VAX della Digital -1983- , sistemi IBM JES3 per mainframe 360/370) ma spesso se ne è parlato male confondendo le problematiche hardware con quelle software. Infatti per cluster si può intendere un modello architetturale all'interno del quale si possono ritrovare tutti i paradigmi riconosciuti, ovvero una sorta di contenitore generale da non mettere in contrapposizione con i vari tipi e modelli in esso contenuti.

Con il termine " Cluster " si intende quindi una particolare configurazione hardware e software avente per obiettivo quello di fornire all'utente un insieme di risorse computazionali estremamente potenti ed

affidabili. Con l'avvento dei servizi di rete Internet e del web in particolare, i cluster sono cresciuti sempre più in popolarità, proponendosi come sistemi altamente scalabili ed efficienti.

Quando siti web o server che offrono servizi vengono acceduti da molti utenti, si può prevedere l'utilizzo di un cluster al fine di distribuire il carico complessivo sui diversi computer (i " nodi " del cluster) o per garantire la continuità del servizio a fronte dell'indisponibilità di uno dei nodi.

In generale questo tipo di organizzazione è assolutamente trasparente all'utente finale; ad esempio vengono adottate soluzioni per cui all'indirizzo web del sito in questione corrisponderanno fisicamente i diversi nodi del cluster.

Ovviamente, affinché ciò funzioni come descritto macroscopicamente in precedenza, occorre adottare una serie di accorgimenti software ed hardware particolari.

Tra questi, va citato il servizio di Network Information System (NIS) noto anche con il nome Yellow Pages (YP), che consente di distribuire le principali mappe di sistema ad un insieme di macchine, in modo che l'utente sia identificato univocamente sull'intero pool dei nodi. Attraverso il servizio di Network File System (NFS), è invece possibile consentire all'utente (ed al sistema) di condividere lo stesso spazio disco sopra i diversi computer, mediante un meccanismo client-server che consente di montare dischi del server sui client sia in modo statico che dinamico, garantendo un'assoluta trasparenza delle operazioni, coerenza e sincronizzazione dei dati.

In particolari ambienti in cui l'esigenza di risorse computazionali è molto sentita, rivestono un ruolo molto importante i servizi di Batch Queueing, i quali consistono, in sostanza, di un gestore della distribuzione del carico di lavoro su un pool macchine, anche eterogenee. Il sistema rende quindi disponibili all'utente delle code, entità in grado di accettare applicazioni in esecuzione, in modo da consentire il bilanciamento del carico e l'esecuzione di job in base alle specifiche dell'utente ed alla disponibilità dei diversi sistemi. Il sistema decide inoltre quando si verificano le condizioni per eseguire il programma, ed infine, al completamento dello stesso, si occupa della restituzione dei risultati sul richiesto flusso di output.

Esistono dei prodotti evoluti che permettono il bilanciamento del carico all'interno del pool di macchine anche di applicazioni interattive, permettendo quindi il checkpoint dello stato di avanzamento della

computazione e la migrazione da un nodo ad un altro se per qualche ragione il nodo su cui viene lanciato il job dovesse essere spento.

La nuova frontiera va ben oltre questo approccio, fornendo dei meccanismi di migrazione automatica di processi e/o dati per garantire una maggiore omogeneità di distribuzione dei carichi di lavoro.

Si può pertanto parlare propriamente di cluster quando un insieme di computer completi ed interconnessi ha le seguenti proprietà:

- I vari computer appaiono all'utente come una singola risorsa computazionale ;
- Le varie componenti sono risorse dedicate al funzionamento dell'insieme ;

3.2. Tipologie di Cluster

Diverse motivazioni hanno contribuito alla crescente affermazione dei sistemi distribuiti:

- un ambiente distribuito è scalabile, e può essere facilmente incrementato per venire incontro a nuove esigenze computazionali ;
- la richiesta di risorse computazionali può essere distribuita fra un insieme di macchine invece di essere confinata ad un singolo sistema, eliminando i cosiddetti "colli di bottiglia" e fornendo, quindi, migliori prestazioni ;
- la disponibilità delle risorse e dei servizi viene fortemente incrementata ;

Di contro, è possibile individuare alcune problematiche che scaturiscono proprio dalla scelta di adottare un ambiente distribuito:

- l'amministrazione di decine o centinaia di sistemi è molto più onerosa di quella del singolo sistema ;
- la gestione della sicurezza cresce esponenzialmente all'aumentare degli host presenti nella rete ;
- la gestione di risorse condivise via rete, è più complessa rispetto a quella locale.

Al fine di limitare l'effetto delle problematiche legate all'adozione di sistemi distribuiti e favorire la diffusione di sistemi di calcolo ad alte prestazioni implementati attraverso limitate risorse economiche è stato introdotto, dapprima in ambito accademico e poi anche in ambito aziendale, il concetto di cluster di personal computer. Questo ha fatto anche si che il sistema operativo Linux giocasse un ruolo chiave per la definizione e l'implementazione di modelli di clustering, basati su tale concetto, atti ad affrontare e risolvere le diverse esigenze specifiche ed i vari macro–problemi.

3.2.1. Il modello generale

Un cluster può , in generale , essere visto come un insieme di nodi ognuno dei quali è un fornitore di servizi a tutto l'insieme. All'interno del cluster si avranno quindi diverse tipologie di nodi come File Server, Batch Server, Tape Server o nodi che offrono servizi specializzati come NIS, DNS, Web, Mail, FTP e così via.

Per passare da un insieme di sistemi interconnessi ad un cluster sono necessari almeno tre elementi:

- un meccanismo di condivisione delle informazioni amministrative ;
- un'architettura con forte centralizzazione delle immagini di sistema (auspicabile ma non essenziale);
- un meccanismo di gestione della schedulazione di richieste delle risorse.

Quest'ultimo punto (insieme alla distribuzione sui vari nodi di tali richieste) è il primo valore aggiunto che deve essere previsto dopo lo startup iniziale.

Ovviamente la topologia del cluster e soprattutto la sua architettura vanno selezionate in base alle particolari esigenze degli utilizzatori ed al particolare contesto in cui si viene ad operare.

Mentre un responsabile IT potrebbe essere interessato al mantenimento della massima affidabilità dei propri server o alla diminuzione del tempo generale di esecuzione delle applicazioni e dei servizi di punta

dell'organizzazione, un fisico delle alte energie potrebbe essere interessato alla simulazione su larga scala o alla disponibilità di enormi quantità di dati sperimentali fisicamente distribuiti su WAN.

Si possono quindi individuare almeno quattro categorie distinte:

1. **cluster per l'affidabilità dei servizi ;**
2. **cluster per l'alta disponibilità dei servizi (High Availability, HA) ;**
3. **cluster per il bilanciamento del carico (Load Balancing, LB) ;**
4. **cluster per il calcolo parallelo (High Performance Computing, HPC).**

3.2.2. Cluster per l'affidabilità dei servizi

Tale soluzione prevede la predisposizione di un insieme di sistemi che devono mantenere copie separate dei servizi e che rispondano comunque ad un unico nome di host virtuale con il quale i client interrogano il singolo servizio. Essenzialmente, avendo a disposizione un cluster con forte centralizzazione delle immagini di sistema ed applicative, diverse tipologie di servizi di rete si prestano per far sì che ogni nodo del cluster possa, se necessario, diventare un nodo-servizio.

Questo permette di avere una scalabilità dei servizi a costi irrisori rispetto ad un eventuale cambio architetturale (ad esempio da low a high level server). Il punto cruciale di questo modello è il servizio di virtualizzazione dei nomi a livello DNS (fino ad arrivare a criteri avanzati di load-balancing) e la gestione del carico di rete (attuabile con l'introduzione di router in grado di effettuarne il bilanciamento).

3.2.3. Cluster per l'alta disponibilità dei servizi

Se il modello di affidabilità risponde alla domanda di scalabilità, quello di alta disponibilità risponde alla domanda di continuità di servizio. Un cluster per l'alta disponibilità (HA cluster) è costituito essenzialmente da due (o più) server gemelli dove meccanismi hardware e software permettono di far sì che se il nodo

principale del cluster HA si blocca, il nodo secondario, fino a quel momento in attesa, si riconfiguri per apparire come il nodo principale del cluster, con un'immagine congrua al nodo originale immediatamente prima della sua indisponibilità (ovvero senza perdite di dati sensibili).

All'utente finale, attraverso complessi algoritmi per la determinazione delle variazioni di stato dei sistemi, l'intero processo di riconfigurazione apparirà come una breve indisponibilità temporanea del servizio.

Tale modello è particolarmente indicato per tipologie di servizi come File Server o Database Server.

3.2.4. Cluster per il bilanciamento del carico

Un load-balancing cluster può essere un incomparabile strumento di produttività.

L'idea (che estende il modello di affidabilità) è quella di introdurre un sottosistema di schedulazione delle richieste applicative, come ad esempio un sistema di code che sia in grado di reindirizzare la richiesta sul nodo più scarico, in grado di far fronte alla richiesta utente. I sottosistemi di questo tipo solitamente sono dotati di vari strumenti amministrativi che permettono un controllo molto fine sull'utilizzo delle risorse e che consentono un monitoraggio avanzato dello stato del cluster.

L'asintoto attuale di questo modello è rappresentato dalla possibilità di includere a livello kernel i meccanismi di schedulazione, la gestione dello spazio globale delle risorse (file system e processi) e la gestione delle politiche di bilanciamento del carico e della migrazione dei processi.

3.2.5. Cluster per il calcolo parallelo

La ragione principale per cui molti sforzi si sono concentrati verso lo sviluppo di architetture cluster per il calcolo parallelo basate su PC sono i costi. In questo modo, invece di rincorrere a colpi di svariati milioni di dollari l'ultimo supercomputer in grado di far fronte alle sempre maggiori richieste di TeraFlops (trilioni di operazioni in virgola mobile per secondo), si è sempre più affermata l'idea di assemblare grandi quantità di

processori classe PC con una struttura di comunicazione a banda alta e bassa latenza, appositamente progettata.

Per mantenere tali caratteristiche a volte viene utilizzato un protocollo di rete diverso dal TCP/IP (come Myrinet), che contiene troppo overhead rispetto alle limitate esigenze di indirizzamento, routing e controllo nell'ambito di una rete in cui i nodi siano ben noti a priori.

In alcuni casi viene utilizzato un meccanismo di Direct Memory Access (DMA) fra i nodi, fornendo una sorta di distribuited shared memory che può essere acceduta direttamente da ogni processore su ogni nodo.

Inoltre, è previsto un livello di comunicazione a scambio di messaggi (message passing) per la sincronizzazione dei nodi , le cui implementazioni più diffuse sono rappresentate da MPI (Message Passing Interface) e PVM (Parallel Virtual Machine). MPI è una API (Application Program Interface) per gli sviluppatori di programmi paralleli che garantisce una piena astrazione dall'hardware correntemente utilizzato senza alcuna necessità di inserire nel codice del programma alcuna direttiva di effettiva distribuzione dei segmenti di codice fra i nodi del cluster garantendo, fra l'altro, una buona portabilità del codice prodotto.

PVM permette ad un insieme di computer di essere visto come una singola macchina parallela ; la Parallel Virtual Machine è composta da due entità principali: il processo PVM daemon su ogni processore e l'insieme delle routine che ne forniscono il controllo.

3.3. Problematiche inerenti ai cluster

Le problematiche che bisogna affrontare nell'utilizzo e nella gestione dei cluster sono molteplici e abbracciano gran parte degli aspetti inerenti ai sistemi distribuiti.

Nel presente lavoro di tesi non ci si pone l'obiettivo di dare una risposta a questi problemi (attraverso una dettagliata trattazione delle tematiche a riguardo), ciononostante è utile in questo frangente dare una breve panoramica di questi problemi.

3.3.1. I processi nei sistemi cluster

Una tematica centrale è sicuramente costituita dalla gestione dei processi nei cluster. Il concetto di processo proviene dall'ambito dei sistemi operativi, dove è generalmente definito come " un programma in esecuzione " (15). Dal punto di vista dei sistemi operativi, le questioni più importanti di cui discutere sono probabilmente la gestione e lo scheduling dei processi. Tuttavia, parlando di sistemi cluster, anche altre questioni si rivelano ugualmente o anche più importanti. Per esempio, per organizzare efficacemente un sistema cluster , è spesso conveniente utilizzare tecniche di multithreading. Uno dei maggiori contributi dei thread nei sistemi distribuiti è di permettere ai client e ai server di essere costruiti in modo tale che la comunicazione e l'elaborazione locale siano sovrapponibili, ottenendo così un alto livello di prestazioni. I processi giocano un ruolo importante nell'ambito dei cluster, in quanto costituiscono la base per la comunicazione tra macchine distinte. Una questione importante riguarda l'organizzazione interna dei processi e, in particolare, il loro eventuale supporto, come appena detto, di molteplici thread di controllo.

Un elemento importante, soprattutto nei sistemi distribuiti globali, è lo spostamento dei processi tra macchine diverse. La migrazione dei processi o, più specificatamente, la migrazione del codice può aiutare a ottenere la scalabilità, ma può anche agevolare la configurazione dinamica delle entità client e server del cluster. La migrazione del codice comporta problemi relativi all'uso delle risorse locali, per risolvere i quali vengono migrate anche le risorse, o vengono stabiliti nuovi collegamenti alle risorse locali della macchina target, o vengono utilizzati per le risorse locali dei riferimenti di rete globali.

La comunicazione tra processi è il cuore di tutti i sistemi distribuiti. Non ha senso studiare i sistemi cluster senza esaminare attentamente i modi in cui processi, su macchine diverse possono scambiarsi informazioni. La comunicazione in questo contesto si basa soprattutto sullo scambio di messaggi a basso livello come fornito dalla rete sottostante. Realizzare la comunicazione attraverso lo scambio di messaggi è più difficile che realizzarla usando le primitive basate sulla memoria condivisa, disponibili nelle piattaforme non distribuite.

Ci sono diversi modelli di comunicazione molto diffusi, tra i quali uno dei più conosciuti ed implementati è *RPC - Remote Procedure Call -* , la cui essenza è che un servizio è implementato per mezzo di una procedura

il cui corpo è eseguito su una macchina server, invocabile da una macchina client (due nodi differenti di uno o più cluster all'interno della stessa o di differenti grid).

3.3.2. Middleware per la gestione dei processi

In questo contesto, avendo capito che la gestione dei task è una questione critica, è fondamentale la presenza di middleware preposto per la gestione degli stessi task sui sistemi cluster. È necessario dotare il proprio sistema di strumenti che gestiscano la distribuzione dei job sui vari nodi di calcolo, la loro comunicazione, lo scheduling tra gli stessi, il recovery delle informazioni in caso di crash del sistema. Si possono individuare, in questi termini, due entità fondamentali: il *resource manager* e lo *scheduler manager*. Il resource manager è costituito a sua volta dall'*execution daemon* e dal *batch system server.* L'execution daemon è una applicazione lato slave che comunica al server lo stato di un nodo, che fa partire jobs e che insieme al server copia dati da e per i nodi. Il *batch system server* è l'applicazione che colleziona tutte le informazioni sui nodi, che si occupa di far iniziare i jobs, comunicando con gli execution daemons , e che gestisce il data transfer con i nodi. Lo scheduler è l'applicazione che sulla base di vari parametri (stato dei nodi, politiche di assegnazione delle risorse, ecc) decide la distribuzione dei job nei nodi.
Tipicamente i middleware di comunicazione rientrano in uno di due principali paradigmi:

- La comunicazione *a memoria condivisa*, nella quale si suppone che i processi coinvolti abbiano uno spazio di memoria comune mediante il quale possono comunicare tra loro.
- La comunicazione mediante *scambio di messaggi*, che prevede un sistema di messaggistica che i processi possono utilizzare per inviare e ricevere informazioni.

Questi paradigmi si riferiscono al modello presentato al programmatore, non alla tecnica di implementazione. Se infatti è possibile e naturale realizzare il paradigma a memoria condivisa dove ci siano degli effettivi banchi di memoria condivisi tra i processi (e.g. nel caso SMP), questa non è l'unica possibilità: è possibile simulare la condivisione della memoria attraverso lo scambio di messaggi (si vedano i vari

sistemi di DSM- Distributed Shared Memory). Viceversa, è possibile utilizzare un paradigma a scambio di messaggi non solo su dispositivi orientati a questa funzione come le interfacce di rete, ma anche su una implementazione che non faccia altro che copiare i “ messaggi ” in banchi di memoria condivisa.

Esempi di middleware che seguono il modello della memoria condivisa sono OpenMP (18) e Unified Paralled C (19).

Esempi di middleware basati sul message passing includono PVM ed MPI, descritto di seguito

3.3.3. Il middleware MPI

La specifica MPI nasce nel 1992 da uno sforzo di diverse organizzazioni scientifiche ed industriali verso la standardizzazione di un’interfaccia per il message passing: gli anni ’80, infatti, avevano visto proliferare molte librerie che offrivano tale funzionalità, sviluppate indipendentemente e quindi tutte mutuamente incompatibili. MPI comunque non ha mai ricevuto il supporto di enti di standardizzazione internazionale e si è affermato dunque come standard de facto. La prima specifica, comunemente nota come MPI-1 è del 1994, la seconda, MPI-2, è del 1997. Esistono differenti implementazioni sia della prima che della seconda specifica: le più diffuse nell’ambito dei sistemi cluster sono MPICH e OpenMPI (oltre al predecessore di quest’ultima, LAM/MPI).

In base alla nota tassonomia di Flynn (20), possiamo definire il modello di esecuzione MPI come “ MIND su SPMD ” : se infatti viene lanciato in generale un solo eseguibile (anche se il supporto per l’MPMD è presente), i singoli processi in esecuzione sui vari processori possono seguire percorsi di computazione diversi, ad esempio in base alla loro identità, costituita dal numero identificativo, o rank , che ricevono partecipando alla computazione MPI.

I processi che partecipano all’esecuzione di un programma MPI comunicano tra loro attraverso primitive di invio e ricezione di messaggi. Essi vivono all’interno dei communicators, cioè contenitori di processi , che identificano i gruppi di processi all’interno dei quali può avvenire la comunicazione. All’inizio della

computazione tutti i processi appartengono al communicator MPI:COMM:WORLD, poi la specifica applicazione può definire nuovi communicator e membership dei processi.

Le primitive di comunicazione di base sono MPI_Send, per spedire un messaggio al processo, e la MPI_receive: tuttavia, se pure è possibile costruire molti programmi con queste due solo primitive, esse costituiscono solo una minima parte della funzionalità offerte da MPI, tra cui ricordiamo le versioni non – bloccanti delle operazioni e le funzioni di comunicazione collettiva.

MPI2 estende MPI1 in questi ambiti funzionali nei quali la prima specifica mostrava delle lacune, che includono:

- Interpolarità tra le implementazioni in diversi linguaggi (presenza di wrappers).
- Possibilità di gestire dinamicamente il numero di processi (spawinf di nuovi processi);
- Comunicazioni single-sided (creazione di una finestra di memoria condivisa nel quale depositare e prelevare dati);
- I/O parallelo.

3.4. Introduzione a Rocks

Come già affermato nei paragrafi precedenti, la realizzazione di cluster di macchine economiche sono un modo conveniente per raggiungere elevati livelli di potenza computazionale. Ma se una strategia di cluster management non è adottata, i benefici economici legati al cluster sono persi in quanto diventerebbe necessario un personale dedicato al controllo del corretto funzionamento delle macchine.

La complessità del cluster management (ad esempio per il controllo che tutti i nodi hanno un insieme di software consistente) occupa spesso più tempo del previsto. Quando ciò accade è inevitabile il verificarsi o dell'instabilità del cluster, o di una falla nel sistema non controllata.

Mentre i primi toolkit per la gestione di un cluster basavano la loro forza su software per il controllo della coerenza dei file di configurazione dei nodi, Rocks (21) rende l'installazione del sistema operativo su di un nodo lo strumento principale per la gestione del cluster. Inoltre è stata posta attenzione alla automazione completa del processo d'installazione dei nodi. Può sembrare una perdita di tempo dover reinstallare un nodo a fronte di una semplice modifica di un file di configurazione, ma questa filosofia è molto scalabile.

Uno degli ingredienti di Rocks è un meccanismo robusto per produrre distribuzioni personalizzate (con patch di sicurezza pre-applicate) che definisce l'insieme completo di software per un nodo particolare. Un cluster può richiedere tipi differenti di macchine, con software specializzato, ad esempio un frontend, file server per frontend, nodi di calcolo, e nodi di monitoraggio.

Ogni nodo viene installato mediante un file specializzato che riporta tutte le informazioni necessarie per l'installazione. Tale file si chiama *file di Kickstart*, ed è generato mediante un altro file XML chiamato *grafo di Kickstart*.

Mediante l'utilizzo di grafi, Rocks può efficientemente definire nuovi tipi di nodi ereditando le caratteristiche dei componenti preesistenti.

Quindi buona parte del software è comune, mentre vengono descritte solo le differenze.

Questa tecnologia con cui si realizzano le installazioni è particolarmente comoda, in quanto permette di adattarsi all'hardware della macchina su cui avviene (ad esempio, tipi di sottosistemi per dischi: SCSI, IDE, dispositivi RAID integrati, interfacce Ethernet, interfacce di rete ad alta velocità).

Rocks utilizza un meccanismo basato su un database per memorizzare le informazioni di interesse globale. Tali informazioni vengono poi utilizzate per la creazione automatica dei file di configurazione specifici per alcuni servizi (ad esempio file di configurazione del DHCP, /etc/hosts, e i file di nodo PBS).

3.4.1. Panoramica della distribuzione

Rocks è una distribuzione Linux per cluster basato sulla distribuzione Red Hat con pacchetti aggiuntivi e configurazioni programmate per automatizzare la messa in opera di clusters Linux ad elevate prestazioni. La

distribuzione ha origine dal progetto NPACI (National Partnership for Advanced Computional Infrastructure) terminato nel 2004 ma tutt'oggi sviluppato ed aggiornato (la prima release è del 2000, mentre l'ultima è del luglio 2009 ed è arrivata alla versione 5.2).

Rocks è messo in opera sulla base dell'architettura tradizionale basilare per i cluster come mostrato nella figura in basso.

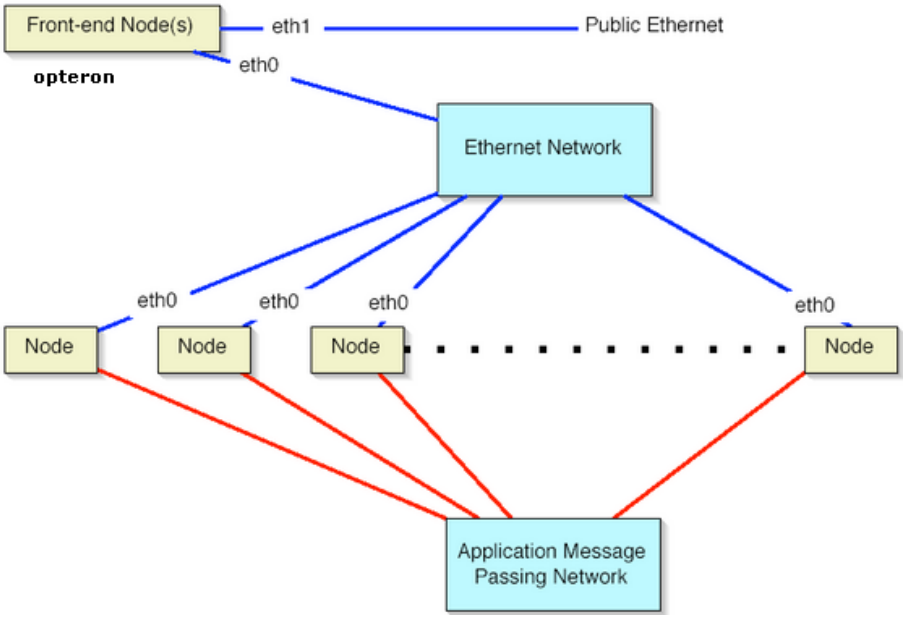


Figura 3.1 – Architettura HW minima di Rocks

Questo sistema è composto da server standard, una rete ethernet, e una rete opzionale ad elevate prestazioni per connettere i nodi di calcolo (ad esempio Myrinet). Questa architettura minimale ha il pregio di essere affidabile, in quanto i componenti utilizzati sono i fondamentali, e quindi la probabilità di rottura è minima.

L'installazione di un cluster Rocks unifica due processi ortogonali: l'installazione dei pacchetti software e la loro configurazione.

L'approccio tradizionale a singola macchina prevede l'installazione del software e la sua configurazione mediante l'inserimento dei dati manualmente sulla base delle proprie necessità. Una modifica a questo approccio può essere quella di configurare un singolo nodo a mano, e poi replicare questa immagine

dell'installazione sulle altre macchine. In realtà Rocks non replica l'immagine dell'installazione, le opzioni di configurazione infatti vengono determinate mediante degli script, una volta determinate le specifiche di configurazione allora esse vengono applicate.

Usare immagini di installazioni realizzate da altre persone non ha molti vantaggi, implicherebbe un numero elevato di immagini e non una procedura unica e valida per tutti i clusters. In Rocks l'installazione e la configurazione viene realizzata mediante l'installazione di pacchetti RPM. Una volta realizzata l'installazione di una macchina sulla base del suo ruolo funzionale, tale macchina prende il nome di *appliance*.

I cluster Rocks spesso contengono appliance di tipo: *Frontend* , *Compute* e *NFS* .

Un semplice grafo di configurazione (espresso in XML) permette ai cluster architects di definire nuovi tipi di appliance ed utilizzare i grandi vantaggi del riuso del codice per l'installazione e configurazione del software.

Lo scopo principale di Rocks è permettere a persone non esperte di cluster di costruire facilmente e gestire il proprio cluster. Per raggiungere questo risultato, in Rocks sono presenti due elementi di progetto chiave:

- Automazione delle funzioni amministrative.
- Ripetibilità.

La ripetibilità assicura che le modifiche apportate nel nostro laboratorio valgono anche in un altro. Il primo punto è critico per le installazioni di nodi completamente automatizzate, ma una proprietà importante al crescere della dimensione del cluster.

Il secondo punto è affrontato mediante il meccanismo chiamato Roll. Infatti mediante un insieme di Roll è possibile installare e configurare in modo automatico un particolare tipo di cluster.

3.4.2. Roll

Un roll è una immagine ISO che immagazzina package e file di configurazione da aggiungere al grafo di configurazione completo.

Tramite un roll è quindi possibile estendere le possibilità del proprio cluster, introducendo un nuovo sottografo , nuovi nodi radice (Appliances) e, dinamicamente, nuovi pacchetti software.

Attualmente in rete sono presenti diversi roll installabili sul proprio cluster Rocks. Sia dal mondo opensource, sia dagli ambiti commerciali sono offerte diverse funzionalità e tools la cui installazione sul proprio sistema è facilitata dai meccanismi di personalizzazione di Rocks e dei roll.

In questo paragrafo sarà fornita una panoramica dei maggiori roll attualmente disponibili.

Uno tra i roll più importanti è il *Grid roll*. Esso è presente sul sito ufficiale di Rocks [12] ed è completamente basato su Globus Toolkit 4. L'installazione di questo roll permette al cluster di accedere a tutte le funzionalità offerte da Globus, come i servizi di sicurezza (gestione di autenticazione , autorizzazione , meccanismi di delega), di data management (GridFTP, Reliable File Transfer, data replication ecc), di controllo e scoperta delle risorse e dei servizi grid e di execution management, cioè di inizializzazione, controllo , scheduling di computazioni remote.

Un ulteriore roll di rilevante importanza è il *Condor roll.* Esso è basato sull'ultima versione di Condor per offrire meccanismi di job management (job submission e job al location), politica di schedulino, schema con priorità, monitoring di risorse ecc.. Attraverso l'utilizzo di questo roll è possibile realizzare un ambiente HTP (High Throughput Computing), cioè in grado di gestire grandi quantità di potenza di computazione fault tolerant.

Degno di nota è sicuramente l'*SGE (Sun Grid Engine) roll* . Questo roll mette a disposizione della macchina su cui viene installato SGE. SGE viene installato in modo tale che NFS non sia richiesto, guadagnando in tal modo in scalabilità in termini di setup, senza perdere i benefici che si hanno unendo SGE con NFS. Una funzionalità molto interessante che viene offerta dall'SGE roll consiste nel fatto che le code generiche per i jobs vengono installate automaticamente quando nuovi nodi vengono installati nel cluster e avviati.

Sempre nell'ambito del monitoraggio dei sistemi è da segnalare il *Ganglia roll.* Questo roll installa sul cluster Rocks il sistema distribuito di monitoraggio Ganglia. Viene quindi offerta la possibilità di rappresentare i dati tramite strumenti basati su XML, per i data transports vengono offerti strumenti per XDR ecc..

Altri roll presenti in rete sono:

- *Area51 roll* : contiene utilità e servizi per controllare l'integrità dei files e del kernel presenti sul cluster ;
- *Bio roll* : che offre i più aggiornati ed utilizzati tools per la bio-informatica, cioè l'utilizzo di tecniche per la matematica, informatica, statistica applicate per risolvere problemi di carattere biologico ;
- *APBS roll* : che mette a disposizione un risolutore di equazioni di Poisson-Boltzmann usate nello studio delle proprietà elettrostatiche di sistemi biomolecolari;
- *PVFS2 roll* : che installa packages , scripts e tools per supportare gli utenti a fare il deploy di un'istanza di PVFS2, il file system distribuito per cluster che nella sua ultima versione offre piena compatibilità con Rocks ;
- *Moab roll* : offre un middleware di gestione intelligente per la gestione Web-based dei jobs, strumenti grafici per l'amministrazione del cluster e tools per il reporting del management ;
- *Pbs roll* : offre gli strumenti di resource manager e scheduler per il job management del cluster installando Torque e Maui sul sistema ;
- *Pgi roll* : è un roll che installa sul cluster i compilatori multicore ottimizzanti PGI (Fortran, C e C++).

3.5. Virtual Cluster

L'unione del concetto di cluster e quello di virtualizzazione danno vita al cluster virtuale. Ci sono molti potenziali vantaggi nell'uso della virtualizzazione negli ambienti cluster, uno di queste può essere dato dal

fatto che un utente può definire il proprio ambiente operativo, in modo del tutto indipendente agli altri ambienti di esecuzione.

In pratica un cluster virtuale non è molto diverso da un cluster fisico, entrambi hanno bisogno sia di un front-end che dei nodi computazionali, però nel caso di un cluster virtuale sia i nodi che il front-end sono virtualizzati così come mostrato nella figura sottostante .

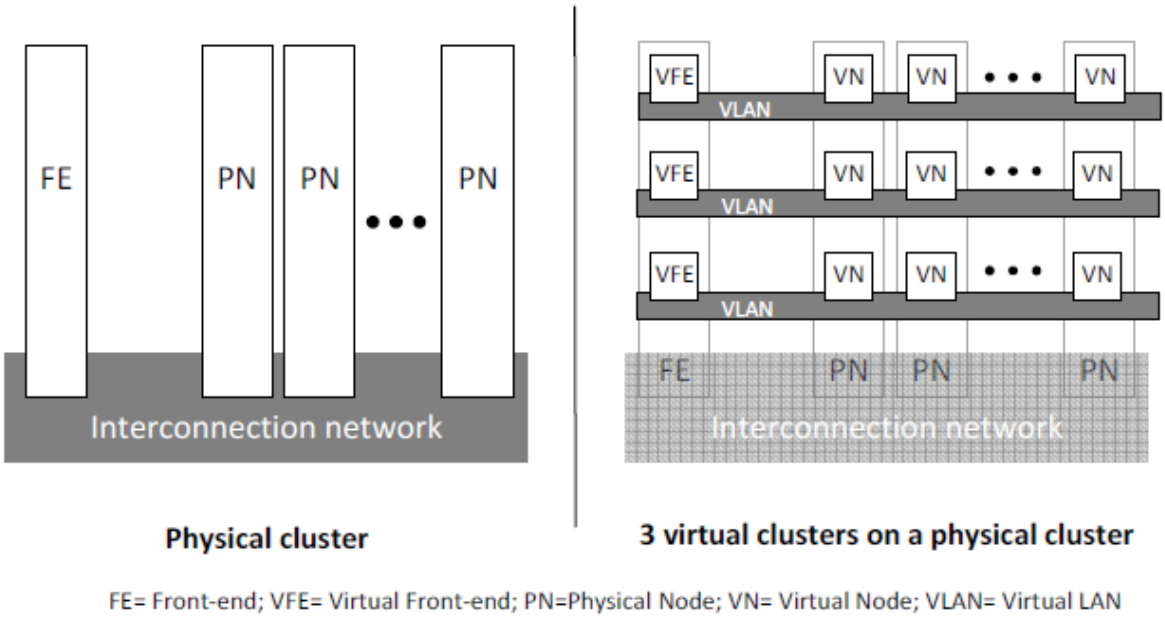


Figura 3.2 – Schema riassuntivo cluster fisico e cluster virtuale

Anche se nella teoria il front-end di un cluster virtuale è virtualizzato, nella pratica molto spesso si virtualizzano solo i nodi lasciando il front-end nativo del proprio Sistema Operativo senza compromettere in questo modo l'ambiente di esecuzione dei nodi virtualizzati (sistema operativo, librerie, tolls ecc..).

Come mostrato nella figura 3.2, ogni nodo computazionale più avere uno o più macchine virtuali, ognuna delle quali esegue una istanza privata del sistema operativo. Le VM di uno stesso nodo possono o non possono appartenere allo stesso cluster virtuale ed il numero massimo delle VM è limitato solo dal consumo delle risorse fisiche.

Ogni macchina virtuale può avere una o più CPU virtuali (VCPU) , ed ogni VCPU può essere associata ad una o più CPU fisiche del nodo di cui fa parte. Nei cluster dove sono settate le affinità dei processori virtuali *v*

con un numero minore di CPU fisiche *n,* dove quindi vale la relazione $v > n$, oltre alla condivisione delle risorse quali: memoria, interfacce di rete, ci sarà anche la condivisione dei cicli della CPU, con un effetto negativo diretto sulle prestazioni di tutti gli ambienti che utilizzano la stesa CPU.

A partire dalla release 5.0 dell'aprile 2008, la distribuzione Rocks mette a disposizione delle utility per installare velocemente dei cluster virtuali: in alternativa a dei nodi *compute ,* permette di installare dei nodi contenitori *vm-container,* capaci di ospitare macchine virtuali. La tecnica utilizzata da Rocks per la virtualizzazione è quella della paravirtualizzazione, ed usa come hypervisor Xen.

3.6. Riepilogo

In questo capitolo si sono introdotti i concetti fondamentali alla base del lavoro svolto. Si è fatta una panoramica sui sistemi cluster e sulle varie tipologie adottate, sui middleware utilizzati per l'esecuzione parallela dei processi, sul Sistema Operativo per cluster Rocks e sull'uso dei roll, nonché una parte relativa alla virtualizzazione dei cluster. Infine si è fatta una breve panoramica sull'organizzazione strutturale di un cluster virtuale e sull'utilizzo delle risorse fisiche condivise.

CAPITOLO 4

4. Analisi delle prestazioni

Negli ultimi anni, vi è una continua crescita del trend che porta a lanciare applicazioni di rete intensive, per esempio un server hosting web, in una macchina virtuale (VM). Spesso molte VM girano sullo stesso hardware fisico, condividendo le risorse di rete, la cpu ecc. Per tale motivo, capire il deterioramento delle prestazioni di rete è un aspetto fondamentale per l'ottimizzazione dei sistemi virtuali.

In questo capitolo vengono dapprima analizzate le prestazioni di alcune caratteristiche di rete delle VMs paravirtualizzate tramite Xen, ed infine viene monitorato l'intero sistema tramite un profiler per verificare l'effettivo overhead introdotto dalla comunicazione di rete tra le interazioni che intercorrono tra la sottostante VMM e le VMs presenti. I risultati dell'ambiente di test si aiuteranno a capire in che modo si deteriorano le prestazione di rete in base alle diverse accoppiate VCPU – CPU, ed avere una statistica sui processi in esecuzione, in modo tale da capire quanto può influire una data applicazione nell' overhead delle performance per I/O di rete .

4.1. Piattaforma Hardware

Gli esperimenti sono stati effettuati sul Cluster denominato Fab4, collocato presso il laboratorio Para.di.se dell'Università degli studi del Sannio.

Il cluster è composto da 4 nodi ed un front-end, ciascun nodo è una workstation HP wx6000 con le seguenti caratteristiche:

- Due processori Intel Xeon 2,80 GHz con 512 KB di cache di secondo livello ;

- Chipset Intel E7505 ;
- 1,5 GB di memoria RAM ;
- Scheda di rete Broadcom Corp. NetXtreme BCM5702 Gigabit Ethernet ;
- Disco rigido Seagate ST336607LW SCSI Ultra320 da 36 GB ;

Il front end è un personal computer Fujitsu-Siemens Esprimo P5905 con le seguenti caratteristiche:

- Due processori Intel Pentium4 3,0 GHz con 20148 KB di cache di secondo livello ;
- Chipset Intel 82945G ;
- 1 GB di memoria ;
- Due schede di rete, una Realtek RTL-8139 e una Broadcom Corp. NetXtreme BCM5751 Gigabit Ethernet ;
- Un disco rigido SATA Samsung HD160JJ da 160 GB e un disco SATA Western Digital WD2500JS-55W da 250 GB ;

I nodi sono interconnessi tra loro da uno switch 3COM Gigabit Ethernet con 8 ingressi LAN.

4.2. Piattaforma Software

Sul cluster è stata installata la distribuzione Rocks 5.1 per architettura i386 . I *rolls* selezionati per l'installazione sono:

- *base*, per i pacchetti base ;
- *Area51*, per i servizi e le utilità di sicurezza ;
- *Ganglia*, per il monitoraggio del cluster ;
- *Hpc*, per le utilità HPC (in particolare Open MPI) ;
- *Java*, per la JVM e lo SDK Java di Sun ;

- *Kernel*, per il kernel Linux avviabile ;
- *Os*, per il sistema oparativo CentOS ;
- *Sge*, per il Sun Grid Engine ;
- *Web-server*, per il server web ;
- *Xen*, per il supporto alla virtualizzazione ;
- *Bio*, per le utilità di bioinformatica ;

Il sistema operativo usato da Rocks è CentOS 5 . Il kernel del front-end è il 2.6.18-92.1.22.el5 SMP, mentre quello usato dai nodi per la virtualizzazione (sia dal Dom0 che dai DomU) è la versione precompilata 2.6.18-92.1.13.el5xen. La versione dell'hypervisor Xen è la 3.03-64.el5 . La versione del gcc usato è la 4.1.2-42.el5 . L'implementazione MPI2 utilizzata è quella Open MPI , nella versione 1.2.5-2-el5

4.3. Tool di analisi

In questa sezione ci sarà una breve descrizione dei tool utilizzati per il monitoraggio e l'analisi delle prestazione. Il primo tool trattato sarà Netperf (22), utilizzato per calcolare il throughput di rete e l'utilizzo della CPU del *sender/receiver* . Pathrate e Pathload (23) usati per calcolare rispettivamente la capacità del path e la banda disponibile, e Xenoprof (24) il quale deriva direttamente dal profiling Oprofile (25) ed utilizzato per il monitoraggio attivo delle varie macchine virtuali.

4.3.1. Netperf

Netperf è un benchmark usato per misurare vari aspetti delle performance di rete. L'obiettivo principale è quello di spedire una grande quantità di dati attraverso il protocollo TCP o UDP e misurare le prestazioni in termini di bitrate massimo tra due host.

E' stato sviluppato prendendo come esempio il modello classico client-server. Ci sono due file eseguibili:

- **Netserver,** lanciato sul server
- **Netperf,** lanciato sul client

Netserver è il primo eseguibile che va lanciato, in quanto è il processo che resta in ascolto sul server in attesa delle ricezione dei pacchetti. Netperf è l'eseguibile da lanciare sul client, il quale appena lanciato stabilisce una connessione di controllo con il sistema remoto. Questa connessione sarà usata per passare le informazioni sulla configurazione dei test ed i risultati da e verso il sistema remoto. Indipendentemente dal tipo di test da effettuare, questa connessione di controllo sarà di tipo TCP usando le socket BSD.

Una volta effettuata la connessione di controllo e passati i parametri di configurazione, si aprirà una nuova connessione che avrà il compito di effettuare le misurazioni usando le API ed i protocolli appropriate per il test scelto. Netperf non effettuerà nessuno scambio di pacchetti sulla connessione di controllo finché il test non avrà fine.

Una informazione molto importante che mette a disposizione Netpef è l'utilizzazione della CPU sia da parte del sender che del receiver, anche se l'accuratezza di tale dato spesso non corrisponde alla realtà.

Esistono molti test che può effettuare Netperf:

- TCP_STREAM, è il test di default ;
- TCP_RR, consente di verificare le prestazioni di più coppie di pacchetti richiesta/risposta. Simula il comportamento di un'applicazione database;
- TCP_CRR, per ciascuna coppia richiesta/risposta crea una nuova connessione ;
- UDP_STREAM, simile al TCP_STREAM ma con l'eccezione di usare pacchetti UDP ;

- UDP_RR, stessa tipologia di TCP_RR ma con pacchetti UDP ;

Opzioni Globali

-l testlen	Setta la durata in secondi del test
-t testname	Specifica il tipo di test da effettuare
-H host	Specifica Hostname o l'indirizzo IP del server
-c rate	Setta il rate di utilizzazione della CPU locale, se non è specificato Netperf provvede a calcolarlo
-C rate	Setta il rate di utilizzazione della CPU remota, se non è specificato Netperf provvede a calcolarlo

Specifiche per il test

-s size	Configura la dimensione del buffer della socket locale
-S size	Configura la dimensione del buffer della socket remota
-m size	Configura la dimensione del messaggio locale
-M size	Configura la dimensione del messaggio remoto
-r req , resp	Configura la dimensione dei messaggi di richiesta/risposta

4.3.2. Pathrate

Pathrate è un tool utilizzato per calcolare la capacità massima di un path di rete sfruttando la tecnica del Packet Train Dispersion. E' un tool che colleziona molte misurazioni di coppie di pacchetti, utilizzando varie dimensioni. Analizzando la distribuzione dei dati si riscontrano vari "modi di capacità", una delle quali rappresenta il valore della capacità del percorso. Successivamente Pathrate utilizza lunghi treni di pacchetti per calcolare l'ADR (Asymptotic Dispersion Rate) e, considerando che questo valore è sempre inferiore alla Capacità reale, per confronto con i risultati ottenuti in una prima fase di analisi con le coppie di pacchetti, si risale alla stima della Capacità del link, come mostrato nella figura 4.1.

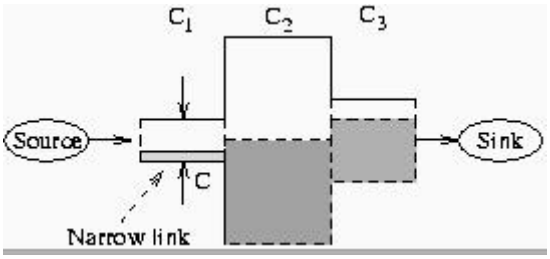


Figura 4.1 – Esempio del calcolo della capacità del path (narrow link)

4.3.3. Pathload

Pathload è un tool usato per stimare la larghezza di banda disponibile tra un collegamento end-to-end tra un host sender ed uno receiver. L'algoritmo di stima usato da Pathload è basato sulla metodologia SLoPS e l'idea alla base del funzionamento è la seguente: quando un processo manda uno stream periodico di pacchetti UDP ad un ritmo più alto rispetto alla banda disponibile nel path, il ritardo dei pacchetti tende a salire. Al contrario, quando il rate di trasmissione è minore rispetto alla banda disponibile non si ha l'incremento del ritardo dei pacchetti. In questo modo possiamo avere una stima molto precisa della massima banda disponibile per quel collegamento.

Pathload è formato da un processo S che funge da sender, ed un processo R che ha la funzione di receiver. S manda N stream periodici di pacchetti UDP, ed R controlla se vi è un aumento del ritardo per ogni singolo stream trasmesso, in questo modo si può capire con esattezza la massima banda disponibile, la figura 4.2 ci mostra schematicamente che cosa si intende per larghezza di banda di un canale.

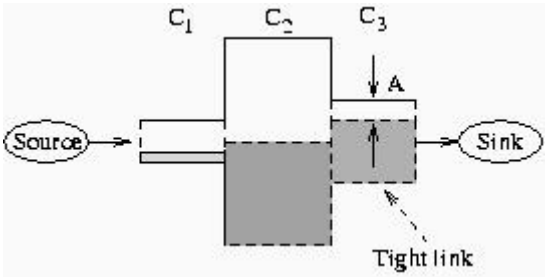


Figura 4.2 – Esempio di banda disponibile in un paht, data dal link con la minima capacità non usata (Tight link)

4.3.4. OProfile

E' un tool statistico di profiling per sistemi Linux. OProfile può essere usato per effettuare il controllo delle prestazioni o l'analisi del codice eseguito ad ogni livello di privilegio: codice a livello kernel, moduli del kernel, applicazioni e librerie a livello user. Consiste in un driver a livello kernel ed un demone per raccogliere i campioni di dati, ed in una serie di tool post-profiling per interpretare i dati e fornire le informazioni necessarie.

OProfile riesce a collezionare molte informazioni sui moderni processori, grazie anche ad un hardware di controllo che quest'ultimi hanno a disposizione, tra cui i cicli di clock, TLB misses, cache misses, branch mispredictions, etc. L' hardware di solito viene rappresentato da uno o più contatori, che vengono incrementati ogni qualvolta si verifica un evento, in questo modo possiamo sapere quando un contatore ha raggiunto un certo valore. Per esempio, OProfile può raccogliere il valore del program counter (PC) ogni volta che i cicli di clock raggiungono un determinato valore, in modo da avere una statistica sul tempo

speso dalle varie routines nel sistema . Allo stesso modo, OProfile memorizza il valore del PC ogni volta che si è raggiungo un certo numero ti TLB miss, in modo da avere un'idea di come si distribuiscono i TLB miss tra le varie routines. L'accuratezza, quindi l'overhead introdotto dal profiler, può essere controllata settando gli intervalli dei campioni. I contatori possono essere caricati con un valore iniziale e produrre una interruzione ogni volta che si presenta un overflow, rendendo così possibile un controllo accurato della quantità dei dettagli, maggiore è il numero dei campioni raccolti, maggiore è l'overhead prodotto dal profiling.

OProfile usa questo hardware di controllo, oppure un sostituto basato sul timer in casi dove non è presente un hardware di controllo della prestazione, per effettuare un'analisi minuziosa sul comportamento dell'intero sistema. I campioni, una volta raccolti, sono registrati periodicamente su di un disco; più in avanti i dati contenuti in questi campioni, possono essere usati per generare dei report sulla prestazione a livello del sistema e delle applicazioni.

Il controllo del sistema con OProfile opera nel seguente modo (semplificato) :

1. L'utente specifica a OProfile quali sono gli eventi da monitorare ed il valore periodico di controllo ;
2. L'hardware di controllo viene programmato da OProfile in modo da conteggiare gli eventi specificati dall'user ed alzare una interruzione ogni volta che il contatore ha raggiungo il valore specificato ;
3. L'hardware di controllo emette il segnale di interruzione quando si raggiunge il valore specificati, e tocca a OProfile gestire tale segnale e memorizza il valore del contatore nel kernel buffer ;
4. Oprofile processa periodicamente il buffer di memoria del kernel per determinare le routines ed i corrispondenti eseguibili per ogni valore contenuto nel buffer. Tutto ciò è effettuato consultando la configurazione della memoria virtuale dei processi e del kernel.

4.3.5. Xenoprof

Xenoprof offre le stesse funzionalità di OProfile ma per gli ambienti Xen. Le prestazioni delle applicazioni in un ambiente virtualizzato dipendono dall'interazione dei diversi processi, dal sistema operativo lanciato su quella macchina, la VMM di Xen, e potenzialmente le altre macchine virtuali che girano sullo stesso sistema. Per poter studiare i costi, in termini di perdita di prestazione, dovuti alla virtualizzazione abbiamo bisogno di uno strumento che mette in relazione le routines della VMM e tutte le macchine virtuali che girano su di essa.

Xenoprof è formato da due livelli applicativi, il primo è a livello VMM ed è responsabile della gestione e catalogazione dei contatori dell'hardware di controllo, il secondo è a livello di dominio dove è responsabile dell'attribuzione dei campioni catalogati alle varie applicazioni che girano sui domini.

Il problema principale del profiling delle applicazioni virtualizzate è dato dal fatto che esse non sono centralizzate in un unico ambiente di esecuzione, le informazioni richieste per un analisi dettagliata sono sparpagliate tra le varie macchine virtuali, e questo tipo di informazioni a livello di dominio spesso non sono accessibili dall'hypervisor. Per esempio, Xen non può determinare il processo in esecuzione in un dominio, o perlustrare la memoria virtuale di un dominio per assegnare i determinati campioni raccolti ad una specifica applicazione. Per tale motivo, il profiling di Xen deve essere distribuito tra i vari domani , in questo modo ogni dominio effettua il profiling del proprio ambiente e passa le proprie informazioni a Xenoprof, il quale è responsabile della coordinazioni delle varie esecuzioni dei domini e la gestione degli accessi delle macchine virtuali ai contatori hardware.

Tale tecnica di profiling viene chiamata " attiva ", in quanto i domini in prima persona sono responsabili dell'analisi del proprio ambiente, un'altra tecnica che può essere adottata è quella " passiva " dove i domini non partecipano al profiling del sistema, tale tecnica è usata nel caso si è interessati alla stima delle esecuzioni globali provenienti dai domini.

Il profiling a livello dominio non è molto dissimile a quello usato negli ambienti non virtualizzati. Per le operazioni a basso livello, come per esempio l'accesso all'hardware di controllo o la collezione dei campioni, c'è il bisogno di interfacciarci con il framework di Xenoprof. Le operazioni ad alto livello

rimangono le stesse. Dopo aver ricevuto i campioni raccolti da Xenoprof, il dominio è responsabile della corretta associazione tra i campioni ed i processi in esecuzione in quel determinato momento. In questo modo il profiling dettagliato dell'intero sistema può essere ottenuto semplicemente mettendo assieme tutte le informazioni ricevute dai vari domini.

Il framework di Xenoprof definisce una interfaccia paravirtualizzata per il supporto alle singole macchine virtuali che rispecchia i seguenti punti:

1. Definisce una interfaccia degli eventi delle prestazioni (events performance), la quale è responsabile della collezione dei contatori hardware e definisce un set di funzioni usate dai domini per specificare i parametri da monitorare, e l'inizio e la fine del profiling ;
2. Permette ai domini di coordinare il proprio profiling tramite dei registri virtuali attivati ogniqualvolta è disponibile un nuovo campione di dati ;
3. Coordina il profiling di domini multipli. In ogni sessione di profiling ci sono dei domini ben distinti, l'iniziatore, il quale è responsabile della configurazione e dell'avvio della sessione di profiling. Dopo che il dominio iniziatore ha selezionato i vari domini da monitorare, Xenoprof si mette in attesa della comunicazione dei domini monitorati del loro stato di ready. Solo dopo tale comunicazione si può avviare il profiling ;

Nella architetture correnti, solo il dominio privilegiato (DOM0) può fungere da iniziatore.

4.4. Metodologia

Le misurazioni effettuate hanno l'obiettivo primario di capire il comportamento di alcuni parametri dell'infrastruttura di rete, virtualizzata tramite Xen, in base alle diverse accoppiate VCPU-CPU. Vari studi effettuati (26) (27) hanno dimostrato che l'overhead introdotto dalla virtualizzazione della infrastruttura di

rete influisce negativamente sulle prestazioni di rete e che per le applicazioni *intensive I/O* vi è un notevole uso della CPU .

Per questo motivo l'attenzione dei test è rivolta al calcolo del Throughput massimo e la percentuale di occupazione della CPU tra i due host, seguiti da un sessione di profiling per vedere quale è la parte del codice sorgente di Xen che maggiormente utilizza risorse CPU.

Ogni nodo fisico del fab4 è configurato come " vm-container ", cioè come contenitore di macchine virtuali. Tale vm-container funge da DOM0, settato in modo da girare sempre sulla prima cpu del nodo e con 256 MB di memoria. In tale scenario vengono eseguite 2 DOMU, configurate come " compute-node ", la prima con due processori virtuali (VCPUs) e 512 MB di memoria, la seconda con la medesima quantità di memoria ma con un solo processore virtuale (VCPU).

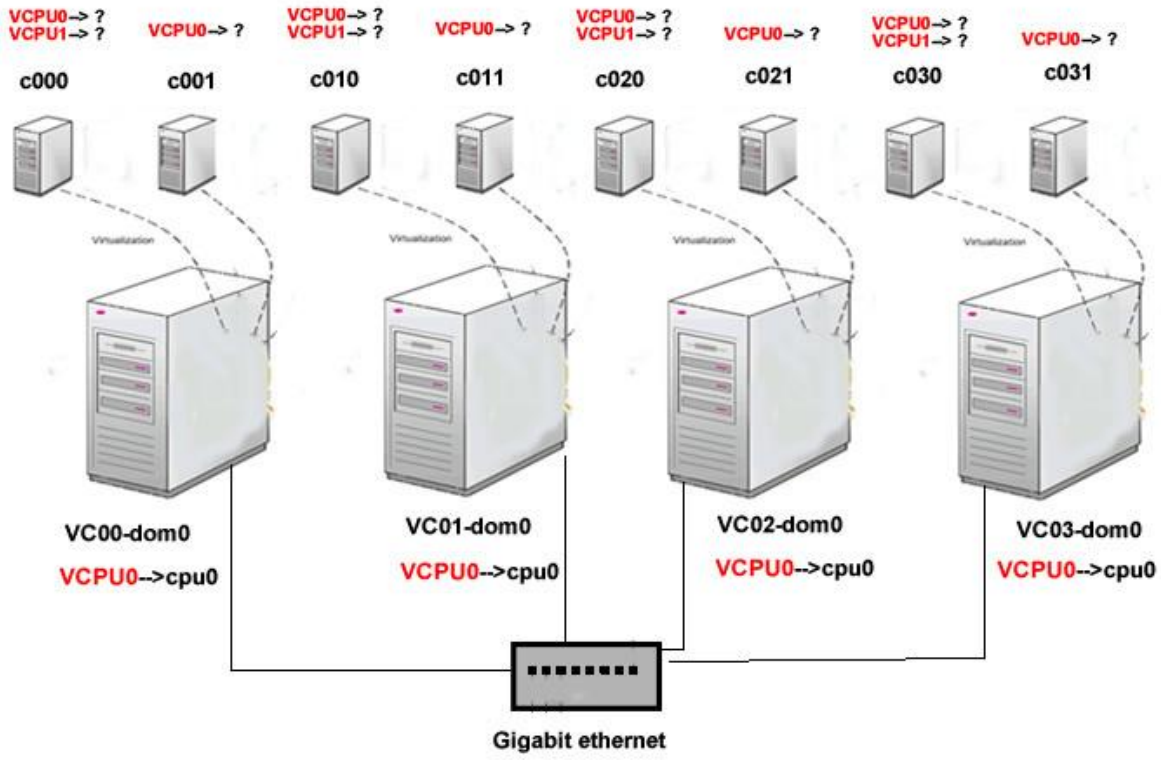


Figura 4.3 – Struttura del cluster fab4 con i nodi virtualizzati

Tra i vari tipi di test messi a disposizione da netperf si è scelto la modalità TCP_STREM, la quale simula una normale comunicazione tra un processo sender ed uno receiver di tipo TCP. Sia il buffer del receiver (87380 bytes) che del sender (16384 byte) non sono stati modificati, l'unico parametro variato è stato quello della grandezza del messaggio inviato (da 120 a 100000 bytes).

I test sono stati eseguiti secondo tre configurazioni:

Configurazione numero 1

La DOM0 gira sulla CPU0, mentre le due DOMU girano sulla CPU1

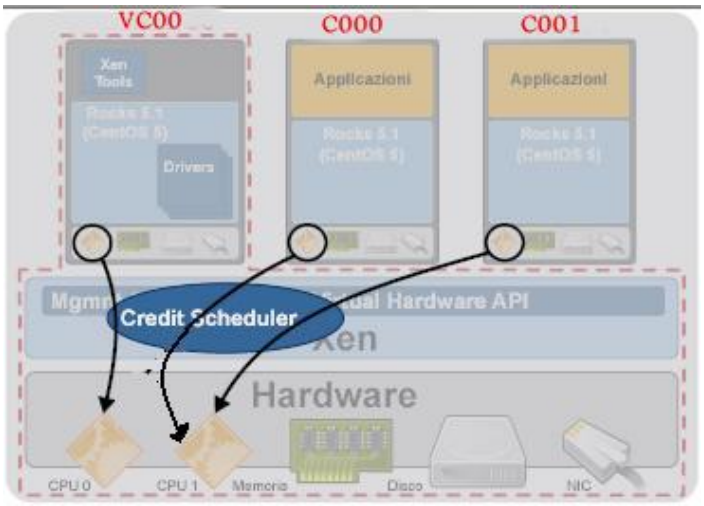


Figura 4.4 – Associazione tra VCPU di vc00 su CPU0 , VCPU domU su CPU1

Configurazione numero 2

Sia la DOM0 che le DOMU con due VCPUs girano sulla CPU0, la DOMU con una sola CPU gira sulla CPU1

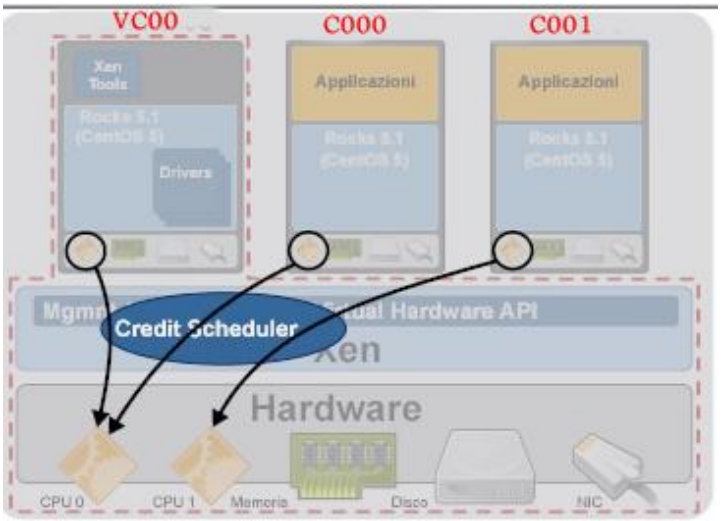


Figura 4.5 – Associazione tra VCPU del vc00 e c000 sulla CPU0 , c001 sulla CPU1

Configurazione numero 3

Sia la DOM0 che le due DOMU girano sullo stesso processore CPU0

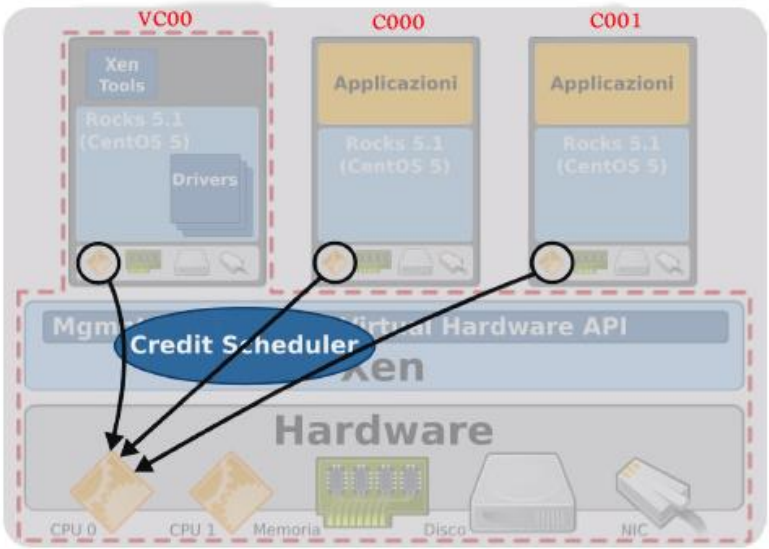
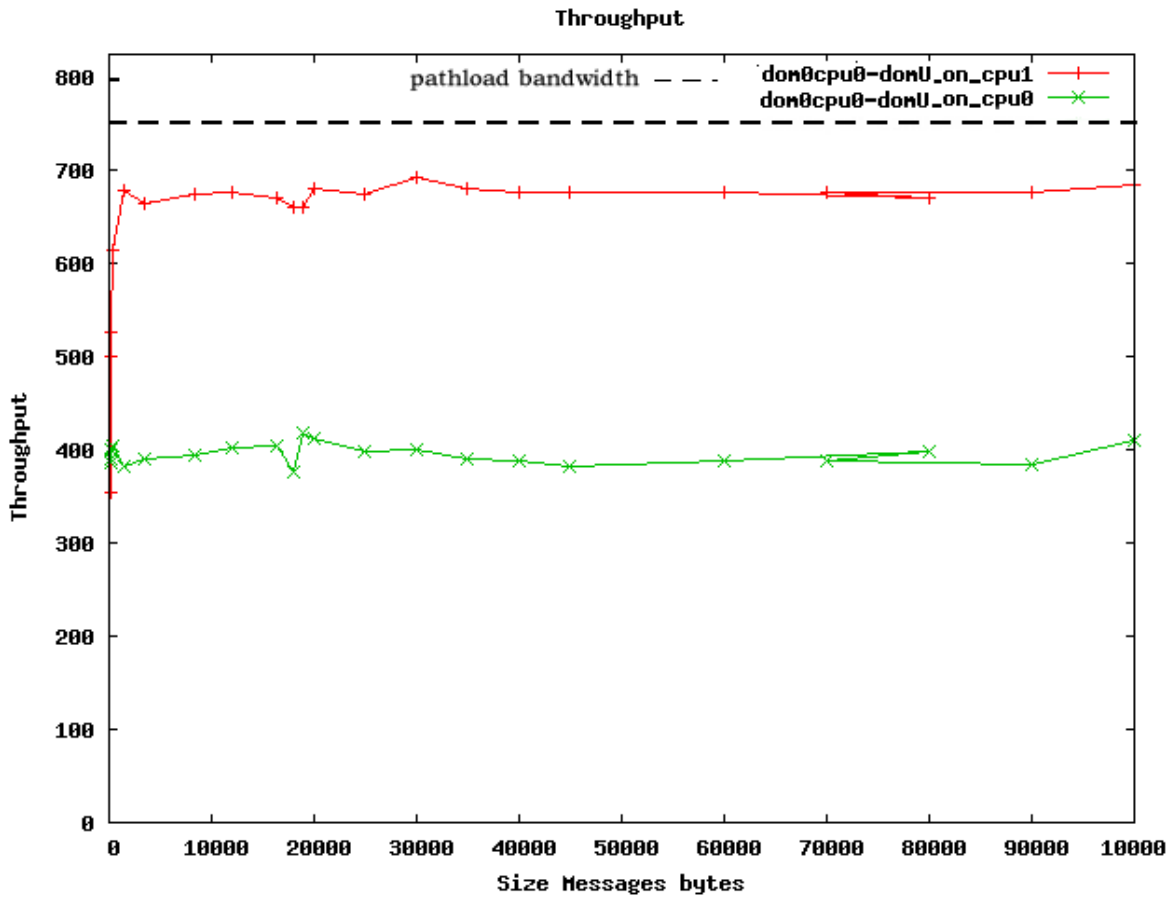


Figura 4.6 – Associazione tra VCPU e la prima CPU fisica

4.5. Risultati

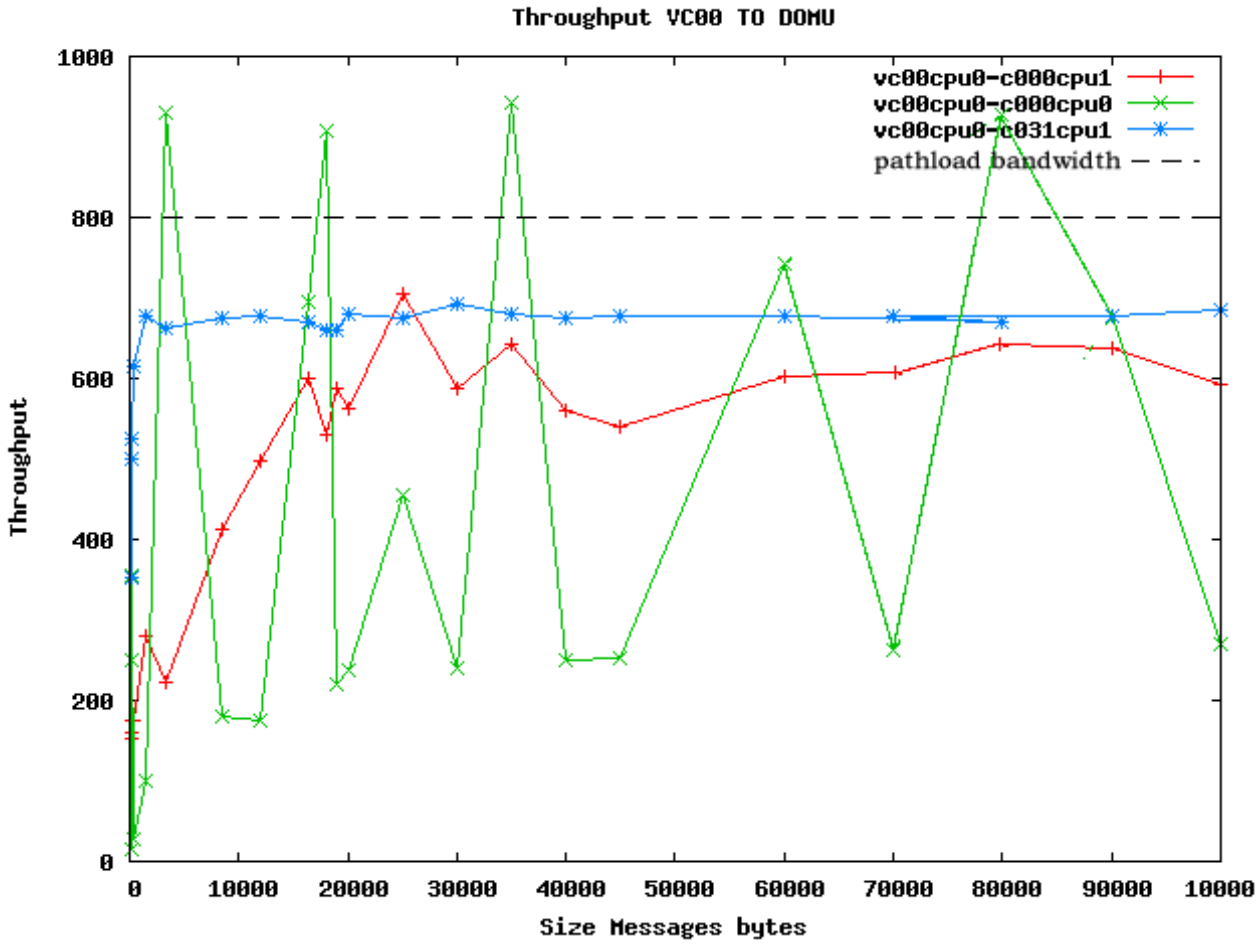


La prima figura si riferisce alle configurazioni in cui il DOM0 fa da sender e la DOMU da receiver:

- Linea rossa, indica il caso in cui la DOM0 gira su una cpu fisica diversa rispetto a quella utilizzata dalla DOMU, e la DOMU gira su un nodo diverso dal nodo della DOM0. In questo caso DOM0-cpu0 e DOMU-cpu1
- Linea verde, indica il caso in cui sia la DOM0 che la DOMU utilizzano lo stesso processore fisico

Pathload calcola una banda disponibile di 750 Mb/s.

Il caso migliore è quando le macchine virtuali girano su processori fisici diversi raggiungendo un Throughput di 700 Mb/s . Il caso peggiore è quando entrambi usano lo stesso processore e si ha quasi un dimezzamento del Throughput 400 Mb/s (-42.85%)



In questa figura si ha il confronto tra le diverse configurazioni dove il DOM0 fa da sender e DOMU da receiver:

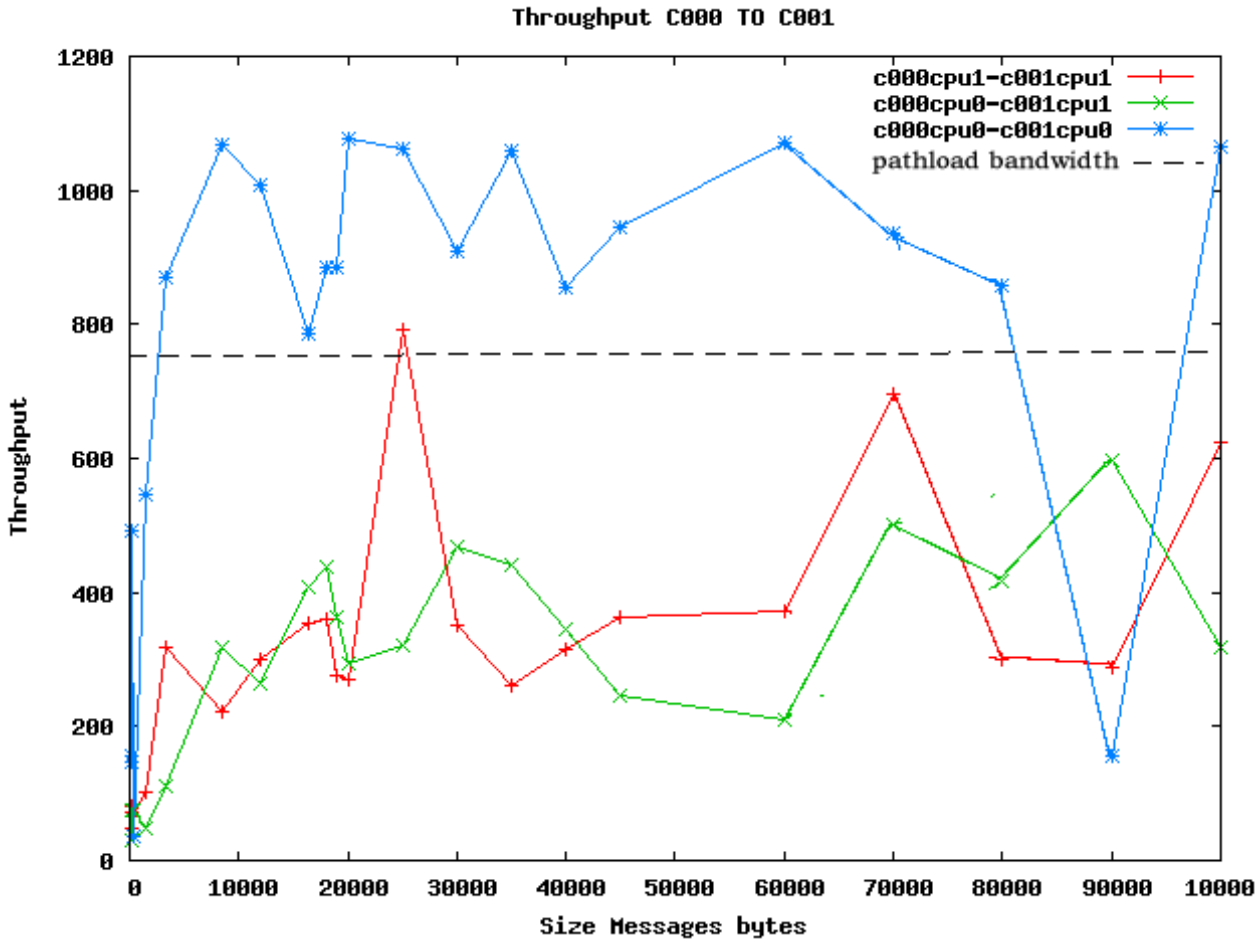
- Linea blu, il caso in cui DOM0 e DOMU girano su processori diversi e su nodi diversi
- Linea rossa, il caso in cui DOM0 e DOMU girano su processori diversi ma stesso nodo
- Linea verde, il caso in cui DOM0 e DOMU girano sullo stesso processore e nodo

Tra DOM0 e DOMU dello stesso nodo Pathload calcola una banda disponibile di 800 Mb/s.

In questo caso possiamo notare che si ha il migliore Throughput 700 Mb/s quando la DOM0 invia pacchetti alle DOMU presenti su altri nodi, in questo caso si raggiunge il valore di picco più velocemente.

Quando DOM0 e DOMU sono sullo stesso nodo si ha un valore di poco inferiore a 700 Mb/s, ed il tempo impiegato per arrivare al picco è molto più lungo.

Il caso in cui DOM0 e DOMU usano lo stesso processore e stesso nodo possiamo notare un andamento altalenante dovuto allo switch dell'uso della cpu fisica.

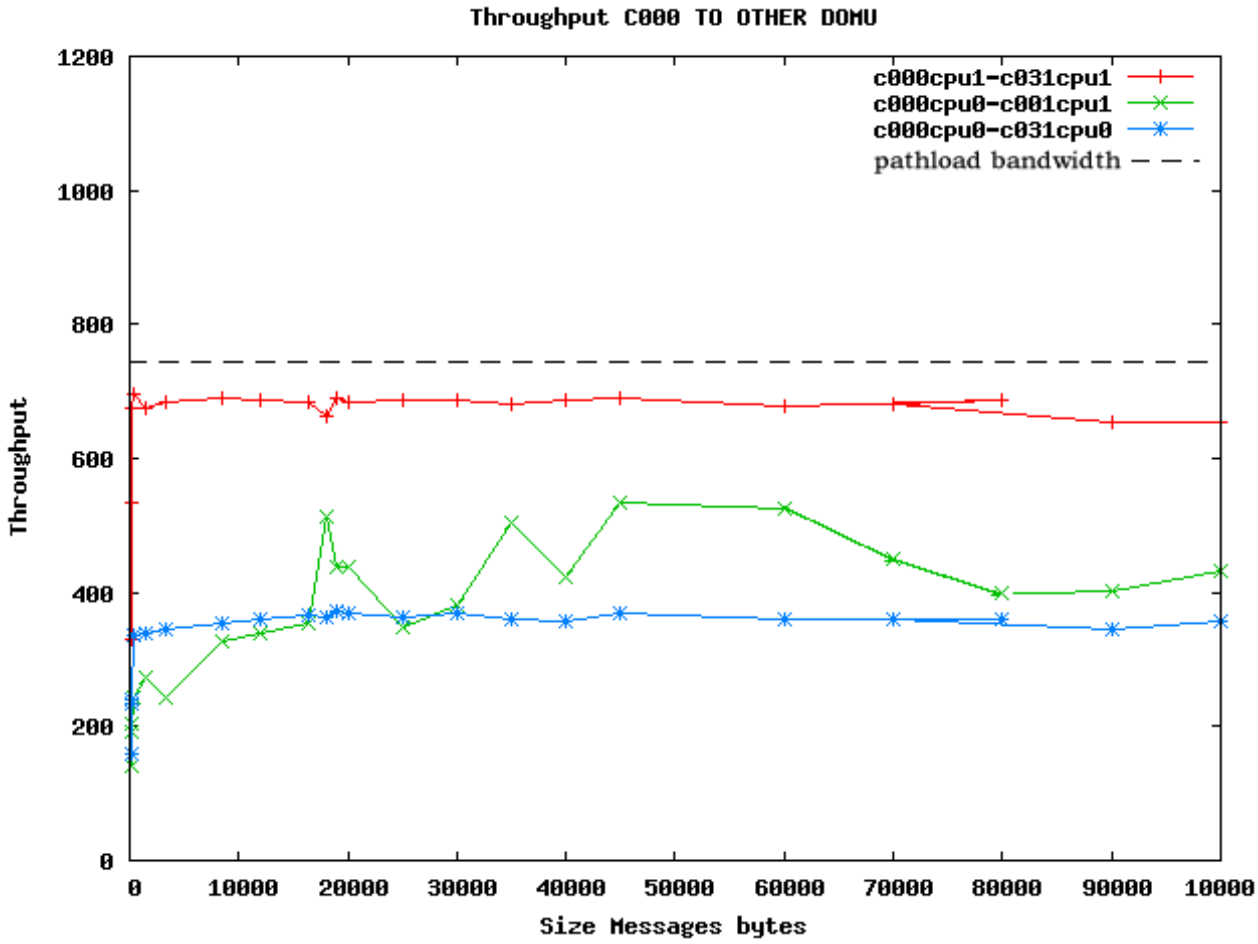


Vengono rappresentati i casi in cui il sender ed il receiver sono entrambi sulle DOMU :

- Linea blu, il caso in cui tutte e due le DOMU di un nodo girano sulla stessa cpu fisica del DOM0
- Linea verde, il caso in cui le due DOMU di un nodo girano su cpu fisiche diverse
- Linea rossa, il caso in cui le due DOMU girano sulla stessa cpu fisica che è diversa da quella della DOM0

Pathload calcola una banda disponibile tra le DOMU di 750 Mb/s

Il caso migliore è quando sia le DOMU che la DOM0 condividono la stessa cpu fisica, mentre gli altri due casi si equivalgono.



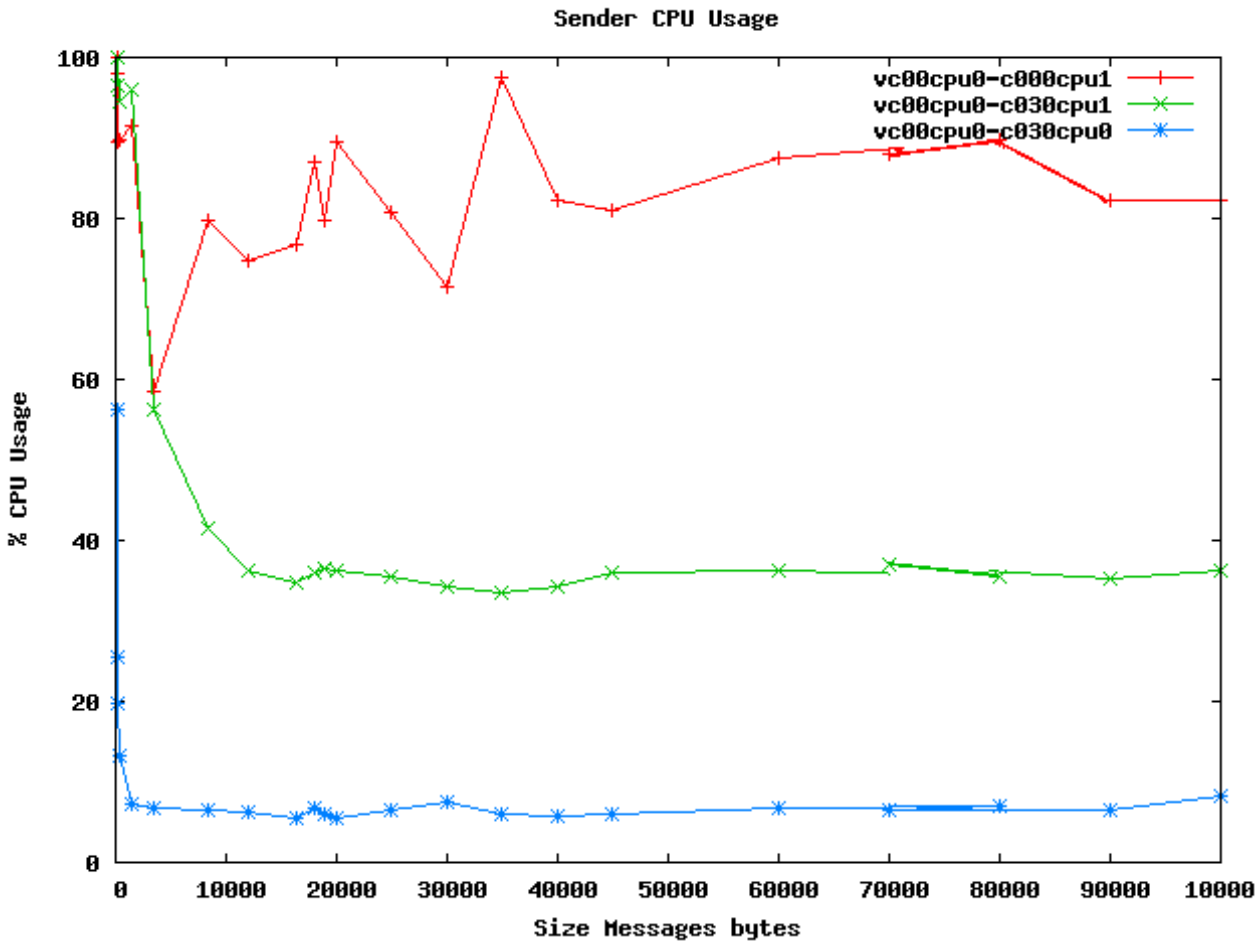
Le configurazioni mostrate sono nel caso in cui le DOMU fanno sia da receiver che da sender

- Linea rossa, le DOMU sono su nodi diversi ed usano una cpu fisica diversa dalla DOM0
- Linea blu, le DOMU sono su nodi diversi ma usano la stessa cpu fisica della rispettiva DOM0
- Linea verde, la DOMU sender (c000) è sulla stessa cpu della sua DOM0 , mentre il receiver (c031) è su un altro nodo ed usa una cpu diversa dalla cpu della propria DOM0

Il caso migliore (equivalente a quello in cui la DOM0 fa da sender) 700 Mb/s è il caso in cui le DOMU girano su cpu fisiche diverse dalle cpu usate dalle rispettive DOM0.

Il caso peggiore 350 Mb/s (-50% rispetto al caso migliore) è quando le DOMU usano la stessa cpu del DOM0, possiamo notare che è ulteriormente scesa rispetto alla stessa configurazione in cui la DOM0 fa da sender (400 Mb/s) di un 12,5%

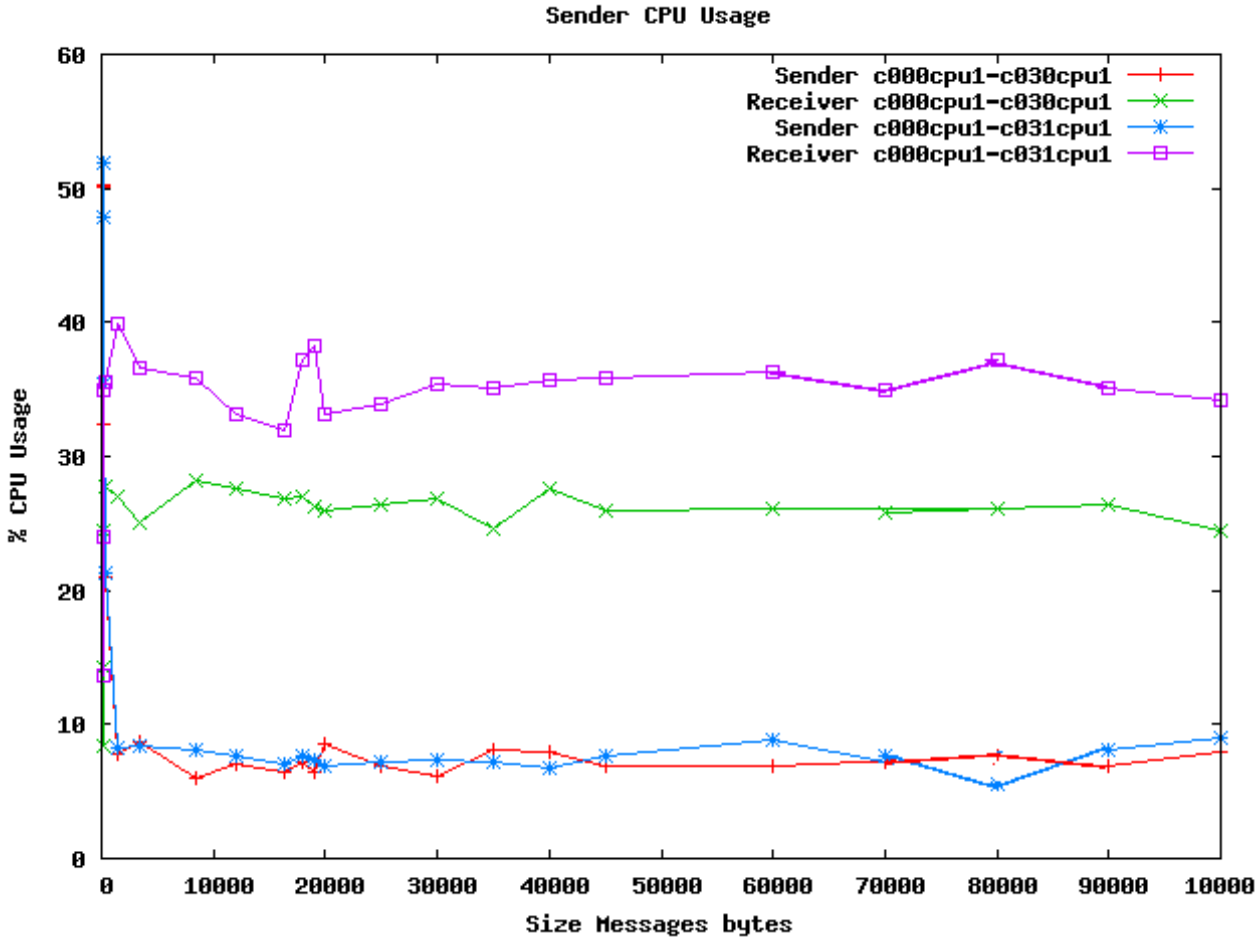
FIGURA 5



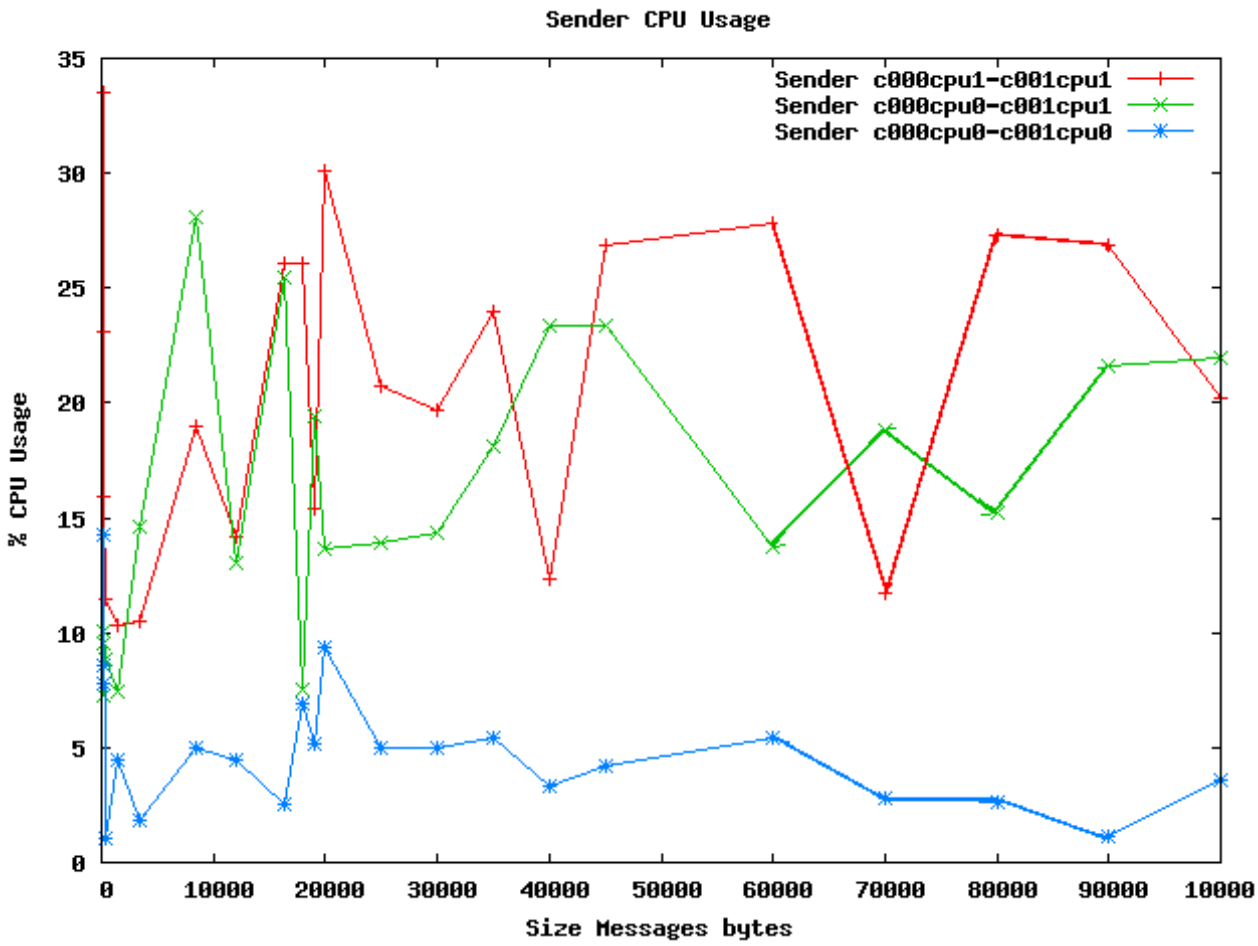
In questo grafico è mostrato l'utilizzo della CPU del sender (DOM0) nei tre casi specificati nella didascalia del grafico.

Come si può notare la CPU è occupata quasi del 100% quando la DOM0 manda pacchetti alla propria DOMU, mentre ha un utilizzo inferiore negli altri due casi.

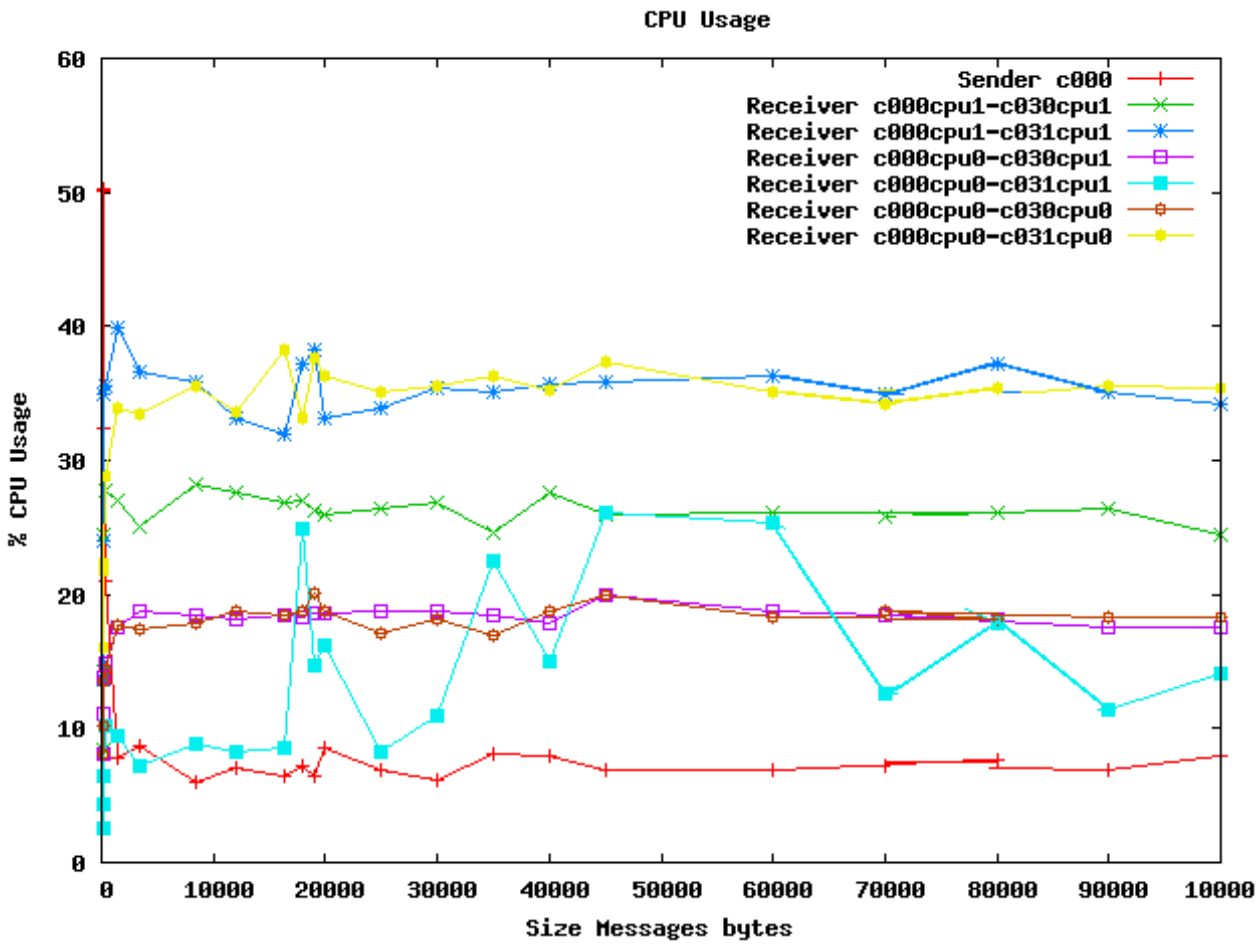
FIGURA 6



In questo grafico viene mostrato come nella fase in cui è la DOMU che fa da sender, l'occupazione della CPU del receiver è maggiore rispetto a quella del sender.



L'occupazione della CPU della c000 che funge da server rispetto alle varie tipologie di affinità tra VCPU e CPU fisica.



In questo grafico vengono mostrati l'occupazione della CPU del Receiver , in base alle diverse affinità VCPU-CPU fisica

ANALISI TRAMITE XENOPROFILE

L'analisi è stata effettuata mentre il DOM0 (vc00) funge da server e la sua DOMU (c000) da receiver.

L'evento scelto è stato: GLOBAL_POWER_EVENTS, il quale considera il tempo durante il quale il processore non viene fermato.

Tabella 1

Sample	**%**	**image**
9444138	95.7425	netperf

Questa tabella mostra la percentuale di occupazione della CPU del DOM0 , come possiamo vedere la quasi totalità della CPU è usata da netperf.

Questo vuol dire che il traffico di rete ha un enorme dispendio di CPU.

Tabella 2

Sample	**%**	**image**
4968415	52.6085	vmlinux
3048907	32.2836	xen-syms-2.6.18-92.1.13.el5
545881	5.7801	netbk
316974	3.3563	ip_conntrack
213704	2.2628	bridge

Questa tabella mostra la distribuzione del tempo di esecuzione di netperf, come possiamo notare la maggior parte del tempo è occupata dal kernel, da xen e dal driver di rete

Tabella 3

Sample	%	Image name	App name
698856	7.0848	Vmlinux	**__copy_from_user_ll**
502514	5.0944	Vmlinux	**get_page_from_freelist**
427408	4.3330	vmlinux	**skb_copy_bits**

Questa tabella mostra la distribuzione dei simboli maggiormente usati dal kernel, come possiamo vedere __copy_from_user_ll è la funzione usata per copiare i dati dal dominio utente al driver della periferica di rete. Skb_copy_bit è una semplice funzione che copia i dati dal buffer di una socket ad una regione nella memoria del kernel . mentre get_page_from_freelist è la funzione che ha il compito di trovare l'indirizzo della pagina al livello kernel dove saranno copiati i dati.

CAPITOLO 5

5. Diagnosi delle performance dei Middleware MPI

In questo capitolo useremo Xenoprof per cercare di spiegare alcuni comportamenti anomali riscontrati durante l'esecuzione di un tool di benchmarking, sviluppato presso questo ateneo (28) (29), sulle prestazioni delle comunicazioni di rete tra Virtual Cluster.

Il tool in questione ha lo scopo di verificare l'efficienza della paravirtualizzazione in sistemi paralleli, in cui le prestazioni sono un aspetto molto critico. I test effettuati hanno il compito di valutare le prestazioni relative alla libreria di comunicazione MPI eseguendo due tipi di test contemporaneamente, il primo funge da carico computazionale, il secondo da micro benchmark per le librerie MPI.

Entrambi i test sono eseguiti su due cluster virtuali, cioè su un insieme di macchine virtuali che girano su nodi computazionali fisici.

5.1. Testbed

Configurazione Fisica:

Il test è stato effettuato utilizzando 3 nodi del cluster fab4 , ognuno dei nodi è composto da due processori e da una memoria di 1536 MB . I nodi sono configurati come vm-container , e il dom0 di ogni nodo è configurato in modo da utilizzare sempre la prima CPU fisica e con 256 MB di memoria.

Tabella riepilogativa

Nodi Fisici	CPUs fisiche	Mem fisica	CPUs dom0	Mem dom0
3	2	1536 MB	1	256 MB

Configurazione Virtuale:

Ogni vm-container (dom0) lancia due macchine virtuali, entrambe le macchine sono configurate con una sola VCPU e con 512 MB di memoria. La prima macchine virtuale gira sulla stessa CPU fisica del vm-container, mentre la secondo gira sulla seconda CPU fisica. Il nostro test sarà composto da due cluster virtuali, Il primo cluster virtuale (VC0) ha come nodi virtuali le domU che hanno in comune la cpu con la dom0, il secondo cluster virtuale (VC1) ha come nodi virtuale le domU che girano su un processore fisico diverso rispetto al dom0.

Tabella riepilogativa

VC	Nodi Virtuali	Mem virtuale	CPUs totali	CPUs condivisa Dom0	CPUs condivisa domU
0	3	512	1	1	0
1	3	512	1	0	0

Sia la piattaforma hardware che quella software, usate per effettuare il test, sono le stesse di quelle usate nel precedente capitolo sulla misurazione delle caratteristiche di rete.

I benchmark utilizzati sono stati : IMB e NAS

IMB

I benchmark utilizzati per valutare l'impatto di Xen sui programmi MPI sono quelli appartenenti alla suite IMB (Intel MPI Benchmarks) , nuovo nome della suite PMB (Pallas MPI Benchmarks). IMB ha lo scopo di fornire un insieme conciso di micro-benchmark per misurare le prestazioni delle principali funzioni MPI. Per fare ciò adotta una precisa metodologia e non impone particolari interpretazioni sui risultati misurati. I risultati dei benchmark sono semplici tempi e vengono accompagnati dai dati sul throughput solo in

casi ben definiti. La versione della suite utilizzata è la 3.1. IMB è diviso in più parti. Quella usata in questa tesi è la prima: IMB-MPI1. La versione 3.1 di IMB-MPI1 contiene i seguenti benchmark:

• PingPong ;

• PingPing ;

• Sendrecv ;

• Exchange ;

• Bcast ;

• Allgahter ;

• Allgatherv ;

• Alltoall ;

• Reduce ;

• Reduce_scatter ;

• Allreduce ;

• Barrier ;

Ciascun benchmark viene eseguito con una lunghezza dei messaggi variabile. I tempi forniti in output, espressi in microsecondi, rappresentano il valore medio calcolato su campioni multipli.

I benchmark sono classificabili in tre categorie: trasferimento singolo, trasferimento parallelo e collettivo. I benchmark a trasferimento singolo sono PingPong e PingPing. In essi viene trasferito un singolo messaggio tra due processi. Il primo processo invia un messaggio e attende la risposta proveniente dal secondo. Il PingPing effettua la misurazione nella circostanza in cui il messaggio inviato viene ostacolato da uno in arrivo (inviato contemporaneamente dal processo ricevente).

I benchmark a trasferimento parallelo sono Sendrecv ed Exchange. In questi benchmark, le attività di un processo sono eseguite in concorrenza con gli altri. I processi vengono organizzati come una catena circolare. Ciascuno di essi ha un processo che lo precede ed uno che lo segue nella catena.

Viene misurata l'efficienza dello scambio di messaggi sotto condizioni di carico globale. Sendrecv valuta il tempo impiegato da un messaggio per attraversare l'intera catena, mentre Exchange valuta il tempo necessario allo scambio di messaggi con i processi vicini (precedente e successivo).

I benchmarck collettivi sono Bcast, Allgahter, Allgatherv, Alltoal , Reduce, Reduce_scatter, Allreduce e Barrier. Essi misurano l'efficienza delle funzioni MPI per la comunicazione collettiva. In questi test diventa rilevante la qualità dell'implementazione della libreria MPI. I risultati dei benchmark che coinvolgono le operazioni di riduzione dipendono anche dall'implementazione di alcune operazioni numeriche.

Nel nostro caso verrà preso di esempio solo il test PingPong.

NAS

Per i nostri test avevamo bisogno di un benchmark che funzionasse da carico computazionale. Per questo scopo è stato usato uno dei macro-benchmark contenuti nella suite NAS (NASA Advanced Supercomputing) Parallel Benchmarks. La suite valuta l'efficienza di un sistema HPC nella gestione di operazioni critiche, appartenenti a simulazioni di missioni spaziali. I benchmark, divisi in varie classi, simulano le caratteristiche di un'applicazione di fluidodinamica computazionale (CFD , Computational Fluid Dynamics) .

In particolare, è stato usato il macro-benchmark EP (Embarrassingly Parallel), appartenente alla classe C della suite. Questo implementa un generatore di numeri casuali in modo " embarrassingly parallel ", cioè la computazione è distribuita e non richiede comunicazione per la generazione dei numeri. La versione del benchmark utilizzata è la 2.4.

5.2. Metodologia

L'attenzione dei test è rivolta alle prestazioni della comunicazione di rete che avvengono nei sistemi paralleli, virtuali e non, in modo da evidenziare possibili fonti di overhead introdotti dalla paravirtualizzazione di Xen. A tale scopo sono stati usati dei domini concorrenti che principalmente condividono l' interfaccia di rete, oppure l'interfaccia ed il processore.

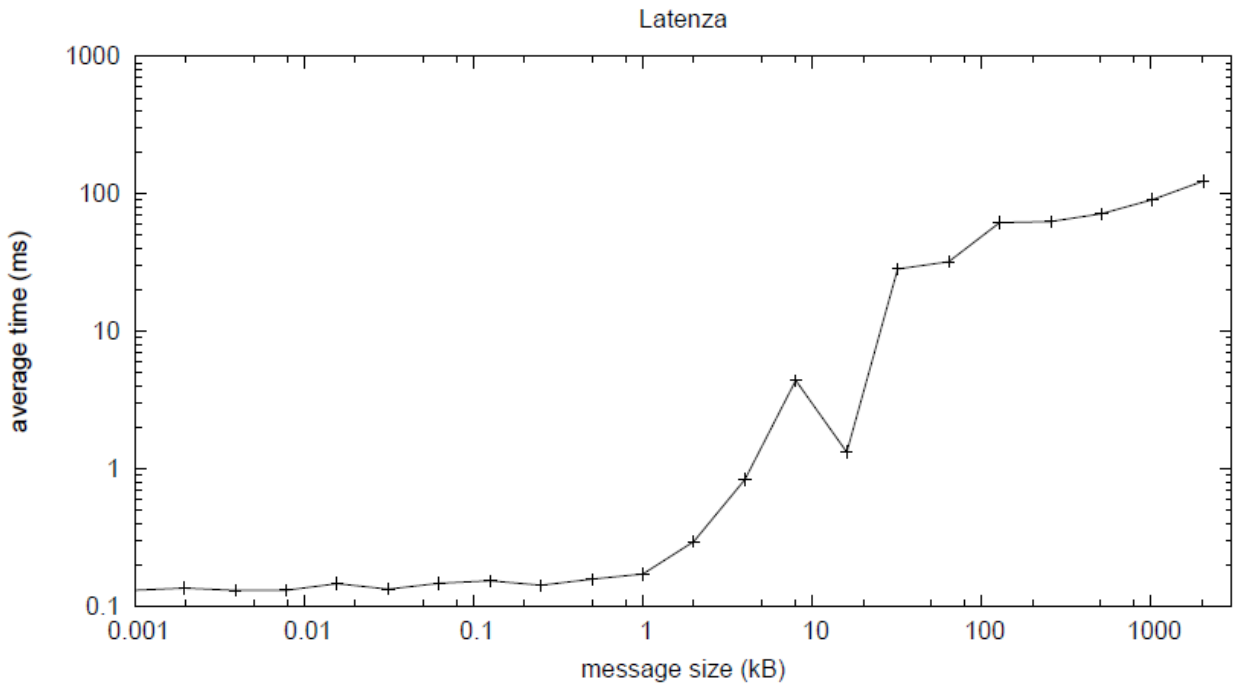
I test sono suddivisi in due configurazioni. La prima configurazione valuta le prestazione dei nodi nel caso in cui essi usino un solo processore. Per questo motivo i domini vengono forzati a competere tra di loro ed il

dominio amministrativo (dom0). Nella secondo si valutano le prestazioni dei nodi usando ambedue i processori fisici.

Nel caso preso in esame per effettuare il profiling si utilizzerà solo la seconda configurazione, in cui le macchine virtuali del primo cluster virtuale (VC0) condividono la cpu fisica con il dom0, mentre il secondo cluster virtuale (VC1) utilizza la seconda cpu.

Il Middleware MPI usato per lanciare i processi sui domini virtuali dei virtual cluster è stato MPICH

5.3. Risultati



In questo grafico viene mostrata la latenza media, calcolata sull'invio di 1000 ripetizioni di un messaggio di 16k, nel caso in cui su VC0 gira il test IMB PingPong, mentre su VC1 gira il bench NAS-EP come carico computazionale .

Come ben visibile dal grafico, abbiamo un' anomalia delle performance per i messaggi di dimensione 16 384 byte.

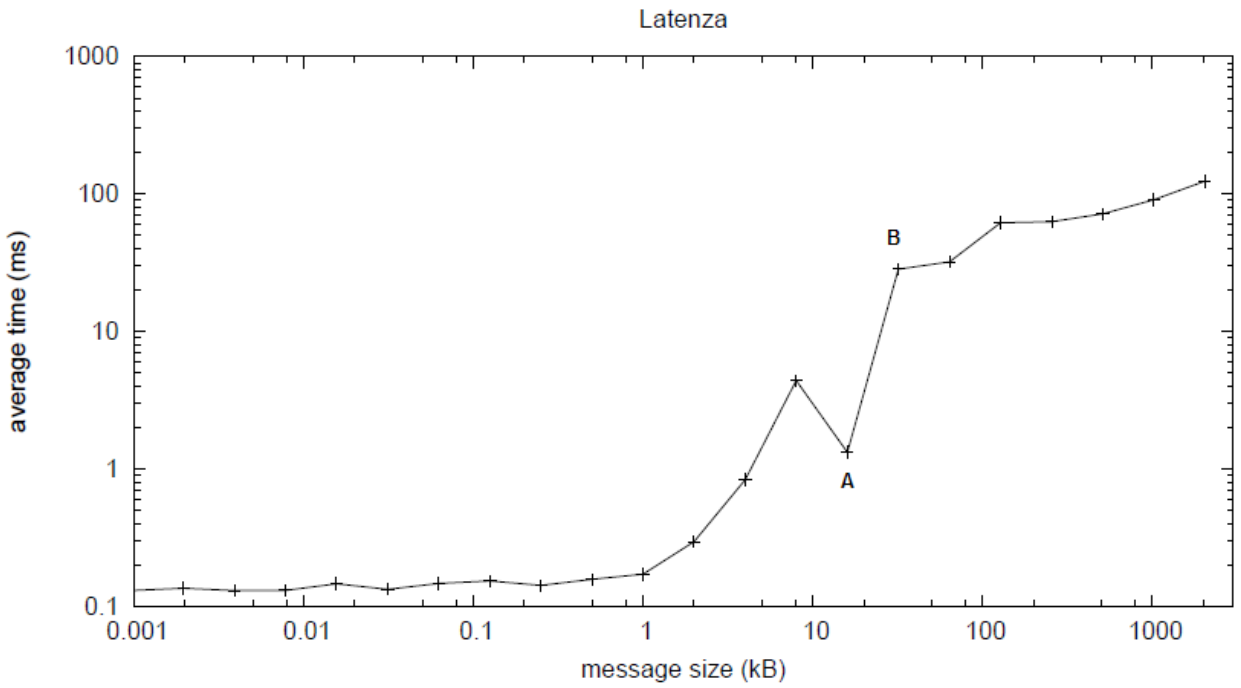
5.4. Analisi tramite Xenoprof

In tale scenario ci può essere di aiuto Xenoprof, in quanto grazie ad un analisi accurata dell'intero sistema e dei singoli ambienti di esecuzione delle domU, possiamo riuscire a localizzare il problema grazie alle distribuzione di eventi hardware avvenuti durante l'esecuzione dei test.

Per capire a cosa è dovuta la variazione della latenza nei messaggi di grandezza 16k, effettuiamo il profiling su due punti del grafico ben precisi dove si riscontra una notevole differenza di prestazione . I punti presi in considerazione saranno:

A → messaggi di grandezza 16K

B → messaggi di grandezza 32K



In questo modo andiamo ad analizzare la distribuzione dei samples raccolti nei due punti del grafico .

La tabella mostrata di seguito si riferisce alla distribuzione degli eventi della macchina virtuale su cui viene effettuata la misurazione del test PingPong, emersi dal contatore Intel-P4 GLOBAL_POWER_EVENTS , inteso come il tempo durante il quale il processore non viene interrotto.

Tabella Riepilogo samples totali

	Punto A #samples	Punto B #samples	% punto A	% punto B
Xen	6867	7745	5%	1%
Vmlinux	13964	14631	11%	2%
IMB-MPI1	77830	1E^06	61%	95%
Xennet	338	725	0%	0%
Strings	17434	17389	14%	2%
Phyton	7991	7857	6%	1%
Sge_execd	2076	2038	5%	1%
Irqbalance	837	1226	1%	0%

L'unica differenza, in termini di samples, tra le due configurazioni è data dal simbolo IMB-MPI1. Infatti nel punto B il numero dei samples è molto maggiore rispetto a quello del punto A.

Andando ad ispezionare le chiamate ai simboli effettuati solo da IMB-MPI1 si ha la seguente tabella.

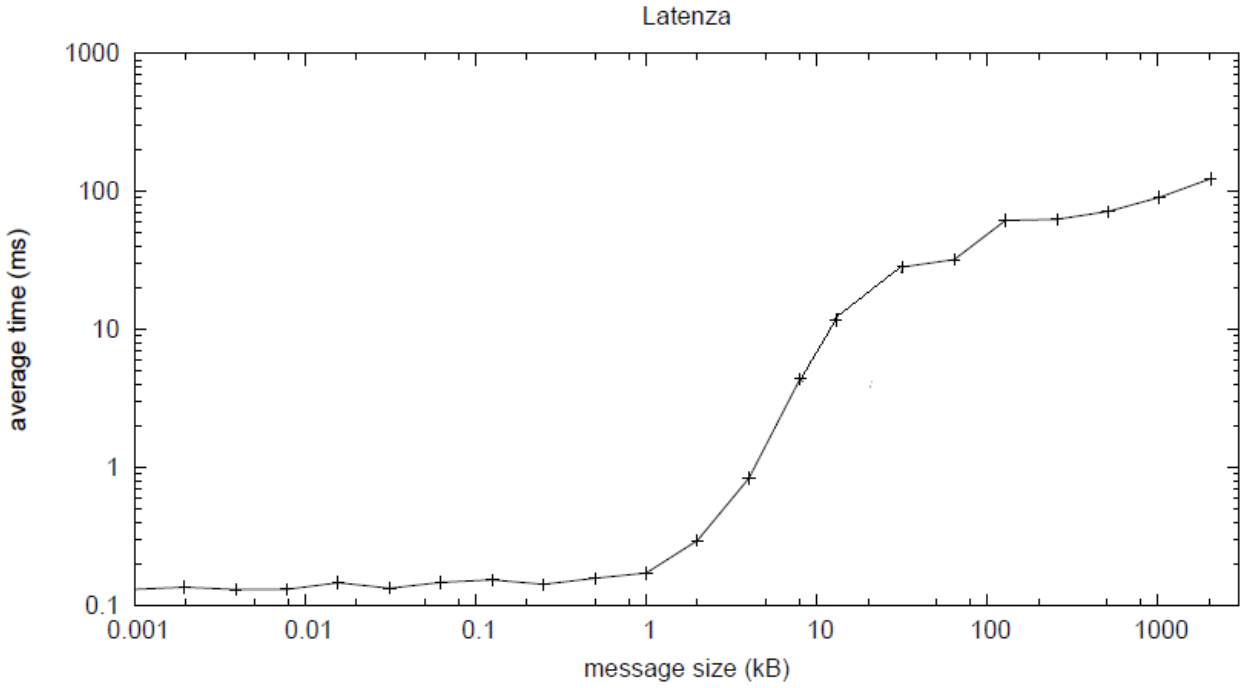
Tabella riepilogo samples totali di IMB-MPI1

IMB-MPI1	Punto A #samples	Punto B #samples	% punto A	% punto B
Vmlinux	44567	584375	57.26 %	56.17 %
IMB-MPI1	21744	331766	27.93 %	33.17 %
[vdso]	4648	93690	5.97 %	9.05 %
libc-2.5.so	692	6927	0.88 %	0.66 %
xen-syms	3822	6425	4.19 %	0.61 %
Xennet	1317	2135	1.69 %	0.20 %

Come è possibile dedurre da questa tabella, anche le distribuzione dei simboli chiamati dallo specifico test IMB-MPI1 è quasi identica in ambedue i casi.

Fatte queste osservazione si è arrivati a dedurre che la colpa dell'andamento anomalo nel punto A fosse imputabile al middleware usato, in quanto il numero minore delle occorrenze di IMB-MPI1 nel punto A rispetto a quello del punto B non è da addebitare a routines interne che introducono overhead.

Infatti come dimostra il grafico seguente, con l'uso della libreria di comunicazione OPENMPI al posto della MPICH tale anomalia scompare.


Latenza
average time (ms)
1000
100
10
1
0.1
0.001
0.01
0.1
1
10
100
1000
message size (kB)

Conclusioni

In questa tesi è stata valutata l'efficacia dell'uso della paravirtualizzazione nei sistemi cluster, ovvero di sistemi paralleli in cui le prestazioni sono un aspetto critico.

Nella prima parte della tesi si è parlato della virtualizzazione in generale, mettendo a confronto le varie tipologie adottate. Una volta scelta la tipologia a noi più consona, la paravirtualizzazione, si è passati a descrivere Xen ed il suo ambiente di esecuzione. Un ulteriore capitolo ci è servito a descrivere i sistemi cluster e le varie tipologie adottate, descrivendo anche i virtual cluster.

Le applicazioni parallele richiedono un comportamento di I/O efficiente e consistente. Tradizionalmente, gli hypervisor hanno avuto l'obiettivo di assicurare una condivisione equa dei processori, trattando lo scheduling delle risorse di I/O come un problema secondario.

In quest'ottica si sono analizzate alcune prestazioni di rete delle macchine virtuali, tra cui il throughput di rete, la banda disponibile e l'uso della CPU tra due processi sender e receiver. Mettendo in evidenza la notevole perdita di performance nel caso in cui sia il sender (DOM0) che il receiver (DOMU) si trovino sulla stessa CPU fisica, oppure il caso in cui il sender (DOMU) ed il receiver (DOMU) utilizzino la stessa CPU fisica ma non quella utilizzata dal DOM0.

I casi migliore in cui il DOM0 funge da sender e le DOMU da receiver si ha quando le macchine virtuali (DOMU) si trovano su CPU fisiche diverse dal DOM0, mentre nel caso in cui la DOMU fa sia da sender che da receiver, il caso migliore è dato quanto entrambe le DOMU si trovano sulla stessa CPU fisica del DOM0.

Un ulteriore analisi dei risultati è stata effettuata grazie a Xenoprof, il quale effettuando il profiling dei sistemi virtuali ci ha aiutato a capire la causa delle anomalie presenti in un altro tool di benchmarking delle comunicazioni di rete, sviluppato nel nostro stesso ateneo.

In questo caso si è effettuato il profiling su due punti ben precisi del grafico mostrato, il primo (A) in cui si presentava l'anomalia, il secondo (B) dove essa non si presentava. Così facendo siamo giunti alla conclusione che la perdita delle performance era da imputare alla libreria di comunicazione MPICH, la quale

era l'unica responsabile delle perdita del numero delle occorrenze di IBM-MPI1 nel caso A rispetto al caso B, dimostrando che tramite OPENMPI tale anomalia scompare.

Sviluppi futuri

Un ulteriore analisi più approfondita, tramite l'uso del profiling sull'anomalia riscontrata sulla libreria di comunicazione MPICH può essere di aiuto nel capire e analizzare nel dettaglio il problema. Per ora si è solo identifica l'area del problema, ma lavori successivi potrebbero focalizzarsi sulla libreria di comunicazione per capire da cosa è dato questo squilibrio delle performance.

Per quanto riguarda il tool implementato in questo lavoro di tesi, sarebbe interessante confrontare i risultati ottenuti con quelli che si otterrebbero in un ambiente nativo. In modo da avere un confronto di prestazioni tra le varie configurazioni provate (comprese quelle con i migliori risultati) e le prestazioni di un sistema nativo, così da avere una visione più ampia e significativa della perdita delle prestazione di rete.

Bibliografia

1. Umberto Villano, Emilio Mancini , Raffele Tretola, Massimiliano Rak. "*A framework for Virtual Cluster Performance Evaluation*," 2009.

2.Wikipedia. Timeline of *Virtualization Development.* [Online]. http://en.wikipedia.org/wiki/Virtualization_Development.

3. Java programming language. *Sun microsystems.* [Online] http://java.sun.com.

4. Robert P. Goldberg, Gerald J. Popek . "*Formal Requirements for Virtualizable third generation architectures".*Communications of the ACM 17(7), 1974.

5. Hystory Virtualization. *VMWare.* [Online] http://www.vmware.com/it/overview/history.html.

6.AMD.VirtualizationAMD-V.[Online] http://www.amd.com/us/products/technologies/virtualization/Pages/amd-v.aspx.

7. Intel. Intel Virtualization Overview. [Online] http://www.intel.com/technology/virtualization/.

8. Xen. *Xen.* [Online] http://www.xen.org.

9. Sourceforge. User-Mode Linux. [Online] http://user-mode-linux.sourceforge.net/.

10. Official Linux Kernel. The Linux Kernel Archives. [Online] http://www.kernel.org/.

11. University of Cambrige. XenoServers. [Online] http://www.xenoservers.net.

12. K. A. Fraser, S. M. Hand, T. L. Harris, I. M. Leslie, and I. A. Pratt. "*The Xenoserver computing infrastructure". Technical Report*. january *2003*

13. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris. "*Xen and the Art of Virtualization",* in Proceedings of the nineteenth ACM symposium on Operating systems principles, October 2003.

15. Gagne, Abraham Silberschatz Peter Baer Galvin Greg. "*Operating System Concepts".* John Wiley & Sons, 2004.

16. John Smith. XenStore . [Online] http://wiki.xensource.com/xenwiki/XenStore.

17. David Chisnall. "*The definite guide to the Xen Hypervisor,"* Pearson Education, 2008.

18. The OpenMP API specification for parallel programming . [Online] http://openmp.org/wp/.

19. Berkeley UPC - Unified Parallel C Project. [Online] http://upc.lbl.gov/.

20. Tassonomia di Flynn. [Online] http://it.wikibooks.org/wiki/Supercomputer/Tassonomia_di_Flynn.

21.Rocks User Documentations. Wiki Rocks. [Online] http://www.rocksclusters.org/wordpress/?page_id=4.

22.Hewlett-Packard. Netperf Benchmark. [Online] . www.netperf.org.

23. Prof.Constantinos Dovrolis,Manish Jain, Ravi Prased. Pathrate A measurement tool for the capacity of network paths. [Online] http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bw-est/pathrate.html.

24. HP & OpenSource. Xenoprof - System-wide profiler for Xen VM. [Online] http://xenoprof.sourceforge.net/.

25. Oprofile. A system profiler for Linux. [Online] http://oprofile.sourceforge.net/news/.

26. Aravind Menon, Jose Renato Santos , Yoshio Turner , G.Janakiraman , Willy Zwaenepoel." *Diagnosing Performance Overhead in the Xen Virtual Machine Environment".* Chicago, Illinois , USA : ACM, 2005.

27. Aravind Menon, Alan L. Cox , Willy Zwaenepoel. "*Optimizing Network Virtualization in Xen, in 2006 USENIX Journal on Selected Areas in Communications, 2006*.

28. Tretola, Raffaele.*"Virtual Clustering: valutazione di prestazioni in ambiente Xen/Rocks". Tesi di laurea* A.A. 2007/2008.

29. Cuomo, Antonio. *"Valutazione delle prestazioni di sistemi cloud nel contesto dell'High Performance Computing". Tesi di laurea* A.A. 2008/2009.

30. Michael J. Flynn, " Some computer organization and their effectiveness," IEEE Transactions on Computers, vol 21.

31. David E. Williams , Juan Garcia , *" Virtualization with Xen,"* Syngress Publishing , 2007.

31. William von Hagen, " Professional Xen Virtualization," Wiley Publishing , 2008.