ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

TESI DI LAUREA

in Sistemi Operativi T

Monitoraggio e bilanciamento del carico in sistemi virtualizzati tramite Xen Cloud Platform

CANDIDATO

Andrea Giardini

RELATORE

Chiar.ma Prof.ssa Anna Ciampolini

Anno Accademico 2011/2012

Sessione III

Indice

0. Introduzione: inquadramento generale e obiettivi della tesi

Il lavoro di tesi qui presentato è orientato allo sviluppo di un'applicazione per il bilanciamento di carico e migrazione di macchine virtuali all'interno di una infrastruttura cloud. Negli ultimi periodi infatti si è sentito molto parlare del cosiddetto "cloud computing", ovvero la possibilità per un utente di utilizzare una risorsa o un insieme di risorse solo per il tempo necessario, garantendo alti margini di affidabilità ed efficienza. L'elasticità che caratterizza una piattaforma di questo tipo è data dalla virtualizzazione, ovvero la capacità di suddividere le risorse di un server fisico tra più clienti, fornendo solo una porzione delle risorse disponibili.

L'obiettivo di questa tesi è quello di fare un di chiarezza sulle attuali tecniche di virtualizzazione presenti sul mercato e districarsi tra le numerose alterative e piattaforme in grado di fornire un servizio di questo tipo. Inizialmente verranno dunque analizzate i pregi e i difetti di una infrastruttura cloud, evidenziandone le caratteristiche e confrontando tra loro le soluzioni gratuite e a pagamento attualmente presenti.

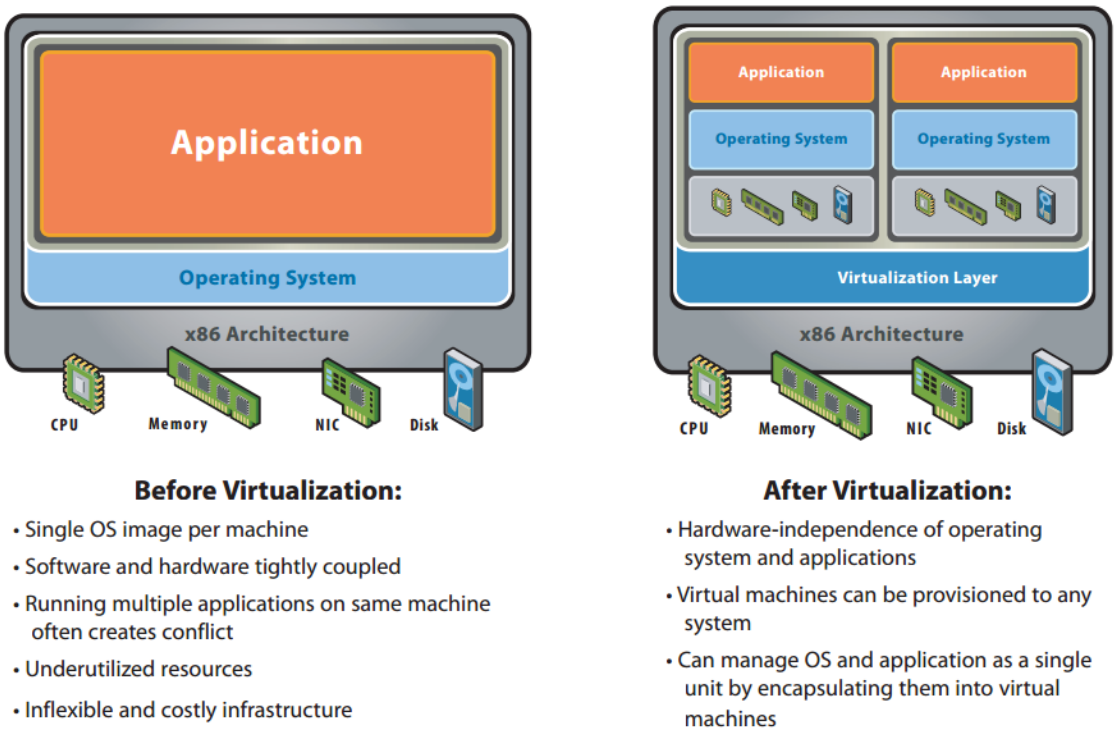
Successivamente si è concentrata l'attenzione sulla piattaforma Xen Cloud Platform, studiandone i meccanismi di virtualizzazione, gli applicativi che la compongono e l'interazione con le API di programmazione. Durante la configurazione dell'ambiente di test sono state affrontate problematiche che hanno evidenziato i punti deboli di una architettura di questo tipo, portando allo sviluppo di una soluzione software che potesse risolvere, almeno in parte, alcune delle mancanze osservate.

In seguito è stato dunque progettato e realizzato un programma in linguaggio Java che, tramite API di programmazione, potesse fornire all'amministratore la capacità di definire regole personalizzate per il bilanciamento del carico e l'automatizzazione delle operazioni di migrazione ed, allo stesso tempo, fornire informazioni dettagliate sull'infrastruttura osservata. Tramite lo studio di un algoritmo di bilanciamento adeguato e l'estensiva fase di test è stato possibile verificare l'effettiva efficienza dell'applicazione nel redistribuire dinamicamente il carico su server meno problematici.

Nella parte finale inoltre sono state inserite alcune conclusioni sull’esperimento effettuato, riportando alcuni risultati sperimentali per poterne valutare le performance, inoltre sono state suggerite alcune possibili estensioni dell’applicazione.

1. Sistemi di Virtualizzazione

1.1. Introduzione alla virtualizzazione



1. Virtualizzazione – Fonte: http://www.vmware.com/pdf/virtualization.pdf

In informatica il termine virtualizzazione è utilizzato per indicare una tecnica che permette di astrarre dall'hardware di un computer (host) delle componenti hardware virtuali permettendo all'utilizzatore di suddividere un'unica macchina fisica in più macchine virtuali, ognuna con il proprio sistema operativo e con le proprie risorse dedicate (Figura 1). Ogni macchina virtuale è detta guest, cioè ospite, dell'host fisico: ad essa sono affidate un numero limitato delle risorse della macchina che la ospita. Il software di virtualizzazione, detto Hypervisor o Virtual Machine Monitor, nasconde alle macchine guest l'hardware sottostante mostrando un'astrazione delle risorse presenti, si occupa di gestire le macchine virtuali decidendo come suddividere le risorse tra i vari guest e dedicando un tempo limitato di esecuzione ad ognuno di questi. Per poter utilizzare delle risorse la macchina guest deve effettuare continue richieste all'hypervisor che si occuperà di fornirle le risorse di cui necessita, agendo da intermediario. Ovviamente è compito dell'hypervisor limitare le risorse a disposizione di una macchina virtuale, evitando che questa possa andare ad interferire con gli altri

guest presenti sullo stesso nodo o con la stabilità del nodo stesso. La macchina virtuale risulta dunque un'entità completamente separata dall'host fisico e dalle altre macchine virtuali sullo stesso nodo e non deve essere in grado di influenzare in alcun modo la stabilità del sistema. Risulta dunque evidente il compito di grande responsabilità affidato al meccanismo dell'hypervisor che risulta molto complesso ed articolato.

1.2. Differenti tipologie di virtualizzazione

Il tema della virtualizzazione ha avuto molte applicazioni negli ultimi anni, ciò ha permesso lo sviluppo di differenti tecnologie e metodi di virtualizzazione ognuno dei quali è indicato per un determinati tipo di applicazioni [1]. Ogni tipologia introduce vantaggi e svantaggi, affrontando con tecniche differenti i problemi di un sistema virtualizzato [2]. Andremo ora ad analizzare le tre principali tecniche di virtualizzazione descrivendone l'efficacia e le problematiche a cui sono soggette.

1.2.1. Paravirtualizzazione

Un sistema paravirtualizzato è un sistema che presenta alla macchina virtuale un hardware simile ma non identico a quello dell'host sottostante: in questo tipo di virtualizzazione il sistema operativo della macchina guest, tramite opportune modifiche, è in grado di comunicare con l'hypervistor dell'host fisico. L'obiettivo di questo collegamento tra guest e hypervisor è quello di sgravare il sistema virtualizzato di alcune istruzioni che risulterebbero complesse da eseguire in un ambiente virtuale, lasciando all'hypervisor il compito di eseguirle al suo posto e riportarne i risultati. E' dunque necessario che il sistema operativo sia modificato in modo da rispettare alcune API che consentiranno la comunicazione con l'hypervisor.

1.2.2. Virtualizzazione Completa

L'utilizzo di questo tipo di virtualizzazione permette di installare un qualsiasi tipo di sistema operativo sulle macchine guest senza dover apportare alcuna modifica al suo codice e senza introdurre moduli aggiuntivi: il codice binario viene eseguito direttamente sul processore dell'host o eventualmente viene tradotto, nel caso in cui l'architettura della macchina virtuale e quella dell'host fisico siano differenti,

per corrispondere all'architettura dell'host. Ciò può risultare molto utile nella virtualizzazione di sistemi operativi non modificabili per licenza, come ad esempio Windows, che non permettono l'aggiunta di moduli aggiuntivi al Kernel. La possibilità di installare un qualsiasi sistema operativo senza dover modificare il codice della macchina guest è indubbiamente un notevole vantaggio, ma la traduzione del codice binario nell'architettura corrispondente può aggiungere un notevole overhead computazionale che può comportare notevoli decadimenti prestazionali: la virtualizzazione completa risulta infatti la peggiore delle soluzioni dal punto di vista delle performance. Sebbene quindi esistano simulatori di CPU, come ad esempio Qemu, molto completi e in grado di simulare numerose architetture differenti, il carico computazionale dovuto all'esecuzione e alla duplice traduzione dell'istruzione risulta comunque molto elevato. Un aspetto chiave della virtualizzazione completa è l'intercettazione e la simulazione di operazioni privilegiate: ogni operazione privilegiata invocata da una macchina virtuale deve essere intercettata e gestita dall'hypervisor.

1.2.3. Virtualizzazione Assistita

La virtualizzazione assistita permette, tramite l'utilizzo di tecnologie come Intel VT-x ed AMD-V, di fornire una virtualizzazione completa in maniera più efficiente e performante, permettendo di eseguire direttamente sull'hardware alcune istruzioni della macchina virtuale. E' importante notare che l'attuale virtualizzazione assistita permette di facilitare solo le operazioni che riguardano direttamente la cpu: le chiamate alla memoria o alle periferiche devono essere comunque emulate come accade nella virtualizzazione completa. Oltre ai vantaggi precedentemente descritti, questo tipo di soluzione permette di simulare anche sistemi operativi a 64 bit anche su host fisico a 32 bit, cosa che non è possibile fare con la virtualizzazione completa.

1.3. Vantaggi e svantaggi di un sistema virtualizzato

L'utilizzo di un sistema virtualizzato, soprattutto a livello aziendale, permette di ottimizzare notevolmente l'utilizzo degli host disponibili utilizzandoli al massimo delle loro capacità [3]. In particolare una infrastruttura virtualizzata consente di:

- Ridurre il numero di server fisici – Una soluzione virtualizzata permette di sostituire più server fisici con una molteplicità di macchine virtuali su un numero più ridotto di host. Aggregando dunque più servizi su diverse macchine virtuali posso ridurre il numero di server fisici presenti nel datacenter e abbattere notevolmente i costi di gestione. Un numero inferiore di host fisici comporta infatti una riduzione degli spazi all'interno del rack, un consumo inferiore di energia elettrica per alimentazione e raffreddamento, minori spese di mantenimento e di installazione.
- Ottimizzazione delle risorse – All'interno di un'azienda possono esistere numerosi server che fanno un utilizzo molto basso di risorse: può essere utile dunque unire tutti questi server a basso consumo su un unico host fisico.
- Indipendenza Hardware – L'installazione di un sistema operativo è fortemente legata alla macchina sulla quale tale sistema operativo è in esecuzione ed ha uno stretto legame con il suo hardware. Nel caso in cui un'installazione dovesse essere spostata su un'altra macchina con hardware diverso è dunque necessario tener sempre conto di eventuali problemi di incompatibilità che possono verificarsi. La virtualizzazione in questo caso, mostrando al sistema operativo un'astrazione dell'hardware sottostante, permette di rimanere sempre indipendenti dalle caratteristiche fisiche dell'host, dando la possibilità di spostare una macchina da un server fisico ad un altro senza alcun problema di incompatibilità.
- Allocazione dinamica delle risorse - Con il tempo alcune macchine potrebbero richiedere più risorse di quelle a disposizione o magari risultare sovradimensionate per il loro compito, in questo caso la virtualizzazione permette di allocare o deallocare dinamicamente risorse computazionali ad una determinata macchina virtuale senza dover intervenire a livello hardware come accadeva precedentemente.
- Creazione di macchine personalizzate – Grazie all'astrazione dell'hardware un sistema di virtualizzazione permette di creare delle immagini di macchine virtuali pronte all'uso velocizzando notevolmente l'installazione e la messa in produzione di macchine di uso comune.

- Creazione di ambienti di test – Può capitare frequentemente di dover utilizzare ambienti di test per provare nuove configurazioni prima di applicarle su un sistema di produzione. Un ambiente virtuale permette di clonare una macchina di produzione e vedere in anticipo eventuali problematiche che possono verificarsi prima di applicare la configurazione sull'ambiente di produzione.

D'altro canto la virtualizzazione non è esente da problematiche e svantaggi, in particolare possiamo citare:

- Overhead – Una soluzione virtualizzata, per quanto possa essere ottimizzata, non sarà mai in grado di confrontarsi con una soluzione tradizionale a parità di hardware. In particolare una soluzione virtualizzata potrebbe non essere adatta per applicativi che abbiano richieste molto esigenti in termini di performance.
- Hardware non virtualizzabile – Alcune periferiche potrebbero non essere virtualizzabili, in tal caso l'unica soluzione adottabile è quella tradizionale.

1.4. Principali software di Virtualizzazione a confronto

1.4.1. Xen

Xen [4] è un monitor di macchine virtuali open source sviluppato dall'Università di Cambridge [5]. La tecnologia utilizzata è quella della paravirtualizzazione: Xen infatti non emula un hardware virtuale, ma piuttosto cerca di regolare e bilanciare al meglio l'utilizzo di risorse fisiche tra le macchine virtuali. Un approccio di questo tipo permette di ottenere un decadimento delle prestazioni minimo.

1.4.2. VMWare

VMWare [6] prevede lo sviluppo di più soluzioni dedicate alla virtualizzazione, sia gratuite che a pagamento. Tutte queste si basano sul meccanismo di virtualizzazione completa, intercettando le operazioni privilegiate delle macchine virtuali e traducendole nelle istruzioni corrispondenti.

1.4.3. OpenVz

OpenVz [7] è un software in grado di creare una molteplicità di istanze di sistemi operativi differenti detti Container: non si tratta infatti di un vero e proprio software di virtualizzazione tradizionale poiché non è in grado di gestire allo stesso tempo più sistemi operativi con kernel differenti. La tecnologia di OpenVz infatti, al contrario di Xen e VMWare, si basa su un unico kernel linux installato sull'host avente al suo interno una patch per poter gestire più container, ed è in grado di avviare solo sistemi operativi basati su linux poiché è necessario che tutti questi condividano la stessa architettura e versione del kernel. Poiché OpenVz non necessita di un hypervisor risulta molto veloce e performante visto che l'overhead viene ridotto notevolmente, allo stesso tempo però un'architettura di questo tipo comporta anche degli svantaggi dovuti alla presenza di un unico kernel alla base di tutti i container: tutti i sistemi virtualizzati devono essere infatti in grado di funzionare correttamente con la stessa versione kernel dell'host. Un altro vantaggio notevole dato da OpenVz è la possibilità utilizzare la memoria non utilizzata dai container come cache per il disco, velocizzando ulteriormente le operazioni di lettura e scrittura da disco rigido. Ogni container è rappresentato all'interno dell'host da una semplice cartella, contenente tutti i suoi dati, e da un file di configurazione nel quale sono descritte tutte le proprietà del container indicato: i dati vengono isolati tramite un meccanismo di chroot che impedisce ai container all'interno di uno stesso host di interferire tra di loro.

1.4.4. KVM

KVM [8], acronimo di Kernel-based Virtual Machine, è un sistema di virtualizzazione basato sul kernel linux che supporta la virtualizzazione completa tramite i meccanismi hardware citati in precedenza (Intel VT e AMD-V). Per alcuni dispositivi è inoltre disponibile un supporto alla paravirtualizzazione nel caso di interfacce di rete e controller disco.

2. Cloud Computing

2.1. Introduzione al Cloud Computing

Con il termine Cloud Computing si intende la possibilità, per un utente, di utilizzare delle risorse computazionali attraverso la rete. In particolare un provider che mette a disposizione dei suoi clienti una infrastruttura cloud è in grado di fornire risorse in base alle loro reale necessità, aumentando o riducendo le capacità delle istanze in maniera dinamica e trasparente. D'altro canto il cliente potrà gestire indipendentemente la potenza, il numero e la capacità di ogni istanza impostando regole e definendo, tramite un'interfaccia, le modalità di gestione della sua infrastruttura virtuale. Una architettura cloud prevede l'installazione di più server fisici, collocati all'interno della server-farm del provider fornitore del servizio, generalmente configurati con meccanismi di alta affidabilità e ridonandanza in modo escludere ogni possibile malfunzionamento.

2.2. Obiettivi di una infrastruttura Cloud

L'obiettivo di una infrastruttura cloud è generalmente quello di fornire un servizio che sia scalabile, personalizzabile e potenzialmente illimitato. Il cloud punta a fornire all'utente o all'azienda delle risorse che autonomamente non sarebbe stata in grado di sostenere, pagandole solo per l'effettivo utilizzo, senza poi contare l'estrema elasticità con cui queste possono essere diminuite o aumentate a seconda del carico di lavoro.

2.3. Vantaggi e svantaggi di una infrastruttura Cloud rispetto ad una soluzione tradizionale

Il Cloud Computing introduce un'importante innovazione in quello che è il mondo dell'informatica, nel dettaglio questi sono i principali vantaggi:

- Abbattimento dei costi – L'avvento della tecnologia cloud ha introdotto all'interno del mercato la possibilità di pagare una risorsa per il tempo esatto in cui viene utilizzata. Il cliente è dunque in grado di acquistare un determinato servizio esattamente per il tempo necessario, rilasciandolo al termine. L'introduzione di questa nuova metodologia di pagamento è dovuta all'estrema velocità con cui è possibile allocare e successivamente

ridistribuire una risorsa all'interno di una infrastruttura cloud cosa che, in una architettura tradizionale, non era sostenibile.

- Indipendenza dalla locazione – L'accesso ad una risorsa sul cloud è estremamente semplice ed indipendente da dove il cliente si trova: è necessario avere una connessione ad internet per poter accedere, ovunque ci si trovi, alle nostre risorse in maniera affidabile ed avere disponibili i nostri dati e servizi.
- Virtualizzazione – Tramite la virtualizzazione, caratteristica fondamentale del cloud, ho la possibilità di modificare, aggiungere o togliere risorse ogni volta che ne ho bisogno e senza dover affrontare alcuna instabilità del sistema poiché l'assegnazione avviene in maniera del tutto automatica. E' importante anche notare che questa caratteristica ci permette di clonare, modificare e migrare le macchine virtuali da un host ad un altro senza dover interrompere la fornitura del servizio: ciò risulta molto utile nel caso di servizi critici per un'azienda che devono risultare sempre accessibili.
- Affidabilità – Se una infrastruttura di cloud computing è bel progettata un punto a favore di questa soluzione si trova appunto nella sua estrema affidabilità: introdurre meccanismi di ridondanza dei dati e delle connessioni garantisce le risorse anche nel caso in cui ci siano problemi alla server farm ospitante. Poiché infatti i dati e i servizi sono replicati in più datacenter l'eventuale isolamento di uno di essi non influisce sulla stabilità del servizio.
- Presenza di API – Generalmente le strutture cloud attuali forniscono all'utilizzatore un'interfaccia di programmazione API con la quale il software locale al pc dell'utente è in grado di comunicare con il cloud in maniera trasparente ed efficace.

Da un altro punto di vista una tecnologia di questo tipo introduce tutta una serie di problematiche non indifferenti, che vanno analizzate e valutate prima di fornire un servizio di questo tipo:

- Privacy – Essendo un servizio non più centralizzato all'interno dell'azienda, ma distribuito all'esterno, è possibile andare incontro a delle problematiche dal punto di vista della privacy poiché documenti riservati e

altri files vengono memorizzati su server di proprietà del provider, spesso in stati diversi da quello del cliente, che sono quindi soggetti a differenti politiche di privacy.

- Stabilità del servizio – Decentralizzando i servizi di un'azienda e portando il tutto su una soluzione cloud in caso di una interruzione del servizio o della connessione ad internet risulterà impossibile poter accedere ai propri dati. Pur garantendo alti margini di affidabilità e ridondanza, rispetto ad una struttura tradizionale, anche una soluzione cloud non risulta comunque immune da imprevisti.

2.4. Principali tecnologie Cloud a confronto

2.4.1. XCP

XCP [9], acronimo di Xen Cloud Platform, è una tecnologia di virtualizzazione open source che permette all'utente di creare una propria piattaforma di cloud computing con estrema semplicità. E' basato sull'hypervisor di Xen e fornisce delle API per l'utilizzo enterprise chiamate Xapi con le quali è possibile interagire con il cloud ed allocare le risorse tramite alcune chiamate software.

2.4.2. VMWare vSphere

VMWare vSphere [10] è il software di virtualizzazione dedicato alle imprese prodotto dalla nota VmWare, leader nel campo nella virtualizzazione a livello enterprise. Si tratta di un prodotto proprietario con alti costi di licenza ma che garantisce performance elevate. vSphere punta soprattutto ad automatizzare il più possibile l'infrastruttura cloud in base alle esigenze del cliente, definendo delle politiche personalizzate di gestione e controllo.

2.4.3. Openstack

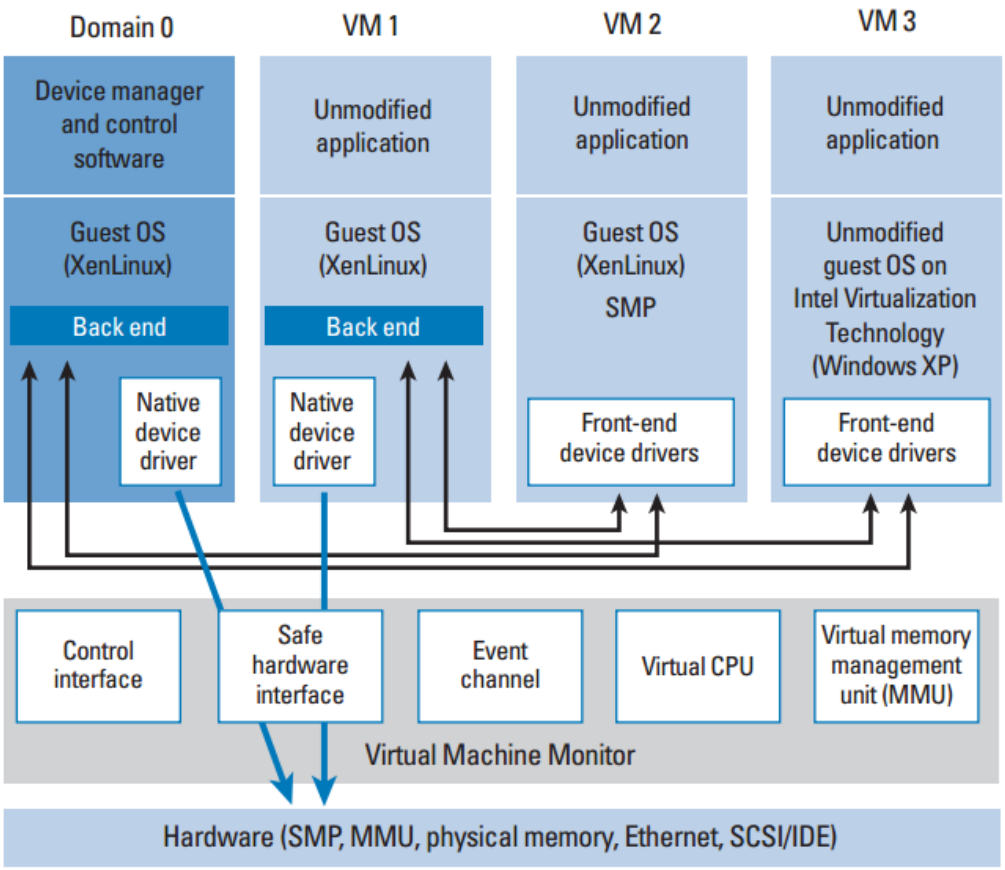
Openstack [11] è un progetto di cloud computing open-source che raccoglie al suo interno un insieme di applicativi per gestire con efficienza pool di dimensioni notevoli. Ogni applicativo si occupa di gestire una parte del pool (elaborazione, connessione, dati, ecc) dando la possibilità all'amministratore di monitorare il tutto tramite una interfaccia grafica molto intuitiva e semplice da utilizzare. Questa

architettura modulare di Openstack e l'estrema semplicità di configurazione e gestione ne hanno causato l'estremo successo negli ultimi anni.

3. Xen

3.1. Introduzione a Xen

Come già accennato precedentemente Xen è un cosiddetto Virtual Machine Monitor (o Hypervisor) che permette a più sistemi operativi di funzionare su un unico sistema fisico condividendo delle risorse senza però dover sacrificare le performance. Xen utilizza la tecnica della paravirtualizzazione per evitare gli svantaggi di una virtualizzazione tradizionale, mostrando quindi alla macchina virtuale un hardware simile, ma non identico, a quello dell'host.



2. Virtualizzazione in Xen

Fonte: http://www.dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf

Nella figura sopra riportata (Figura 2) abbiamo un esempio di un caso d'uso concreto di Xen: come è possibile vedere il dominio zero (Dom0) è l'unico ad avere accesso all'interfaccia del Virtual Machine Monitor (VMM) attraverso la quale è possibile creare, modificare o distruggere le altre macchine virtuali. L'amministratore è inoltre in grado di creare macchine virtuali privilegiate – come VM1 – che sono in grado di accedere alla stessa interfaccia di controllo del Dom0.

3.2. Meccanismi di virtualizzazione: analisi approfondita delle caratteristiche di Xen

Per questa analisi focalizzeremo la nostra attenzione su tre principali componenti e vedremo come questi vengono presentati alla macchina virtuale dall'hypervisor [12]:

- Gestione della memoria – La gestione della memoria costituisce probabilmente una delle parti più complesse ed articolate di un software di virtualizzazione. Questo compito tuttavia risulta molto più semplice se l'architettura sottostante è in grado di fornire un TLB software (Translation Lookaside Buffer) in grado di gestire in maniera efficiente la traslazione delle pagine da indirizzo software a indirizzo fisico. Poiché, considerando la molteplicità delle macchine virtualizzate, risulta impossibile mantenere l'intera tabella di traslazione in memoria questa viene svuotata e popolata nuovamente ogni volta che l'esecuzione passa da una macchina virtuale a quella successiva. Sono quindi i sistemi operativi guest a doversi occupare dell'allocazione e della gestione della tabella TLB, evitando un coinvolgimento da parte dell'hypervisor, e garantendo stabilità ed isolamento. La gestione della memoria dell'hypervisor viene mantenuta negli ultimi 64MB della tabella evitando che venga svuotata ad ogni cambio di esecuzione. In questo modo ogni volta che un sistema guest necessita di creare una nuova pagina prima provvede alla sua allocazione e poi informa l'hypervisor: in questo modo Xen è in grado di registrare la macchine virtuale alla quale appartengono le pagine in memoria ed è in grado di impedirne l'accesso in caso di tentativi di lettura o scrittura non autorizzati. In caso di segmentazione il meccanismo è analogo, ma prevede la creazione di segmenti con privilegi più bassi rispetto a quelli dell'hypervisor e ovviamente impedisce a qualsiasi guest di accedere alla porzione di indirizzi dedicata a Xen.
- CPU – L'inserimento di un hypervisor al di sotto dei sistemi operativi delle macchine guest viola uno dei principi della computazione che indica il sistema operativo come entità più privilegiata del sistema. In un sistema virtualizzato è infatti necessario declassare il sistema operativo dei guest

ad un livello meno privilegiato al fine di mantenere l'hypervisor ad un livello più alto. Nei sistemi ad architettura x86 ad esempio, avendo quattro distinti livelli, è logico pensare che tradizionalmente il sistema operativo venga eseguito all'interno del livello zero (il più privilegiato) mentre le applicazioni vengono eseguite nel livello tre (meno privilegiato), i livelli uno e due generalmente non vengono utilizzati. In un sistema virtualizzato tramite Xen invece è l'hypervisor ad essere in esecuzione nel livello zero, il sistema operativo guest nel livello uno e le sue applicazioni nel livello tre. Per permettere l'esecuzione di un sistema operativo in un ambiente di virtualizzazione Xen è dunque necessario modificarlo affinchè possa esser eseguito all'interno del livello uno ed istruirlo in modo da permettergli di inoltrare tutte le richieste privilegiate all'hypervisor nel livello zero.

- Dispositivi di Input/Output – Xen fornisce alla macchina virtuale un'emulazione dell'hardware, fornendo un dispositivo fittizio sul quale il sistema operativo della macchina guest esegue le proprie richieste. Le richieste effettuate a questo dispositivo vengono inoltrate all'hypervisor Xen che esegue le operazioni richieste ponendo i risultati in memoria. Terminato il trasferimento dei dati l'hypervisor, tramite un meccanismo ad eventi, segnala alla macchina virtuale corrispondente il completamento dell'operazione e, attraverso un meccanismo a memoria condivisa, inoltra i dati ricevuti alla macchina guest.

3.3. XenServer - La versione Enterprise

XenServer [13] è la prima soluzione commerciale offerta da Citrix [14] dopo l'acquisizione del progetto Xen: sono state sviluppate in parallelo una soluzione open-source (Xen) e una closed-source (XenServer). XenServer offre una versione gratuita molto limitata nell'utilizzo ed estendibile tramite il pagamento di alcune licenze software, si differenzia inoltre rispetto alla soluzione tradizionale Xen per la sua notevole semplicità d'uso e configurazione. Quando infatti parliamo di Xen si intende il solo hypervisor senza comprendere alcuna console

per la gestione o funzioni extra, al contrario quando citiamo XenServer comprendiamo anche tutta la suite che permette la gestione dell'infrastruttura.

3.4. XCP – Xen Cloud Platform

3.4.1. Introduzione ad XCP

Xen Cloud Platform è una soluzione open-source che prevede una piattaforma pronta all'uso di virtualizzazione e cloud computing, infatti include al suo interno un insieme di applicativi per la gestione di strutture di storage e rete distribuite. Sono disponibili una molteplicità di interfacce grafiche, sia web che desktop, per permettere un'efficiente amministrazione del cloud ed inoltre, per permettere configurazioni più a basso livello, è disponibile anche una comoda interfaccia a riga di comando.

3.4.2. Caratteristiche aggiuntive di XCP rispetto alla soluzione Xen standard

Come precedentemente anticipato, rispetto ad una soluzione Xen standard, XCP prevede tutta una serie di caratteristiche aggiuntive orientate al cloud:

- Gestione avanzata delle macchine virtuali – In particolare è data la possibilità di agire in maniera diretta sugli stati delle macchine virtuali creando snapshot o checkpoint. Inoltre un'altra importante caratteristica di questo programma è la possibilità di migrare una macchina virtuale da un host fisico all'altro senza avere la necessità di spegnerla, riducendo quindi al minimo ogni possibile interruzione del servizio.
- Gestione delle risorse – XCP permette di avere storage e rete distribuiti, è quindi probabile che all'interno di un pool XCP i dischi delle macchine virtuali non si trovino all'interno dell'host fisico su cui sono in esecuzione, ma piuttosto in uno storage esterno collegato via rete (NFS o iSCSI). E' inoltre possibile creare delle infrastrutture di rete personalizzate che connettano tra di loro due o più macchine virtuali come se fossero fisicamente collegate.
- Tracciamento degli eventi – Alcuni eventi possono essere monitorati dall'amministratore: è possibile vederne il progresso o ricevere una

notifica al loro verificarsi. Questo può risultare utile nel caso in cui si effettui un monitoraggio avanzato dell'infrastruttura.

- Avanzamenti di versione – E' possibile effettuare in maniera trasparente avanzamenti di versione o applicare delle patch.
- Monitoraggio performance – XCP è in grado di fornire all'amministrazione una visione dell'interno parco macchine in tempo reale, fornendo informazioni complete e dettagliate.
- Supporto per Windows – Pur essendo principalmente un software di paravirtualizzazione, XCP fornisce supporto anche per la virtualizzazione assistita, permettendo di creare macchine virtuali Windows tramite l'utilizzo di estensioni hardware della CPU.
- Supporto Open vSwitch – XCP permette l'installazione di un programma dedicato al networking, detto Open vSwitch, che semplifica l'amministrazione della rete virtuale fornendo meccanismi molto elaborati di automazione e monitoraggio. In particolare Open vSwitch è dedicato alle situazioni in cui la topologia della rete può cambiare molto frequentemente, come appunto nel caso del cloud, dove le macchine virtuali possono essere aggiunte e rimosse molto velocemente, garantendo migliori performance e strategie di routing molto performanti.

3.4.2.1. XenMotion e LiveMigration

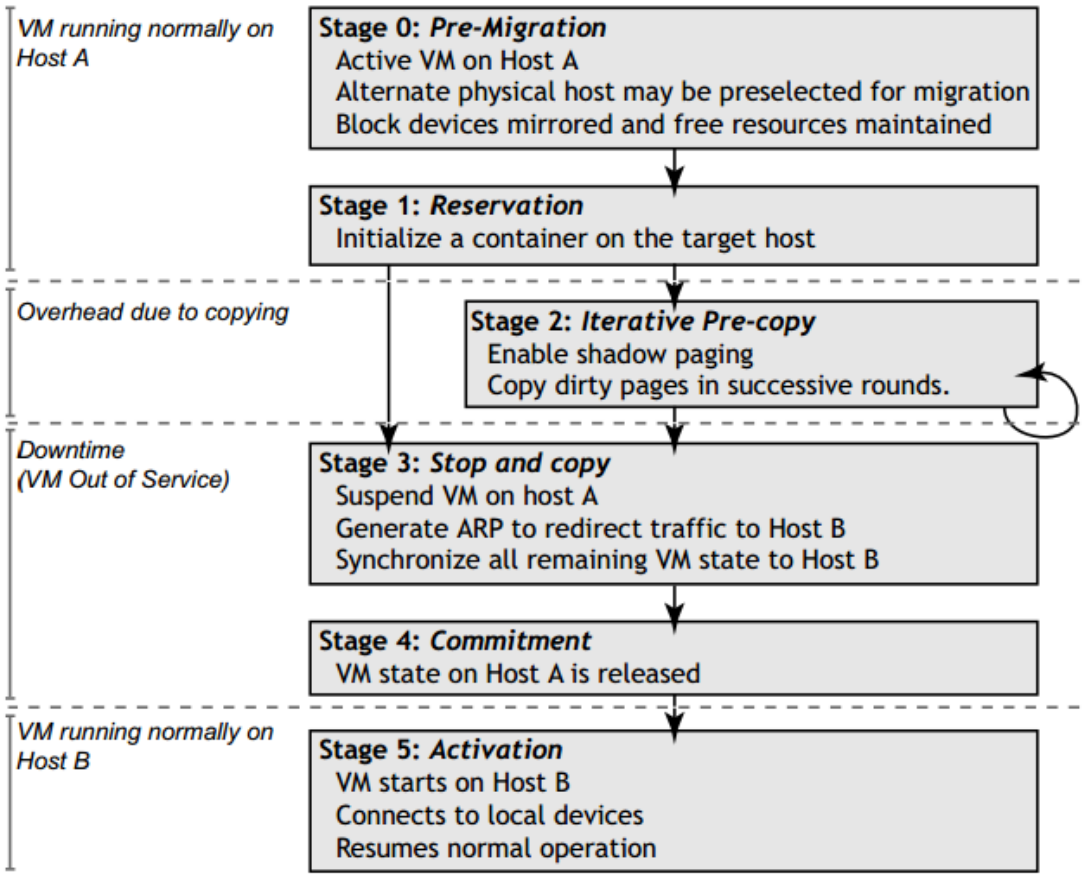
Un'importante caratteristica aggiunta a XCP è la cosiddetta XenMotion [15], che consente ad una macchina virtuale in esecuzione di migrare da un host fisico all'altro senza alcuna interruzione del servizio. I requisiti per una migrazione di questo tipo sono:

- Avere almeno due host all'interno dello stesso pool.
- Gli host devono avere tutti processori simili, dello stesso venditore, famiglia e modello.
- Uno storage condiviso per permettere ai dischi virtuali delle macchine di essere accessibili da entrambi gli host coinvolti nella migrazione. Nel

dettaglio XCP attualmente supporta: condivisioni NFS, SAN iSCSI, SAN in fibra.

- Connettività Gigabit preferibile, ma non necessaria per macchine di piccolo volume

Una migrazione provoca generalmente una interruzione di 100-150 millisecondi che non influisce sulla stabilità del servizio offerto; la maggior parte di questo tempo è impiegato per aggiornare le regole di routing, instradando i pacchetti in arrivo verso il nuovo host.



3. Migrazione live di macchine virtuali

Fonte: Università di Cambridge http://www.cl.cam.ac.uk/

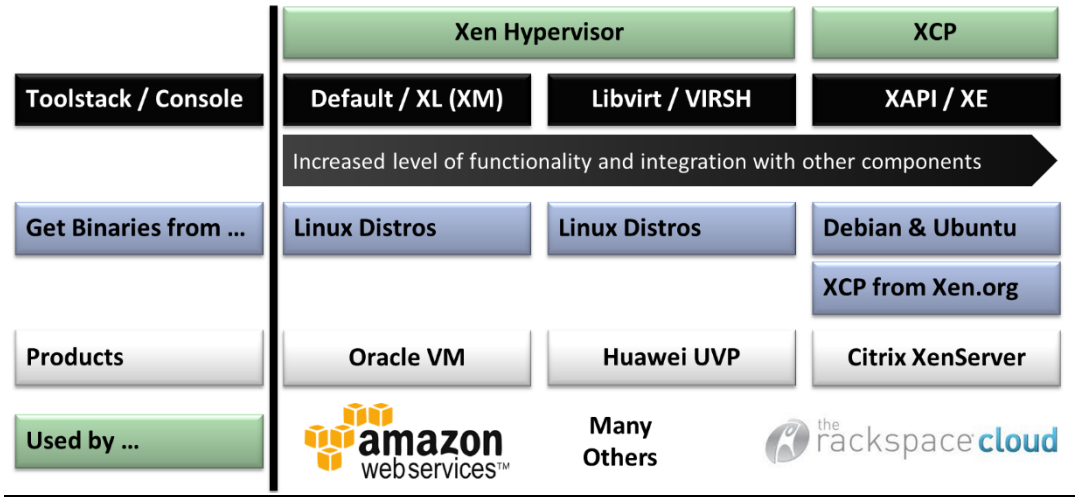
Il procedimento di migrazione illustrato (Figura 3) ha sei fasi principali [16]:

- Passo 0: Pre-Migrazione – Prima di iniziare la migrazione sarà presente una macchine virtuale su un Host A mentre un Host B verrà selezionato come destinatario. E' possibile definire un host come predefinito per una

migrazione, in modo che le risorse su questo vengano sempre garantite per una possibile migrazione.

- Passo 1: Allocazione – Sull'host di destinazione viene riservato lo spazio per accogliere la macchina virtuale, nel caso ciò non fosse possibile la migrazione viene sospesa e la macchina guest rimane in esecuzione sull'host di partenza.
- Passo 2: Pre-Copia iterativa – Durante questa fase tutta la memoria occupata dalla macchina virtuale nell'host A viene copiata verso l'host B. La macchina prosegue la sua esecuzione nell'host A e le pagine modificate dall'inizio della copia continuano ad essere inviate al nodo di destinazione per sincronizzare la memoria tra le due macchine con il minimo errore possibile.
- Passo 3: Sospensione e copia – Questo passo è uno dei più critici tra quelli presenti poiché sospende la macchina virtuale sull'host A, genera un pacchetto ARP per instradare i pacchetti sul nuovo host e termina la sincronizzazione della memoria. Sospendendo la macchina la memoria rimane immutata e quindi si è in grado di creare una copia completa dello stato attuale della macchina, in questo modo al termine di questa fase entrambi gli host conterranno una copia della memoria della macchina virtuale in fase di migrazione. E' importante sottolineare che, durante la migrazione, la maggior parte del tempo in cui risulta impossibile raggiungere la macchina virtuale dall'esterno (downtime) è dovuto a causa della definizione delle nuove regole di routing, motivo per cui il pacchetto ARP viene inviato in uno dei primi passi.
- Passo 4: Invio – L'host B segnala ad A la corretta ricezione della memoria della macchina virtuale. Una volta ricevuta questa conferma l'host A rilascia la macchina virtuale e B diventa l'host primario.
- Passo 5: Attivazione – La macchina virtuale viene avviata sull'host B, vengono collegati nuovamente tutti i dispositivi virtuali presenti e l'indirizzo ip della macchina viene assegnato nuovamente.

3.4.3. Struttura di XCP



4. Xen Toolstack

Fonte: http://wiki.xen.org/wiki/Xen_Overview

Quando si parla di Xen Cloud Platform si intende sempre una piattaforma di virtualizzazione Xen integrata con dei programmi di manutenzione e configurazione (Toolstacks) appartenenti ad un unico pacchetto chiamato Xapi. Come è possibile vedere (Figura 4) l'hypervisor Xen risulta altamente portabile e configurabile con diverse soluzioni software di gestione a seconda dell'utilizzo che bisogna farne. Facendo quindi riferimento all'immagine possiamo quindi dire che Xen può essere utilizzato in diverse modalità: partendo dalla soluzione a sinistra abbiamo una serie di applicativi che vengono installati nel caso di una installazione di default (XL/XM) permettendo una configurazione altamente personalizzabile, pur rendendo la gestione dell'intera infrastruttura molto più complessa. Al centro abbiamo invece un'installazione con supporto Libvirt (VIRSH), ovvero viene aggiunta all'hypervisor una libreria esterna dedicata alla configurazione ed installazione della piattaforma: una soluzione di questo tipo si trova a metà tra un utilizzo di alto livello (Xapi) e uno base. L'ultima soluzione, quella più a destra, risulta più completa e fornisce un software di supporto molto di alto livello, garantendo comunque un'alta configurabilità e personalizzazione dell'infrastruttura. Quest'ultima soluzione in cui l'hypervisor Xen collabora con Xapi viene definita nel complesso Xen Cloud Platform e viene fornita, nella maggior parte delle distribuzioni linux, come un unico pacchetto d'installazione.

4. Monitoraggio e bilanciamento del carico con XCP

4.1. Illustrazione del lavoro di tesi e modello implementativo ideato

L'obiettivo di questo lavoro di tesi è quello di ideare e realizzare un'applicazione che permetta il monitoraggio e il bilanciamento di carico all'interno di un pool XCP, fornendo all'amministratore dell'infrastruttura la possibilità di avere in ogni istante la situazione complessiva del pool e definire regole di bilanciamento personalizzate. In base alle regole definite dall'amministratore ed al carico degli host il programma decide autonomamente quando e come è necessario bilanciare le risorse, migrando le macchine virtuali da un'host all'altro secondo una logica definita.

4.2. Configurazione dell'ambiente di test

4.2.1. Hardware utilizzato

Per creare un ambiente di test consono allo sviluppo della nostra applicazione è stata create una rete dedicata composta da tre macchine Dell Optiplex 390 con le seguenti caratteristiche:

- Scheda Madre - Motherboard Dell 0M5DCD
- Processore - Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz Quad-Core 64Bit
- Ram - 4GB - DIMM DDR3 Synchronous 1333 MHz (0,8 ns)
- Disco Rigido - Seagate KC45 500GB

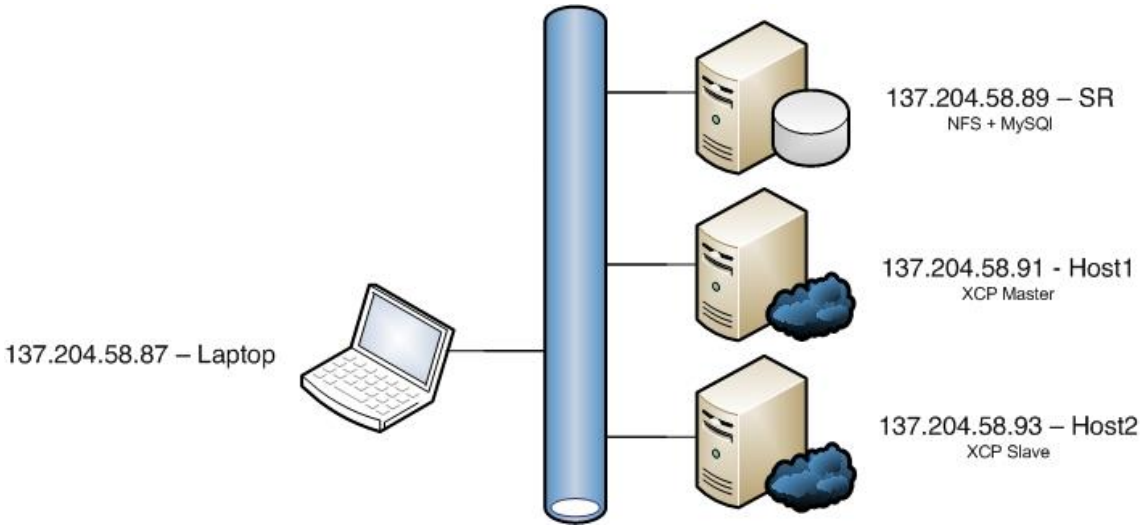
Come sistema operativo la scelta è ricaduta sulla distribuzione Debian Wheezy 7.0 (testing) per amd64, in quanto la versione stabile 6.0.7 non fornisce attualmente pacchetti di installazione e supporto per XCP. Inoltre, trattandosi di host con processore a 64bit la scelta si è indirizzata ovviamente sulla versione corrispondente del sistema operativo.

Per connettere tra di loro le unità di rete è stato utilizzato un HP ProCurve Switch 2324 (J4818A) che fornisce:

- Porte di connessione - 24 porte RJ-45 supportando velocità di 10/100 Mbit/s
- Capacità di routing/switching – fino a 9.6 Gbps

- Processore - Tipologia ARM7TDMI @ 62.5 MHz
- Buffer di trasmissione – 6 MB

4.2.2. Struttura della rete



5. Struttura della rete

L'immagine sopra riportata (Figura 5) mostra la struttura dell'ambiente di test creato, nel dettaglio sono stati assegnate le seguenti coppie di ip-hostname:

- 137.204.58.87 – Laptop: Utilizzato per lo sviluppo ed il testing del programma
- 137.204.58.89 – SR: Acronimo di Shared Resources, macchina dedicata a funzioni di storage condiviso NFS (contenente le iso di installazione e i dischi virtuali delle macchine) e database Mysql.
- 137.204.58.91 - Host1: Primo host XCP, master del pool
- 137.204.58.93 - Host2: Secondo host XCP, slave del pool

4.2.3. Configurazione Shared Storage

Come già specificato in precedenza la macchina SR contiene al suo interno un server NFS [17] per condividere due cartelle (ISO e DISK) e un server Mysql [18]. La cartella ISO conterrà all'interno le immagini disco di installazione dei sistemi operativi, mentre la cartella DISK avrà al suo interno i dischi virtuali di tutte la

macchine virtuali presenti nel pool. Qui di seguito riportiamo i passi necessari all'installazione.

- Vengono create prima di tutto due cartelle, ognuna delle quali rappresenta una risorsa da condividere:

```
mkdir /storage
mkdir /storage/DISK
mkdir /storage/ISO
```

- Installazione del server NFS tramite aptitude

```
apt-get install nfs-server
```

- Aggiungo queste linee al file */etc/exports* per abilitare la condivisione delle due cartelle:

```
/storage/ISO      137.204.58.0/24(rw,sync)
/storage/DISK     137.204.58.0/24(rw,sync)
```

Nel dettaglio abbiamo impostato la condivisione di due cartelle, indicando al demone NFS che queste sono disponibili in lettura e scrittura per tutta la sottorete 137.204.58.0/24 (ovviamente in un ambiente di produzione è bene applicare regole più restrittive) e che è richiesto un export di tipo sync, riducendo la possibilità di avere una corruzione dei dati nel caso di un malfunzionamento del server.

- Riavvio il server tramite il comando:

```
service nfs-kernel-server restart
```

- Installazione del server Mysql:

```
apt-get install mysql-server
```

- Modifica del file */etc/mysql/my.cnf* per permettere connessioni dall'esterno impostando:

```
bind-address = 0.0.0.0
```

- Riavvio del server mysql:

```
service mysql restart
```

A questo punto la configurazione del nostro server SR è completata, nel prossimo paragrafo viene illustrata un procedimento opzionale, non necessario all'esecuzione di XCP, ma che ci permetterà di avere delle statistiche sull'uso del server installando un semplice sistema di monitoraggio centralizzato.

4.2.4. Configurazione sistema di Monitoring

La scelta di un sistema di monitoraggio adatto ai nostri scopi è risultata tutt'altro che semplice: molto sistemi attualmente sul mercato hanno intervalli di polling troppo ampi (ad esempio Munin [19] ha un minimo di 5 minuti). Nel nostro caso avevamo bisogno di un sistema che ci permettesse di fare misurazioni e di avere grafici con tempi molto più brevi. La nostra scelta è dunque ricaduta su Collectd [20], un sistema di monitoraggio molto leggero e personalizzabile, in grado di fornirci intervalli molto più brevi evitando di sovraccaricare il server nonostante l'elevata frequenza delle richieste. L'applicativo verrà installato sul server SR (in modalità Master) e su tutti gli host (in modalità Slave); ad intervalli regolari, tutti gli host provvederanno ad inviare ad SR il loro stato. Per l'installazione procediamo in questa maniera:

- Installazione di collectd su tutti i server:

  ```
  apt-get install collectd
  ```

- Su SR viene creato un file di configurazione */etc/collectd/auth_file* contenente le coppie di nome utente e password che useranno gli host per autenticarsi. Il file dovrà avere questo formato:

  ```
  user0: pass0
  user1: pass1
  ```

- Il file di configurazione */etc/collectd/collectd.conf* deve essere modificato in questo modo:

  Per SR:

  ```
  Hostname "SR"
  Interval 10
  LoadPlugin syslog
  <Plugin syslog>
        LogLevel info
  </Plugin>
  LoadPlugin cpu
  ```

```
LoadPlugin df
LoadPlugin disk
LoadPlugin entropy
LoadPlugin interface
LoadPlugin irq
LoadPlugin load
LoadPlugin memory
LoadPlugin mysql
LoadPlugin network
LoadPlugin nfs
LoadPlugin processes
LoadPlugin rrdtool
LoadPlugin swap
LoadPlugin users
<Plugin interface>
      Interface "eth0"
      IgnoreSelected false
</Plugin>
<Plugin mysql>
      <Database Xen>
            Host "localhost"
            Port "3306"
            User "mysqlUser"
            Password "mysqlPassword"
            Database "Xen"
            MasterStats true
      </Database>
</Plugin>
<Plugin network>
      <Listen "137.204.58.89" "25826">
            SecurityLevel "Sign"
            AuthFile "/etc/collectd/auth_file"
      </Listen>
</Plugin>
<Plugin rrdtool>
      DataDir "/var/lib/collectd/rrd"
</Plugin>
Include "/etc/collectd/filters.conf"
Include "/etc/collectd/thresholds.conf"
```

Per Host1 e Host2:

```
Hostname "nomeHost"
Interval 10
LoadPlugin syslog
<Plugin syslog>
      LogLevel info
</Plugin>
LoadPlugin battery
LoadPlugin cpu
LoadPlugin df
LoadPlugin disk
LoadPlugin entropy
LoadPlugin interface
LoadPlugin irq
LoadPlugin load
LoadPlugin memory
LoadPlugin network
LoadPlugin nfs
LoadPlugin processes
```

```
LoadPlugin rrdtool
LoadPlugin swap
LoadPlugin users
<Plugin network>
      <Server "137.204.58.89" "25826">
            SecurityLevel "Sign"
            Username "hostUser"
            Password "hostPassword"
      </Server>
</Plugin>
<Plugin rrdtool>
      DataDir "/var/lib/collectd/rrd"
</Plugin>
Include "/etc/collectd/filters.conf"
Include "/etc/collectd/thresholds.conf"
```

- Infine il demone collectd viene riavviato su ognuno dei server con il comando:

```
/etc/init.d/collectd restart
```

A questo punto il nostro sistema di monitoraggio risulta attivo e funzionante ed inizierà a raccogliere lo stato di tutti gli host ad intervalli di 10 secondi. Tramite uno script cgi fornito con il pacchetto collectd, modificato all'occorrenza, è stato possibile ottenere grafici su tempi molto più ridotti, permettendo di fare un'analisi molto dettagliata sullo stato del pool.

4.2.5. Configurazione degli Host XCP

La configurazione degli host XCP è risultata più complessa del previsto in quanto il pacchetto non è esente da errori di compilazione e configurazioni errate. Inoltre è necessario effettuare prima un'installazione separata di Xen e, successivamente, di Xapi per poter avere un host XCP funzionante. Ecco come è proceduta l'installazione e la configurazione dei due host:

- Viene modificato il file */etc/hosts* aggiungendo le seguenti linee:

```
137.204.58.89   SR
137.204.58.91   Host1
137.204.58.93   Host2
```

- Si creano due cartelle per effettuare l'export dal server NFS, per comodità verranno chiamate allo stesso modo:

```
mkdir /storage
mkdir /storage/DISK
mkdir /storage/ISO
```

- Vengono aggiunte al file /etc/fstab le seguenti linee, in questo modo le due directory NFS verranno montate automaticamente all'avvio:

```
SR:/storage/ISO /storage/ISO  nfs   defaults    0     0
SR:/storage/DISK /storage/DISK nfs  defaults    0     0
```

- Montaggio delle directory appena configurate:

```
mount -a
```

- Poiché la scheda di rete richiede driver software proprietari è necessario provvedere ad una installazione manuale, andando a modificare il file */etc/apt/sources.list* ed aggiungendo ad ogni linea l'opzione:

```
non-free
```

- Dopo di che si provvede all'aggiornamento e all'installazione del pacchetto:

```
apt-get update
apt-get install firmware-realtek
```

- Vengono installati i pacchetti necessari all'avvio dell'hypervisor Xen:

```
apt-get install xen-linux-system-amd64 xen-utils-4.1 xen-
tools xen-hypervisor-4.1-amd64
```

- E successivamente il pacchetto per XCP/Xapi:

```
apt-get install xcp-xapi
```

- A questo punto, evitando di riavviare, è necessario modificare l'installazione di Xen affinchè possa collaborare con XCP/Xapi. Per far ciò bisogna aprire il file */etc/init.d/xen* e commentare le linee con scritto:

```
xend_start
xend_stop
```

- E disabilitare xendomains dall'avvio:

```
update-rc.d xendomains disable
```

- Installazione di Qemu per l'emulazione di processori e creazione link simbolico:

```
apt-get install qemu
ln -s /usr/share/qemu-linaro/keymaps
/usr/share/qemu/keymaps
```

- Deve essere inoltre modificato il file */etc/default/grub* rendendo il kernel Xen quello di default e aggiornando grub tramite il comando:

```
update-grub
```

- Viene modificato il file */etc/default/xen* inserendo:

```
TOOLSTACK=xapi
```

- E vengono ridefinite le interfacce di rete all'interno del file */etc/network/interfaces* nel seguente modo:

```
auto lo
iface lo inet loopback

auto xenbr0
      iface xenbr0 inet static
      bridge_ports    eth0
      address 137.204.58.91
      netmask 255.255.255.0
      network 137.204.58.0
      broadcast 137.204.58.255
      gateway 137.204.58.254
      dns-nameservers 137.204.58.4
      dns-search ing.unibo.it
```

Ovviamente è opportuno modificare le informazioni dei file di configurazione a seconda della struttura della rete, adeguandole alla propria architettura di rete.

Una volta riavviato dovremo avere due host XCP funzionanti, successivamente dovremo collegarci tramite XenCenter [21] ed ultimare la configurazione creando un nuovo pool con i due host appena configurati, definendo Host1 come Master e Host2 come slave. Successivamente, sempre da XenCenter, sarà necessario definire un repository per i dischi virtuali (/storage/DISK) ed uno per le immagini ISO utilizzabili dalle macchine virtuali (/storage/ISO).

4.2.6. Difficoltà rilevate

Il processo di installazione è risultato particolarmente complesso in più di un passaggio, soprattutto per la necessità di trovare soluzione ad alcune problematiche riportate durante la configurazione dei pacchetti. Oltre alle difficoltà di installazione sono state inoltre osservate le seguenti problematiche:

- Installazione macchine virtuali - Il procedimento di installazione delle VM risulta abbastanza lento e, durante questo tempo, il carico dello storage condiviso è risultato molto elevato: ciò rende impossibile l'installazione di più macchine virtuali contemporaneamente. La causa di questo fenomeno è stata analizzata e deve essere ricercata nella bassa velocità del collegamento tra le macchine: le numerose operazioni di I/O eseguite attraverso un collegamento a 100Mbit e agenti tutte sullo stesso disco comportano un rallentamento notevole del processo di installazione. L'utilizzo di un collegamento Gigabit, come consigliato nelle specifiche, e di dischi più performanti o in RAID porterebbero sicuramente ad un notevole aumento delle prestazioni nell'installazione e nell'esecuzione delle macchine virtuali.
- Migrazione macchine virtuali – Durante numerosi test di migrazione effettuati nell'ambiente di test è stato rilevato che successivamente alla migrazione è necessario un periodo di assestamento per permettere all'host di fornire informazioni accurate sulle macchine virtuali. Dopo la migrazione è capitato frequentemente, soprattutto con macchine di dimensioni elevate, che per un determinato periodo di tempo le chiamate API e lo stesso XenCenter fornissero informazioni errate sulla macchina virtuale appena migrata. Mentre in alcuni casi era possibile rilevare l'errore, poiché l'host rispondeva con una eccezione indicando che effettivamente la macchina era in fase di migrazione e quindi le informazioni erano temporaneamente non disponibili, in altri casi venivano riportati risultati plausibili, senza sollevare eccezioni, aggiungendo ulteriore complessità al programma.
- Logging di Xapi – Di default, all'installazione del pacchetto, il livello di logging definito nei file di configurazione è impostato a "debug". Ciò

comporta una crescita smisurata del file xcp-xapi.log all'interno di /var/log che ha causato, in più di un'occasione, un blocco del sistema. Si ritiene che questo problema verrà corretto in un prossimo rilascio poiché avere log molto verbosi dovrebbe essere una scelta dell'utente, e non un comportamento di default.

4.3. SDK Xen

4.3.1. Descrizione del protocollo Xapi

Il protocollo di comunicazione Xapi [22] si basa su chiamate XML-RPC [23] che forniscono l'accesso a funzioni di controllo e configurazione del pool a cui fanno riferimento. Inoltre tali chiamate possono essere invocate, previa autenticazione, sia da un host remoto che dall'host stesso. Per uno sviluppatore è possibile creare applicazioni che sfruttino tali API tramite due strade: la prima prevede delle chiamate dirette XML-RPC sull'host Xapi, tale pratica è sconsigliata a causa della complessità di uso di tali funzioni che introdurrebbero una mole di codice notevole e non necessaria. L'alternativa consigliata prevede l'utilizzo delle SDK Xen: un insieme di librerie, fornite per numerosi linguaggi, che danno la possibilità al programmatore di eseguire funzioni e metodi più di alto livello e lasciare alla libreria il compito di effettuare le chiamate RPC. In questo modo il codice che si ottiene risulta molto meno complesso e più efficiente. Attualmente sono disponibili le librerie per i seguenti linguaggi: C, C#, Java, Python e PowerShell. Nell'ambito di questa tesi si è deciso di utilizzare il linguaggio Java. Le API Xen sono caratterizzate da:

- Possibilità di controllare ogni aspetto di un host XCP – Tramite le API è possibile gestire autonomamente qualsiasi componente di un host: macchine virtuali, interfacce di rete e dischi fissi possono essere aggiunte e rimosse tramite chiamate specifiche. Inoltre è possibile ottenere informazioni anche sullo stato e il carico dell'host.
- Invocazioni sincrone e asincrone – La maggior parte delle chiamate API, soprattutto quelle che necessitano tempi lunghi per essere completate, possono essere invocate in due modalità differenti: sincrona e asincrona. Come risulta facilmente intuibile dal nome, le chiamate sincrone

interrompono l'esecuzione del programma fino al completamento, mentre quelle asincrone avviano l'operazione senza sospendere il programma, restituendo un oggetto di tipo Task in grado di riportarci la percentuale di completamento e lo stato della chiamata.

- Chiamate remote – Come già precedentemente accennato non è necessario che il software risieda sullo stesso host che fornisce le API, poiché è possibile effettuare chiamate da remoto tramite autenticazione. Inoltre non è nemmeno necessario avere un accesso ssh alla macchina.
- Accesso sicuro e autenticato – Oltre all'autenticazione, obbligatoria per eseguire chiamate API, Xapi permette anche chiamate XML-RPC sicure tramite il protocollo https (SSL).

4.3.2. Illustrazione dei metodi principali utilizzati

Prima di iniziare la descrizione del progetto di tesi è necessario fare una introduzione sugli oggetti e sui metodi principali forniti da Xapi e fornire un'idea di come questi oggetti e metodi possano collaborare tra loro. Nel dettaglio gli oggetti principali sono:

- VM – Un oggetto VM rappresenta una istanza di una macchina virtuale all'interno di un host o di un pool XCP. E' importante notare che oggetti di questo tipo comprendono sia macchine virtuali che template. A questi oggetti sono applicabili metodi come:

```
start
suspend
pool_migrate
```

- Host – Un oggetto Host identifica un host fisico all'interno di un pool XCP. A questi oggetti sono applicabili metodi come:

```
reboot
shutdown
```

- VDI – Un oggetto VDI, acronimo di Virtual Disk Image, rappresenta un disco rigido virtuale che può essere assegnato o rimosso da una macchina virtuale.

- SR – Un oggetto SR (Storage Repository) contiene al suo interno un insieme di dischi virtuali VDI e le informazioni sui server che contengono tali dischi.
- Network – Un oggetto Network rappresenta appunto la rete all'interno del quale è presente il pool XCP. Ogni Host e VM sono collegati ad uno o più oggetti Network

Sono inoltre presenti altri quattro oggetti speciali che fungono da connettori, specificando le relazioni tra VM, Host, SR e Network. Possiamo descriverli come:

- VBD – Un oggetto VBD (Virtual Block Device) rappresenta un collegamento tra una macchina virtuale (VM) e il suo disco virtuale (VDI). Quando una macchina virtuale viene avviata i suoi oggetti VBD vengono interrogati per capire quali immagini disco VDI devono essere connesse.

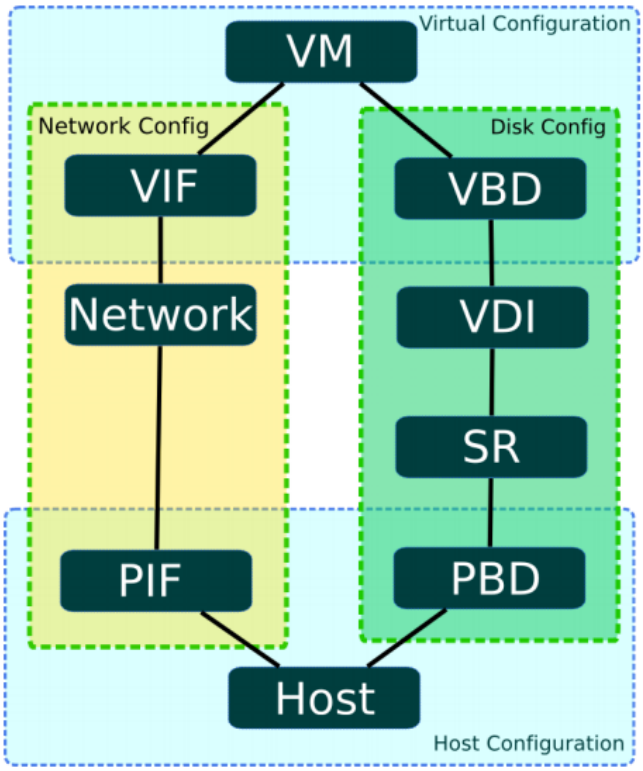
```
plug
unplug
```

- VIF – Un oggetto VIF (Virtual network Interface) rappresenta una connessione tra una macchina virtuale ed una connessione di rete. Allo stesso modo degli oggetti VBD quando una VM viene avviata tutti i suoi oggetti VIF vengono interrogati e viene connessa alle reti corrispondenti.

```
plug
unplug
```

- PIF – Un oggetto PIF (Physical Interface) rappresenta un collegamento tra un Host e una rete. Ed è in grado di rappresentare sia delle interfacce fisiche che delle VLAN.
- PBD – Un oggetto PBD (Physical Block Device) rappresenta un collegamento tra un Host e un oggetto SR.

Nell'immagine sottostante (Figura 6) è possibile avere una rappresentazione grafica di come tali oggetti interagiscono tra di loro:



6. Classi API di un pool XCP – Fonte: www.citrix.com

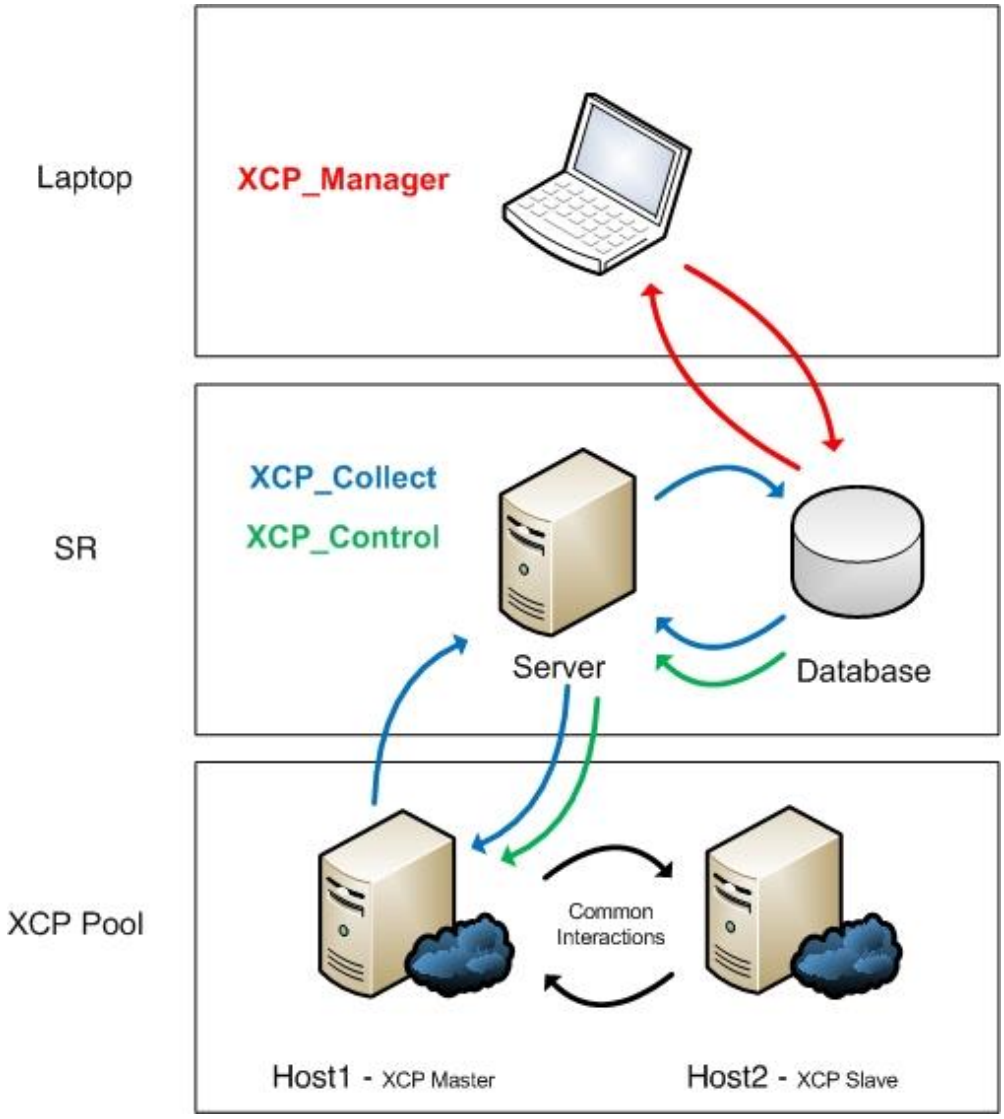
E' inoltre importante sottolineare ai fini dell'utilizzo di Xapi che, all'interno di un pool XCP, le richieste API possono essere effettuate solo all'host che è stato definito come master. Se viene effettuata la connessione ad un host slave Xapi solleva l'eccezione *HOST_IS_SLAVE* specificando l'indirizzo del master.

5. Progetto e sviluppo dell'applicazione

5.1. Introduzione agli applicativi

Data la mancanza all'interno di XCP di un meccanismo che permetta il bilanciamento del carico tramite delle regole personalizzate, si è deciso di sviluppare un insieme di programmi per far fronte a questa problematica.

I tre applicativi che verranno illustrati nel presente capitolo agiscono in collaborazione, usando come intermediario il database Mysql, per garantire che il pool XCP si trovi sempre in una situazione bilanciata, migrando le macchine quando necessario. Il programma controlla periodicamente lo stato del pool e verifica che le regole definite dall'amministratore siano rispettate. Nella figura qui riportata sono illustrati graficamente i tre applicativi e le loro interazioni:



7. Interazione tra gli applicativi realizzati

Come è possibile osservare dalla figura (Figura 7) sono presenti tre entità (Laptop, SR, XCP Pool) che interagiscono tra di loro secondo le direzioni definite dalle frecce. Ogni programma è scritto all'interno del riquadro in cui è in esecuzione ed interagisce con gli altri seguendo le frecce del proprio colore. Il meccanismo di funzionamento è il seguente:

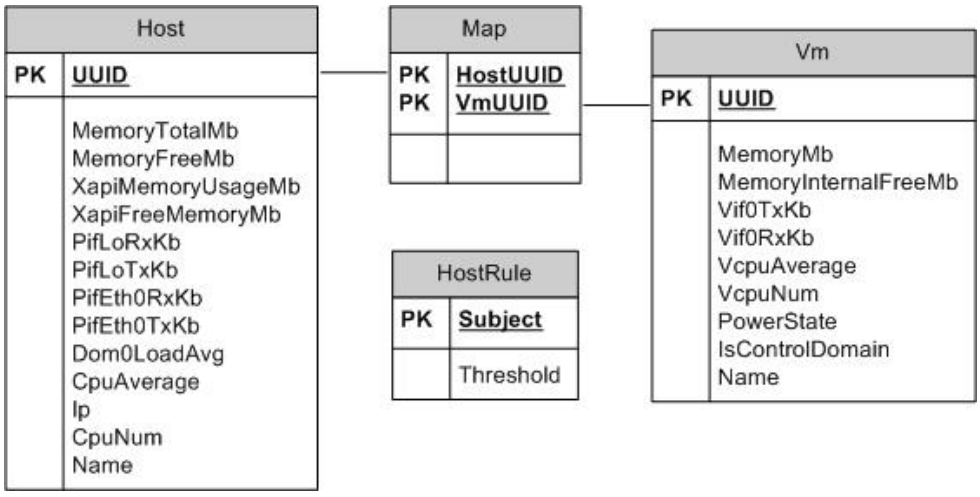
- XCP_Collect raccoglie le informazioni sullo stato del pool XCP facendo delle chiamate API al Master, le elabora e crea le query sql con le quali tiene costantemente aggiornato il database
- XCP_Manager permette ad un amministratore di sistema di collegarsi a tale database, osservare lo stato del pool descritto nel database e definire delle regole di bilanciamento personalizzate
- XCP_Control verifica che le regole inserite nel database siano rispettate e, nel caso non lo fossero, attua un algoritmo che migra le macchine virtuali problematiche in modo da rendere il carico quanto più omogeneo.

5.1.1. Modalità di interazione

L'interazione tra gli applicativi avviene, come già preannunciato, tramite un database che raccoglie l'insieme delle informazioni su ogni host e macchina virtuale del pool e le regole che vengono definite dall'utente.

5.2. Database Mysql

5.2.1. Struttura del database MySql



8. Struttura del Database

L’immagine precedente (Figura 8) è una rappresentazione grafica del database Xen sul quale i tre applicativi andranno ad agire per effettuare il monitoraggio e le query necessarie per bilanciare il carico del pool. Qui di seguito riportiamo le query SQL necessarie alla creazione del database Xen e delle tabelle corrispondenti:

```
CREATE DATABASE Xen;
USE Xen;

CREATE TABLE Host (

      UUID CHAR(36) PRIMARY KEY,
      MemoryTotalMb INT,
      MemoryFreeMb INT,
      XapiMemoryUsageMb INT,
      XapiFreeMemoryMb INT,
      PifLoRxKb INT,
      PifLoTxKb INT,
      PifEth0RxKb INT,
      PifEth0TxKb INT,
      Dom0LoadAvg DOUBLE,
      CpuAverage DOUBLE,
      Ip VARCHAR(15),
      CpuNum INT,
      Name VARCHAR(100)
);

CREATE TABLE Vm (
      UUID CHAR(36) PRIMARY KEY,
      MemoryMb INT,
      MemoryInternalFreeMb INT,
      Vif0TxKb INT,
      Vif0RxKb INT,
      VcpuAverage DOUBLE,
      VcpuNum INT,
      PowerState VARCHAR(20),
      IsControlDomain BOOLEAN,
      Name VARCHAR(100)
);

CREATE TABLE Map (
      HostUuid CHAR(36),
      VmUuid CHAR(36),
```

```
        primary key(`HostUuid`, `VmUuid` ),
        foreign key(`HostUuid`) references Host(`UUID`),
        foreign key(`VmUuid`) references Vm(`UUID`)
);

CREATE TABLE HostRule (
        Subject VARCHAR(100) PRIMARY KEY,
        Threshold DOUBLE NOT NULL
);
```

Possiamo quindi raccogliere per ogni oggetto le seguenti informazioni di monitoraggio:

- Per ogni Host:
    - UUID – UUID univoco, composto da 36 caratteri alfanumerici
    - MemoryTotalMb – Memoria totale dell'host
    - MemoryFreeMb – Memoria libera dell'host
    - XapiMemoryUsageMb – Memoria allocata dal demone Xapi
    - XapiFreeMemoryMb – Memoria disponibile al demone Xapi
    - PifLoRxKb – Dati ricevuti sull'interfaccia lo (Kb/s)
    - PifLoTxKb – Dati trasmessi sull'interfaccia lo (Kb/s)
    - PifEth0RxKb – Dati ricevuti sull'interfaccia eth0 (Kb/s)
    - PifEth0TxKb – Dati trasmessi sull'interfaccia eth0 (Kb/s)
    - Dom0LoadAvg – Carico del Dom0
    - CpuAverage – Carico complessivo dell'host
    - Ip – Ip dell'host
    - CpuNum – Numero di core dell'host
    - Name – Nome dell'host

- Per ogni VM:
    - UUID – UUID univoco, composto da 36 caratteri alfanumerici
    - MemoryMb – Memoria virtuale totale
    - MemoryInternalFreeMb – Memoria totale libera
    - Vif0TxKb – Dati trasmessi sull'interfaccia virtuale (Kb/s)
    - Vif0RxKb – Dati ricevuti sull'interfaccia virtuale (Kb/s)
    - VcpuAverage – Carico complessivo della macchina virtuale

- VcpuNum – Numero di core virtuali della macchina
- PowerState – Stato della macchina
- IsControlDomain – Indica se si tratta di un controller di dominio
- Name – Nome della macchina

La tabella Map contiene una corrispondenza Host-VM e ci permette di associare ad ogni macchina virtuale l'host sulla quale risiede. Come è infatti possibile osservare dalle query SQL le due colonne fanno riferimento rispettivamente alle tabelle Host e Vm ed assieme formano la chiave primaria poiché ad una macchina virtuale può essere associato un solo host. La tabella HostRule definisce invece le regole di monitoraggio che andremo ad analizzare successivamente in un paragrafo dedicato.

5.2.2. JDBC

Per permettere all'applicazione di connettersi ad un database Mysql è stato necessario utilizzare la libreria JDBC [24] che permette ad un programma scritto in Java di collegarsi ad un qualsiasi DBMS, indipendentemente da quello utilizzato. Tramite questa libreria è possibile sia interrogare che modificare le informazioni contenute in un database. La sua architettura prevede delle interfacce standard da mostrare all'applicazione, lasciando alla libreria il compito di caricare dinamicamente, a seconda del DBMS selezionato, il driver corrispondente.

5.3. Illustrazione degli applicativi

5.3.1. XCP_Collect

5.3.1.1. Obiettivi e Struttura

Il primo applicativo che andremo a descrivere è finalizzato a riportare fedelmente indicatori relativi all'attività degli host e delle macchine virtuali all'interno del database Mysql. Attraverso una serie di chiamate API vengono raccolte un insieme di informazioni, utili ai fini di monitoraggio e bilanciamento del carico, per poi essere utilizzate come parametri di query SQL. L'applicativo dispone di un file di configurazione separato dove è possibile inserire i dati di accesso per il database e per il pool XCP. Le informazioni sono inserite all'interno di un file

chiamato settings, contenuto nella stessa cartella dell'applicativo XCP_Collect e strutturato in questo modo:

```
XenHost=$XCPMasterHost
XenUsr=$XCPMasterUser
XenPass=$XCPMasterPassword
MysqlHost=$MysqlHost
MysqlUsr=$MysqlUser
MysqlPass=$MysqlPassword
```

Poiché un pool XCP risulta una struttura altamente personalizzabile è necessario che il programma preveda azioni di inserimento, rimozione ed aggiornamento dei dati contenuti nel database, onde evitare possibili incongruenze con la situazione del pool.

5.3.1.2. Illustrazione del Funzionamento

A causa dell'estrema complessità di una infrastruttura cloud, e della rapidità con cui possono essere modificate le informazioni in essa contenute, il software deve essere in grado di prevedere e gestire un insieme di casi molto vario. Poiché la complessità richiesta è notevole, per questo prototipo si è deciso di adottare un caso semplificato dell'infrastruttura, creando macchine virtuali piuttosto semplici. Il numero di errori che possono presentarsi in un'architettura del genere è molto vario ed è quindi opportuno prevedere dei meccanismi di rilevamento degli errori che permettano di risolvere eventuali problemi senza portare ad instabilità del sistema.

La prima operazione da compiere all'avvio del programma consiste nel recuperare i dati dal file di settings: data la struttura standard del nostro file descritta precedentemente risulta molto comodo utilizzare la classe `Properties` per ottenere automaticamente tutti i parametri di configurazione inseriti all'interno. Ecco qui riportato il codice corrispondente:

```
Properties configFile = new Properties();
try {
      configFile.load(new
      FileInputStream(System.getProperty("user.dir") + "./settings"));
} catch (FileNotFoundException e1) {
      System.out.println("Unable to locate settings file");
      e1.printStackTrace();
      System.exit(1);
}
```

```
String XenHost = configFile.getProperty("XenHost");
String XenUsr = configFile.getProperty("XenUsr");
String XenPass = configFile.getProperty("XenPass");
String MysqlHost = configFile.getProperty("MysqlHost");
String MysqlUsr = configFile.getProperty("MysqlUsr");
String MysqlPass = configFile.getProperty("MysqlPass");
```

Se il file di impostazioni non viene trovato il programma si chiude riportando un errore.

Successivamente andremo ad effettuare il collegamento con il pool XCP, come vedremo si tratta di una semplice connessione http con autenticazione. Durante questa operazione utilizzeremo tre classi definite all'interno della libreria Xapi: `Connection, Session e APIVersion`. La collaborazione di queste tre classi ci permette di ottenere una connessione al pool XCP:

```
Connection connection = new Connection(new URL("http://" + XenHost));
Session XenAPI = Session.loginWithPassword(connection,XenUsr,XenPass,
      APIVersion.latest().toString());
```

Tramite questi due oggetti `Session` e `Connection` sono ora in grado di compiere qualsiasi azione sul pool tramite Xapi come verrà mostrato nei prossimi capitoli.

Infine, come già preannunciato, la connessione al database Mysql:

```
String url = "jdbc:mysql://" + MysqlHost + ":3306/";
String dbName = "Xen";
String driver = "com.mysql.jdbc.Driver";
try {
      Class.forName(driver).newInstance();
      sqlConn =
            DriverManager.getConnection(url+dbName,MysqlUsr,MysqlPass)
} catch (Exception e) {
      System.out.println("Unable to connect to Mysql");
      e.printStackTrace();
      System.exit(1);
}
```

Giunto a questo punto posso passare all'esecuzione vera e propria del programma.

5.3.1.2.1. Raccolta delle informazioni sulle VM

La raccolta di informazioni sulle macchine virtuali in particolare ha richiesto un forte studio dell'aspetto sopra descritto: la molteplicità e la disomogeneità tra le macchina presenti all'interno dello stesso pool. E' Importante inoltre aggiungere che, per ottenere informazioni così dettagliate su ogni macchina virtuale, è stato

necessario installare su ognuna di queste il pacchetto XenServer Tools [25]: un applicativo di modeste dimensioni, disponibile per numerosi sistemi operativi ed installabile tramite un'immagine ISO, in grado di fornire a Xapi informazioni dettagliate sullo stato di una macchina. Tal pacchetto ci ha permesso di ottenere svariate informazioni che altrimenti non saremo stati in grado di raccogliere, tuttavia poteva esser necessario installare anche un altro pacchetto aggiuntivo chiamato "Performance Monitoring Enhancements Supplemental Pack Update" [26] per ottenere informazioni ancora più dettagliate. Tuttavia non è stato possibile effettuare l'installazione di tal pacchetto poiché disponibile solo in formato rpm e per architetture differenti da quella utilizzata. In ogni caso è stato possibile raccogliere le informazioni necessarie da ogni macchina virtuale tramite i seguenti metodi [27]:

```
VM.getUuid(xenConn)
VM.queryDataSource(xenConn, "memory")
VM.queryDataSource(xenConn, "memory_internal_free")
VM.queryDataSource(xenConn, "vif_0_tx")
VM.queryDataSource(xenConn, "vif_0_rx")
getVMAverageLoad(VM)
VM.getMetrics(xenConn).getVCPUsNumber(xenConn)
VM.getPowerState(xenConn).toString()
VM.getIsControlDomain(xenConn)
VM.getNameLabel(xenConn)
```

Come è possibile notare è stato fatto un largo utilizzo del metodo `queryDataSource` che, dato il nome di un parametro come argomento, restituisce il suo valore corrispondente. In altri casi sono stati utilizzati dei metodi separati per ottenere informazioni come l'UUID (`getUuid`) o lo stato della macchine (`getPowerState`). Solo nel caso del carico della macchine è stato necessario scrivere un nuovo metodo `getVMAverageLoad` poiché tramite Xapi è possibile ottenere solamente il carico per ogni singolo core virtuale e devo dunque calcolare il carico complessivo in questo modo:

```
long cpus = vm.getMetrics(xenConn).getVCPUsNumber(xenConn);
double tmp=0;
for(int i=0; i<cpus; i++){
      tmp = tmp + VM.queryDataSource(xenConn, "cpu" + i);
}
return (double)(tmp*100/cpus);
```

Inoltre, durante le procedure di inserimento e aggiornamento delle VM, al fine di evitare errori o possibili incongruenze nell'inserimento è stato necessario valutare particolarmente due parametri: lo stato della macchina virtuale e l'eventuale presenza di un controller di dominio.

Per comprendere meglio il problema descritto viene riportato qui sotto il metodo *getInsertVMQuery* che, data una VM non presente nel database, restituisce la query INSERT corrispondente:

```
public PreparedStatement getInsertVMQuery(VM vm){
      PreparedStatement addVMQuery=null;
      try{

addVMQuery = sqlConn.prepareStatement("INSERT INTO Vm
      VALUES(?,?,?,?,?,?,?,?,?,?)");

if(vm.getPowerState(xenConn).toString().equals("Running")){
      addVMQuery.setString(1, vm.getUuid(xenConn));
      addVMQuery.setInt(2, (int)Math.round(vm.queryDataSource(xenConn,
       "memory")/1024/1024));
      if(!vm.getIsControlDomain(xenConn)){
            addVMQuery.setInt(3,
                  (int)Math.round(vm.queryDataSource(xenConn,
                  "memory_internal_free")/1024));
            addVMQuery.setInt(4,
                  (int)Math.round(vm.queryDataSource(xenConn,
                  "vif_0_tx")/1024));
            addVMQuery.setInt(5,
                  (int)Math.round(vm.queryDataSource(xenConn,
                  "vif_0_rx")/1024));
      } else {
            addVMQuery.setNull(3, Types.INTEGER);
            addVMQuery.setNull(4, Types.INTEGER);
            addVMQuery.setNull(5, Types.INTEGER);
      }
      addVMQuery.setDouble(6, getVMAverageLoad(vm));
      addVMQuery.setInt(7,
            (int)Math.round(vm.getMetrics(xenConn).getVCPUsNumber
            (xenConn)));
      addVMQuery.setString(8, vm.getPowerState(xenConn).toString());
      addVMQuery.setBoolean(9, vm.getIsControlDomain(xenConn));
      addVMQuery.setString(10, vm.getNameLabel(xenConn));
}else {
      addVMQuery.setString(1, vm.getUuid(xenConn));
      addVMQuery.setInt(2,
            (int)Math.round(vm.getMemoryDynamicMax(xenConn)/1024/1024))
            ;
      addVMQuery.setNull(3, Types.INTEGER);
      addVMQuery.setNull(4, Types.INTEGER);
      addVMQuery.setNull(5, Types.INTEGER);
      addVMQuery.setNull(6, Types.INTEGER);
      addVMQuery.setInt(7, (int)Math.round(vm.getVCPUsMax(xenConn)));
      addVMQuery.setString(8, vm.getPowerState(xenConn).toString());
      addVMQuery.setBoolean(9, vm.getIsControlDomain(xenConn));
      addVMQuery.setString(10, vm.getNameLabel(xenConn));
}
```

```
        } catch (Exception e){
              e.printStackTrace();
        }
        return addVMQuery;
}
```

Come è possibile osservare dalla figura i parametri della query vengono impostati in maniera differente a seconda che la macchina virtuale sia accesa o spenta, oppure nel caso in cui sia un controller di dominio. Nel dettaglio una VM spenta non è in grado di eseguire il metodo `queryDataSource` quindi è stato necessario sostituire tali metodi con altri che non prevedano l'esecuzione della macchina virtuale (`getMemoryDynamicMax`) o, nel caso di parametri strettamente legati all'esecuzione, prevedere la possibilità di assegnare un valore nullo (`addVMQuery.setNull`). La logica qui descritta ovviamente è analoga anche nel metodo *getUpdateVMQuery* che, data una VM già inserita nel database, restituisce la query UPDATE corrispondente.

5.3.1.2.2. Raccolta delle informazioni sugli Host

Nel caso della raccolta delle informazioni da parte degli host il metodo si presenta in maniera analoga, semplificando notevolmente i casi d'uso poiché le macchine fisiche non possono essere dei controller di dominio e nemmeno essere spente (altrimenti non verrebbero visualizzate):

```
public PreparedStatement getInsertHostQuery(Host hs){
      PreparedStatement addHostQuery=null;
    try{

addHostQuery = sqlConn.prepareStatement("INSERT INTO Host
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

addHostQuery.setString(1, hs.getUuid(xenConn));
addHostQuery.setInt(2, (int)Math.round(hs.queryDataSource(xenConn,
"memory_total_kib")/1024));
addHostQuery.setInt(3, (int)Math.round(hs.queryDataSource(xenConn,
"memory_free_kib")/1024));
addHostQuery.setInt(4, (int)Math.round(hs.queryDataSource(xenConn,
"xapi_memory_usage_kib")/1024));
addHostQuery.setInt(5, (int)Math.round(hs.queryDataSource(xenConn,
"xapi_free_memory_kib")/1024));
addHostQuery.setInt(6, (int)Math.round(hs.queryDataSource(xenConn,
"pif_lo_rx")/1024));
addHostQuery.setInt(7, (int)Math.round(hs.queryDataSource(xenConn,
"pif_lo_tx")/1024));
addHostQuery.setInt(8, (int)Math.round(hs.queryDataSource(xenConn,
"pif_eth0_rx")/1024));
addHostQuery.setInt(9, (int)Math.round(hs.queryDataSource(xenConn,
"pif_eth0_tx")/1024));
addHostQuery.setDouble(10, hs.queryDataSource(xenConn, "loadavg"));
```

```
addHostQuery.setDouble(11, getHostAverageLoad(hs));
addHostQuery.setString(12, hs.getAddress(xenConn));
addHostQuery.setInt(13, hs.getHostCPUs(xenConn).size());
addHostQuery.setString(14, hs.getNameLabel(xenConn));

    } catch (Exception e){
            e.printStackTrace();
    }
    return addHostQuery;
}
```

Anche in questo caso è presente un metodo separato per ottenere il carico dell'host fisico ed uno per ottenere la query di UPDATE corrispondente. Nel caso degli Host non è necessario avere installato XenTools per avere delle informazioni dettagliate sullo stato.

5.3.1.2.3. Logica di insert/update/replace

In questo paragrafo vengono descritte le modalità e la logica con cui vengono effettuati gli aggiornamenti delle informazioni con l'obiettivo di rendere la rappresentazione del database il più fedele possibile a quella del pool XCP. Per fare ciò sono state creati tre metodi appositi:

- *updateVmInfo* – Aggiorna informazioni VM
- *updateHostInfo* – Aggiorna informazioni Host
- *updateMapInfo* – Aggiorna le corrispondenze VM-Host

Tutti e tre i metodi utilizzano la stessa logica di aggiornamento, qui sotto è riportato il codice del metodo *updateVmInfo* :

```
private static void updateVmInfo(){

      ps = sqlConn.prepareStatement("CREATE TEMPORARY TABLE deleteVm
                        ( VmUuid CHAR(36) )");
      ps.execute();

      ps = sqlConn.prepareStatement("INSERT INTO deleteVm " +
                        "SELECT UUID " +
                        "FROM Vm " +
                        "WHERE ( `UUID` ) " +
                        "NOT IN (" + VmOp.getAllVmString() + ")" );
      ps.execute();

      ps=sqlConn.prepareStatement("DELETE " +
                        "FROM Vm " +
                        "WHERE ( `UUID` ) " +
                        "IN ( SELECT * " +
                              "FROM deleteVm)");
      ps.execute();
```

```
        for (VM vm : VmOp.getVmList()){

              if (!vm.getIsASnapshot(xenConn)&&
                          !vm.getIsATemplate(xenConn)){
                    if(VmOp.isVmInDatabase(vm)){
                          ps = VmOp.getUpdateVMQuery(vm);
                          ps.execute();
                    } else {
                          ps = VmOp.getInsertVMQuery(vm);
                          ps.execute();
                    }
              }
        }
}
```

Come è possibile osservare dal codice sopra riportato non è necessario solo aggiornare ed inserire nuove macchine virtuali nel database per avere una rappresentazione fedele del pool XCP, ma è importante valutare anche la loro eventuale rimozione. Per rimuovere una macchina virtuale non più presente dal database vengono utilizzate le seguenti query:

```
CREATE TEMPORARY TABLE deleteVm ( VmUuid CHAR(36) )

INSERT INTO deleteVm SELECT UUID FROM Vm WHERE ( `UUID` ) NOT IN
(" + VmOp.getAllVmString() + ")"

DELETE FROM Vm WHERE ( `UUID` ) IN ( SELECT * FROM deleteVm )
```

Il procedimento ideato prevede la creazione di una tabella temporanea all'interno della quale vengono inseriti tutti gli UUID delle macchina virtuali presenti nel database ma non più esistenti all'interno del pool. L'ultima query infine provvede a rimuovere dalla tabella originale tutte le occorrenze contenute nella tabella temporanea. Su questo procedimento è importante fare due osservazioni:

- Impossibilità di JDBC di inserire all'interno di una query un numero variabile di parametri – Come è possibile notare nella seconda query, JDBC non permette di inserire un numero imprecisato di parametri rendendo necessario l'inserimento degli UUID tramite una stringa concatenata. Sarebbe stato possibile inserire un numero variabile concatenando all'interno del metodo `prepareStatement` una stringa contenente N punti interrogativi quanti sono gli elementi da inserire, e compiere un secondo ciclo per popolare tutti gli argomenti inseriti con dei valori. Tuttavia un metodo di questo tipo risulta altamente inefficiente e dispendioso di risorse, quindi si è preferito adottare una tecnica

formalmente meno corretta ma più efficace dal punto di vista computazionale.

- Impossibilità di eseguire un'unica query per la rimozione – Inizialmente si pensava di creare un'unica query pesante per poter rimuovere le macchina virtuali non più presenti: l'idea era di fare una query di DELETE di tutte le VM presenti nella tabella "Vm" ma non più presenti nel pool. Questa idea è risultata non realizzabile per un limite di Mysql che non permette di effettuare in un'unica query il DELETE e il SELECT dei valori di una stessa tabella. L'idea proposta risulta indubbiamente meno efficiente rispetto ad un'unica query, ma è risultata la migliore alternativa.

Nella parte finale del metodo un semplice ciclo analizza ogni VM, escludendo snapshot e template, tramite il metodo `isVmInDatabase(VM)` che, data una VM, restituisce se è già presente o meno all'interno del database e, a seconda della risposta, viene elaborata una query di UPDATE o di INSERT.

Descritti i metodi utilizzati passiamo ora all'analisi del main:

```
try {
	updateVmInfo();
	updateHostInfo();
	updateMapInfo();
} catch (MySQLIntegrityConstraintViolationException e){
	updateMapInfo();
	ps=sqlConn.prepareStatement("DROP TABLE IF EXISTS deleteVm");
	ps.execute();
	ps=sqlConn.prepareStatement("DROP TABLE IF EXISTS deleteHost");
	ps.execute();
	updateVmInfo();
	updateHostInfo();
}
```

Il main può dunque percorrere due differenti percorsi:

- Nessun host o vm è stato eliminato – In tal caso si seguono le operazioni di *updateVmInfo*, *updateHostInfo*, *updateMapInfo*. Che vengono completate senza sollevare eccezioni.
- Sono stati eliminati un host o una vm – In questo caso l'esecuzione dei due metodi *updateVmInfo* e *updateHostInfo* scatena un'eccezione del tipo `MySQLIntegrityConstraintViolationException` poiché viene tentata l'eliminazione di una chiave primaria referenziata nella tabella Map. L'operazione viene dunque interrotta scatenando un'eccezione: in questo

caso è necessario eseguire per primo il metodo *updateMapInfo()* che andrà a rimuovere dalla lista delle corrispondenze i valori non più presenti. Successivamente vengono eliminate le tabelle temporanee e vengono eseguiti i metodi *updateVmInfo* e *updateHostInfo* che ora potranno completarsi senza incorrere in eccezioni.

L'eliminazione, se presenti, delle tabelle temporanee `deleteVm` e `deleteHost` è necessaria per poter proseguire con i due metodi successivi.

5.3.2. XCP_Manager

5.3.2.1. Obiettivi e Struttura

XCP_Manager è un'applicazione a riga di comando che permette all'amministratore di sistema di avere la situazione in tempo reale del pool e di aggiungere o rimuovere regole di bilanciamento del carico. Il programma si interfaccia direttamente con il database Mysql, sempre tramite JDBC, ed estrae di volta in volta le informazioni richieste dall'amministratore mostrandole a video. Sono state implementate le seguenti operazioni:

- `SHOW VMS`
- `SHOW VM $uuid`
- `SHOW HOSTS`
- `SHOW HOST $uuid`
- `SHOW RULES HOST`
- `CREATE RULE HOST`
- `DELETE RULE HOST`

Il programma agisce in lettura e scrittura sulla tabella HostRule, mentre effettua solo delle letture dalle rimanenti.

5.3.2.2. Illustrazione del Funzionamento

5.3.2.2.1. Informazioni sullo stato del sistema

Qui di seguito è riportato un esempio di esecuzione dei primi due comandi inseriti nella lista:

```
XCP_Manager> SHOW VMS

       - 0a511f3b-ca5f-2c5d-7751-55b6a138f86e - testVM
```

```
 - 67bad7be-dcce-a819-f7e2-9c92c9e626bc - Copy of testVM
 - 8f37b891-1a48-c55d-ab76-5e82dd473bf7 - Control domain on
                                          host: Host1
 - c9918a2b-1b72-0cd8-6571-419db7f3be19 - testVM2Ubu
 - e7cdd975-a19a-1442-905a-54dc747adcbe - Control domain on
                                          host: Host2

XCP_Manager> SHOW VM 0a511f3b-ca5f-2c5d-7751-55b6a138f86e

      UUID - 0a511f3b-ca5f-2c5d-7751-55b6a138f86e
      Total Memory (MB) - 1024
      Free Memory (MB) - 814
      Transmitted data on the Virtual Interface (Kb/s) - 0
      Received data on the Virtual Interface (Kb/s) - 10
      VCpu Average - 50%
      VCpu Number - 2
      Power State - Running
      Control Domain - false
      Name - testVM
```

Il risultato fornisce delle informazioni specifiche sulla macchina virtuale selezionata; l'esecuzione e l'output sono analoghi nel caso degli host. Per ottenere tali informazioni XCP_Manager effettua delle query di tipo SELECT sul database usando l'UUID come chiave di ricerca e riporta sullo schermo il risultato dell'operazione.

5.3.2.2.2. Inserimento regole

Per quanto riguarda la gestione del bilanciamento è possibile utilizzare i tre comandi mostrati precedentemente per visualizzare, creare o cancellare le regole. Ecco un esempio di creazione di una regola, visualizzazione ed eliminazione:

```
XCP_Manager> CREATE RULE HOST

      Add a new Host Rule (Pool wide):

      When [PifEth0RxKb/PifEth0TxKb/OccupiedMemoryMb/CpuAverage]:
      OccupiedMemoryMb
      Is more than [MB]:
      200
      above the medium migrate vms to the less loaded host

      When OccupiedMemoryMb is more than 200 MB above the medium
      migrate to the less loaded host
      Confirm ? [Y/N] Y

XCP_Manager> SHOW RULES HOST

      Rules - Balance host when:

       - CpuAverage is more than 25 % above the medium
```

```
       - OccupiedMemoryMb is more than 200 MB above the medium

XCP_Manager> DELETE RULE HOST

      Rules - Balance host when:

       - CpuAverage is more than 25 % above the medium
       Do you want to delete this rule? [Y/N] N

       - OccupiedMemoryMb is more than 200 MB above the medium
       Do you want to delete this rule? [Y/N] Y

Rule deleted
```

5.3.3. XCP_Control

5.3.3.1. Obiettivi e Struttura

XCP_Control è il cosiddetto attuatore di tutte le regole definite precedemente: elabora i dati delle macchine virtuali e degli host, verifica che le regole impostate siano rispettate e, nel caso non lo fossero, si occupa del bilanciamento del carico. Il tempo di esecuzione in questo caso non è definito ed è lasciato a discrezione dell'utente: esecuzioni molto frequenti porterebbero ad un elevato uso di tempo di CPU e memoria, ma al tempo stesso aumenterebbero notevolmente la reattività del sistema. E' importante notare che il tempo di esecuzione di XCP_Control deve andare di pari passo con quello di XCP_Collect onde evitare possibili discrepanze tra i dati presenti nel database e la situazione attuale del sistema.

5.3.3.2. Illustrazione del Funzionamento

5.3.3.2.1. Illustrazione dell'algoritmo di migrazione

Poiché l'obiettivo è quello di creare delle regole che vadano a gestire l'intero pool di macchine abbiamo dovuto dare la possibilità di generare delle regole a livello di pool piuttosto che a livello di singola Vm come era stato inizialmente pensato. Nel nostro caso è stata considerata l'idea di valutare lo stato di un determinato host come scostamento rispetto alla media del pool: un metodo calcola la media di un determinato valore a livello di pool e vede di quanto un singolo host si scosta rispetto alla media. In questo modo è possibile valutare quale host si trova più sopra la media (risultando quindi sovraccarico) e quale host si trova più sotto la media (risultando quindi scarico). Tramite questo algoritmo siamo in grado di

rilevare immediatamente gli host che saranno oggetto della migrazione. Le regole di migrazione vengono salvate sul database utilizzando due campi: il soggetto della regola ed il limite da rispettare per non dover effettuare il bilanciamento. Ovviamente è necessario specificare delle regole sensate ed ideate sulla base delle esigenze dell'amministratore e della struttura del pool: delle regole troppo restrittive potrebbero comportare un'instabilità del sistema, continuando a migrare macchine per ottenere un'ottimizzazione di pochi punti percentuali, al contrario delle regole troppo lasche causerebbero una non ottimizzazione del carico.

5.3.3.2.2. Metodi XAPI per avviare il processo di migrazione

Poiché il processo di migrazione avviene all'interno di host appartenenti ad uno stesso pool, e quindi aventi tutti lo stesso storage condiviso, la mole di codice da utilizzare per la migrazione vera e propria si riduce notevolmente. Le informazioni che sono necessarie per effettuare la migrazione sono due: la macchina virtuale da migrare e l'host di destinazione. L'operazione può esser svolta in due modalità differenti: sincrona o asincrona. Nel caso di una migrazione sincrona il programma viene sospeso e riprende l'esecuzione solo al termine della migrazione, nella modalità asincrona invece il programma continua la sua esecuzione lanciando la migrazione in background. Pur non avendo interesse a far continuare il programma durante la migrazione si è optato per una migrazione di tipo asincrono: in questo modo è possibile mostrare a video il progresso dell'operazione. Il codice in questione è il seguente:

```
System.out.println("Starting Migration ...");
Task tk = sourceVm.poolMigrateAsync(xenConn, destHost, new
      HashMap<String,String>());
while(tk.getProgress(xenConn) != 1){
      System.out.println("Progress: " + tk.getProgress(xenConn));
      Thread.sleep(2000);
}
System.out.println("End Migration");
```

Come è possibile osservare, la funzione `poolMigrateAsync` ha come parametri la connessione al pool, l'host di destinazione e una mappa contenente delle opzioni aggiuntive (nel nostro caso nessuna). Il metodo restituisce un oggetto di tipo Task che rappresenta il progresso dell'operazione di migrazione. La migrazione risulta completata quando il metodo `getProgress(xenConn)` restituirà

uno, fino ad allora viene stampato a video il progresso dell'operazione ad intervalli di due secondi.

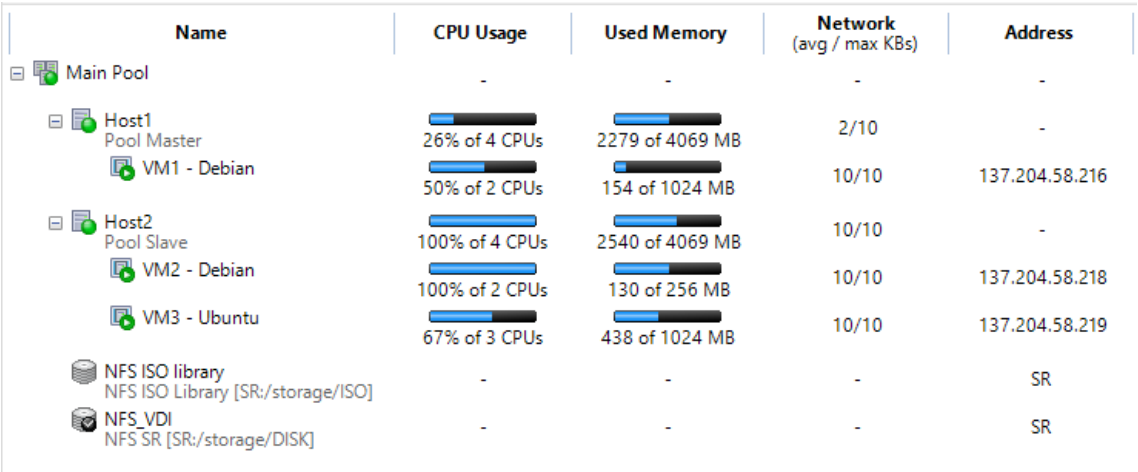
6. Risultati sperimentali

6.1. Esempi di esecuzione

Per portare degli esempi di esecuzione si è fatto un largo uso dell'applicazione `stress` [28] che, tramite alcuni parametri a riga di comando, permette di effettuare degli stress-test sulle macchine virtuali. In questo modo è stato possibile simulare un carico di lavoro su CPU, rete e memoria, in modo da poter effettuare dei test esaustivi dell'applicazione. Qui si seguito riportiamo alcune delle opzioni principali che sono state utilizzate:

```
-c, --cpu N
      spawn N workers spinning on sqrt()
-i, --io N
      spawn N workers spinning on sync()
-m, --vm N
      spawn N workers spinning on malloc()/free()
```

Nell'immagine riportata qui di seguito vien mostrata la situazione presente all'interno del pool prima dell'esecuzione del programma, con alcune macchine sotto stress-test ed una situazione evidentemente sbilanciata. Nel dettaglio sono presenti tre macchine virtuali di diverse dimensioni, aventi un carico variabile:



9. XenCenter – Situazione sbilanciata

Illustriamo brevemente la figura (Figura 9) per comprendere meglio la struttura della rete:

L'Host1 (Pool Master) ha in esecuzione al suo interno un'unica macchina virtuale VM1 con sistema operativo Debian che utilizza il 50% dei due core virtuali a lei dedicati. L'Host2 (Pool Slave) ha in esecuzione due macchine virtuali: VM2 con

sistema operativo Debian utilizza il 99% della CPU, usando tutte la risorse computazionali a sua disposizione (due core virtuali) mentre VM3 utilizza il 66% delle sue risorse (tre core virtuali).

Il test non tiene in considerazione l'utilizzo di memoria RAM, quindi il suo valore in questo test viene trascurato.

Come è facilmente intuibile dall'immagine la situazione in questo caso risulta fortemente sbilanciata:

- Host1 – Questo host risulta molto scarico: il suo carico attuale è del 25% e, anche nel caso in cui VM1 volesse richiedere tutte le risorse a sua disposizione, il carico dell'host non supererebbe mai il 50%. Ciò è indicativo di quanto una configurazione di questo tipo risulti molto inefficiente, comportando un notevole spreco di risorse.
- Host2 – L'Host2, al contrario, risulta evidentemente sovraccarico: il suo carico di CPU risulta attualmente al 100% e la macchine virtuale VM3 ha ancora risorse a disposizione. In questo caso se VM3 volesse richiedere tutte le risorse a sua disposizione potrebbe causare un rallentamento generale dell'host, poiché sono state rese assegnate alle macchine virtuali più risorse di quelle effettivamente disponibili.

6.1.1. Regole sul pool

6.1.1.1. Inserimento di una regola

Nel caso sopra illustrato viene evidenziato un evidente sbilanciamento delle risorse. Ricordiamo che attualmente XCP_Manager non ha ancora inserito alcuna regola all'interno del pool, vediamo quindi come vengono visualizzate le macchine virtuali all'interno del programma:

```
XCP_Manager> SHOW VMS

       - 0a511f3b-ca5f-2c5d-7751-55b6a138f86e - VM1 - Debian
       - 67bad7be-dcce-a819-f7e2-9c92c9e626bc - VM2 - Debian
       - 8f37b891-1a48-c55d-ab76-5e82dd473bf7 - Control domain on
                                                host: Host1
       - c9918a2b-1b72-0cd8-6571-419db7f3be19 - VM3 - Ubuntu
       - e7cdd975-a19a-1442-905a-54dc747adcbe - Control domain on
                                                host: Host2
```

```
XCP_Manager> SHOW VM 0a511f3b-ca5f-2c5d-7751-55b6a138f86e

      UUID - 0a511f3b-ca5f-2c5d-7751-55b6a138f86e
      Total Memory (MB) - 1024
      Free Memory (MB) - 813
      Transmitted data on the Virtual Interface (Kb/s)  - 0
      Received data on the Virtual Interface (Kb/s) - 6
      VCpu Average - 50%
      VCpu Number - 2
      Power State - Running
      Control Domain - false
      Name - VM1 - Debian

XCP_Manager> SHOW VM 67bad7be-dcce-a819-f7e2-9c92c9e626bc

      UUID - 67bad7be-dcce-a819-f7e2-9c92c9e626bc
      Total Memory (MB) - 256
      Free Memory (MB) - 84
      Transmitted data on the Virtual Interface (Kb/s)  - 0
      Received data on the Virtual Interface (Kb/s) - 10
      VCpu Average - 99%
      VCpu Number - 2
      Power State - Running
      Control Domain - false
      Name - VM2 - Debian

XCP_Manager> SHOW VM c9918a2b-1b72-0cd8-6571-419db7f3be19

      UUID - c9918a2b-1b72-0cd8-6571-419db7f3be19
      Total Memory (MB) - 1024
      Free Memory (MB) - 365
      Transmitted data on the Virtual Interface (Kb/s)  - 0
      Received data on the Virtual Interface (Kb/s) - 10
      VCpu Average - 66%
      VCpu Number - 3
      Power State - Running
      Control Domain - false
      Name - VM3 - Ubuntu
```

Come è possibile osservare XCP_Collect è stato in grado di riportare correttamente all'interno del database tutte le informazioni delle macchine virtuali, è inoltre in grado di riportare informazioni sui controller di dominio che invece non vengono visualizzati nella schermata di XenCenter. Ripetiamo l'operazione per i singoli host:

```
XCP_Manager> SHOW HOSTS

       - 6874ed0e-e89f-3fb4-306c-0ae78732ed62 - Host2
       - eb966de6-276a-0428-2ecb-d1b2a69eff57 - Host1

XCP_Manager> SHOW HOST eb966de6-276a-0428-2ecb-d1b2a69eff57

      UUID - eb966de6-276a-0428-2ecb-d1b2a69eff57
      Total Memory (MB) - 4069
      Free Memory (MB) - 1791
      Xapi Memory Usage (MB) - 6
```

```
      Xapi Free Memory (MB) - 2
      Received data on the Physical Interface Lo (Kb/s) - 2
      Transmitted data on the Physical Interface Lo (Kb/s) - 2
      Received data on the Physical Interface Eth0 (Kb/s) - 14
      Transmitted data on the Physical Interface Eth0 (Kb/s) - 6
      Dom0 Load Average - 0%
      Cpu Load Average - 25%
      Ip - 137.204.58.91
      Cpu Number - 4
      Name - Host1

XCP_Manager> SHOW HOST 6874ed0e-e89f-3fb4-306c-0ae78732ed62

      UUID - 6874ed0e-e89f-3fb4-306c-0ae78732ed62
      Total Memory (MB) - 4069
      Free Memory (MB) - 1529
      Xapi Memory Usage (MB) - 2
      Xapi Free Memory (MB) - 1
      Received data on the Physical Interface Lo (Kb/s) - 0
      Transmitted data on the Physical Interface Lo (Kb/s) - 0
      Received data on the Physical Interface Eth0 (Kb/s) - 19
      Transmitted data on the Physical Interface Eth0 (Kb/s) - 15
      Dom0 Load Average - 0%
      Cpu Load Average - 100%
      Ip - 137.204.58.93
      Cpu Number - 4
      Name - Host2
```

Aggiungiamo una regola host che ci permetta di bilanciare le macchine, nel nostro caso lo sbilanciamento evidente è dato dalla CPU, quindi specificheremo una regola di questo tipo:

```
XCP_Manager> CREATE RULE HOST

      Add a new Host Rule (Pool wide):

      When [PifEth0RxKb/PifEth0TxKb/OccupiedMemoryMb/CpuAverage]:
      CpuAverage
      Is more than [%]:
      25
      above the medium migrate vms to the less loaded host

      When CpuAverage is more than 25 % above the medium migrate to
      the less loaded host
      Confirm ? [Y/N] Y

XCP_Manager> SHOW RULES HOST

      Rules - Balance host when:

       - CpuAverage is more than 25 % above the medium
```

La regola in questo momento è stata inserita all'interno del database, alla prossima esecuzione di XCP_Control questo andrà a verificare la regola e valuterà se effettuare o meno il bilanciamento del carico.

6.1.1.2. Analisi dell'algoritmo di migrazione

Come precedentemente anticipato le regole riguardanti le singole macchine virtuali sono state sostituire a favore di regole a livello di pool, riguardanti ovvero i singoli host ed il loro rapporto con l'intera infrastruttura. Quando una nuova regola viene aggiunta al database XCP_Control valuta la media del valore monitorato a livello di pool e vede di quanto i singoli host si scostano rispetto a tale media. In questo modo è possibile individuare con facilità gli host che più si allontanano dalla media ed intervenire per bilanciarli.

Nel nostro caso è stata appena creata una regola che indica di bilanciare il carico se un host ha un utilizzo di CPU superiore del 25% rispetto alla media. La media del carico di CPU a livello di pool è del 62.5% (125/2) dunque nel nostro caso Host2 (che ha il carico al 100%) infrange la regola e dunque necessita di bilanciamento.

6.1.1.3. Test di migrazione

Con l'esecuzione della regola sopra-citata XCP_Control decide di spostare la VM2 all'Host1, ottenendo un risultato di questo tipo:

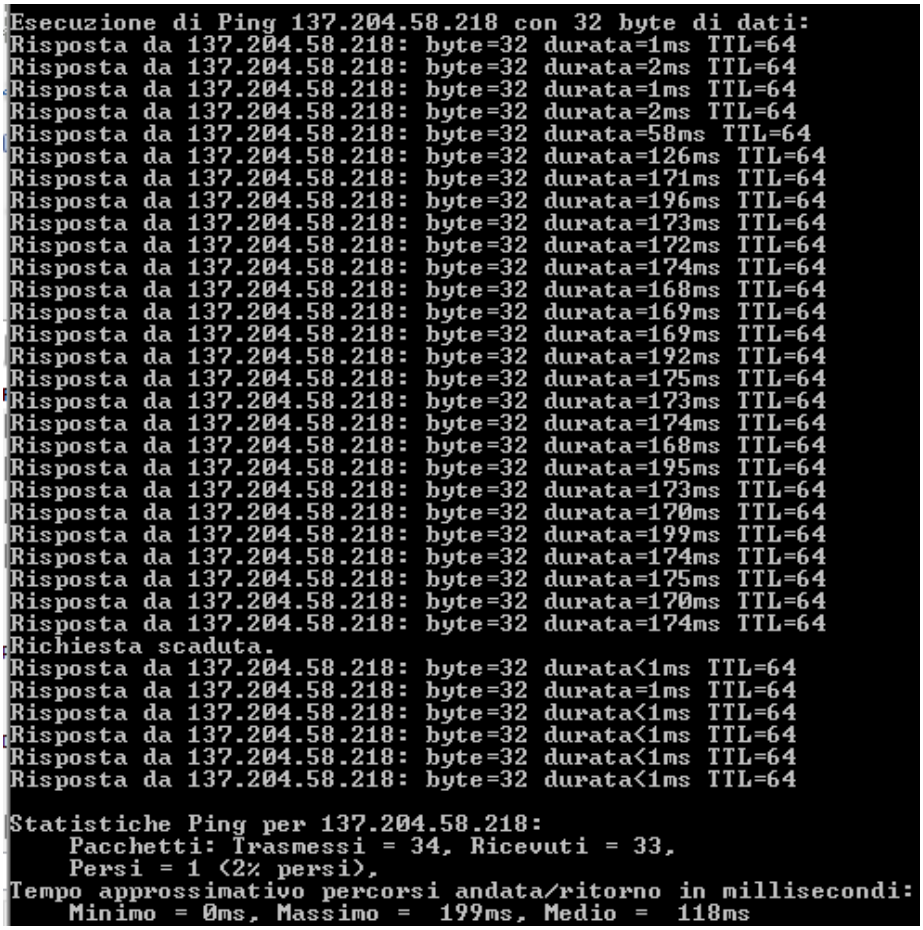
Name	CPU Usage	Used Memory	Network (avg / max KBs)	Address
Main Pool	-	-	-	-
Host1 Pool Master	75% of 4 CPUs	2539 of 4069 MB	2/11	-
VM1 - Debian	50% of 2 CPUs	64 of 1024 MB	10/10	137.204.58.216
VM2 - Debian	100% of 2 CPUs	54 of 256 MB	10/10	137.204.58.218
Host2 Pool Slave	50% of 4 CPUs	2280 of 4069 MB	14/14	-
VM3 - Ubuntu	67% of 3 CPUs	98 of 1024 MB	14/14	137.204.58.219
NFS ISO library NFS ISO Library [SR:/storage/ISO]	-	-	-	SR
NFS_VDI NFS SR [SR:/storage/DISK]	-	-	-	SR

10. XenCenter – Situazione Bilanciata

Com'è possibile apprezzare (Figura 10) attualmente la situazione di sbilanciamento risulta risolta: sia la VM1 che la VM3 possono utilizzare tutte le risorse a loro disposizione senza causare problemi agli host. Attualmente la regola impostata risulta rispettata, poiché con una media di pool del 62.5% ho

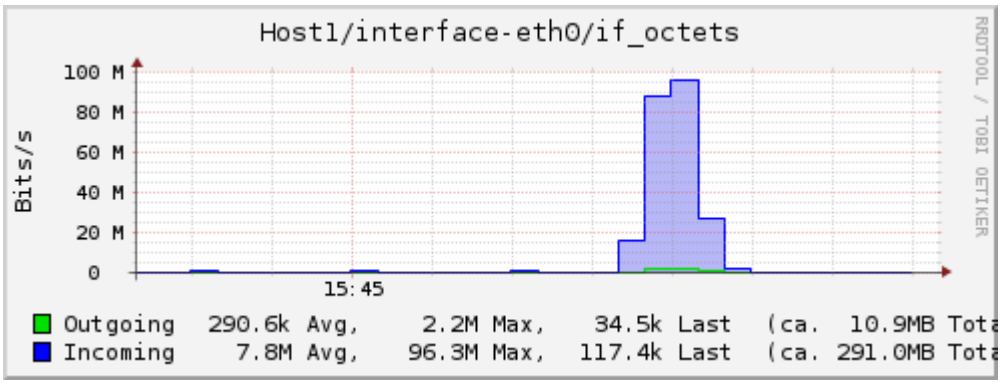
comunque Host1 al 75% e Host2 al 50% quindi lo scarto di entrambi gli host rispetto alla media risulta inferiore al 25%.

La migrazione è stata effettuata correttamente dall'Host2 all'Host1 in un tempo di ventisei secondi durante il quale è stato effettuato un monitoraggio della raggiungibilità della macchina tramite ping, questi sono i risultati ottenuti:



11. Ping della macchina virtuale durante la migrazione

Come è possibile osservare dall'immagine (Figura 11) riportata sopra, durante la migrazione la macchina rimane raggiungibile lungo tutto il processo, si ha infatti un solo pacchetto perso durante l'intera operazione che non compromette in alcun modo il servizio offerto. E' inoltre possibile osservare che durante la migrazione i tempi di ping aumentano fino a raggiungere un valore di 160-170 millisecondi, con un picco massimo di 199. E' inoltre riportato qui di seguito il grafico del traffico di rete di Host1 che, nel nostro caso, è la destinazione della macchina virtuale:



12. Traffico sull'interfaccia eth0 di Host1

Tramite il programma di monitoraggio illustrato nel capitolo iniziale siamo stati in grado di visualizzare il traffico di rete durante la migrazione della macchina virtuale (Figura 12). Nel nostro caso è possibile notare che durante la migrazione si raggiunge la velocità massima di 96.3 Mbit/s (su un massimo teorico di 100 Mbit/s) per un trasferimento totale di 291MB dall'Host2 verso l'Host1. I dati trasferiti corrispondono ovviamente alla dimensione della RAM della macchina virtuale (256MB nel nostro caso) a cui si vanno ad aggiungere i dati di alcune copie iterative che giustificano i 35MB aggiuntivi.

6.2. Performance

6.2.1. Tempi di Esecuzione

Per valutare l'impatto di XCP_Collect sulle performance generali e valutare quanto tempo fosse impiegato dal processo per essere completato abbiamo utilizzato in combinazione il sistema di monitoraggio precedentemente installato e l'utility `time` fornita da linux. Riportiamo l'output del comando time durante l'esecuzione di XCP_Collect, situato sull'host SR:

```
root@SR:~# time java -jar XCP_Collect.jar
real    0m3.642s
user    0m1.836s
sys     0m0.096s
```

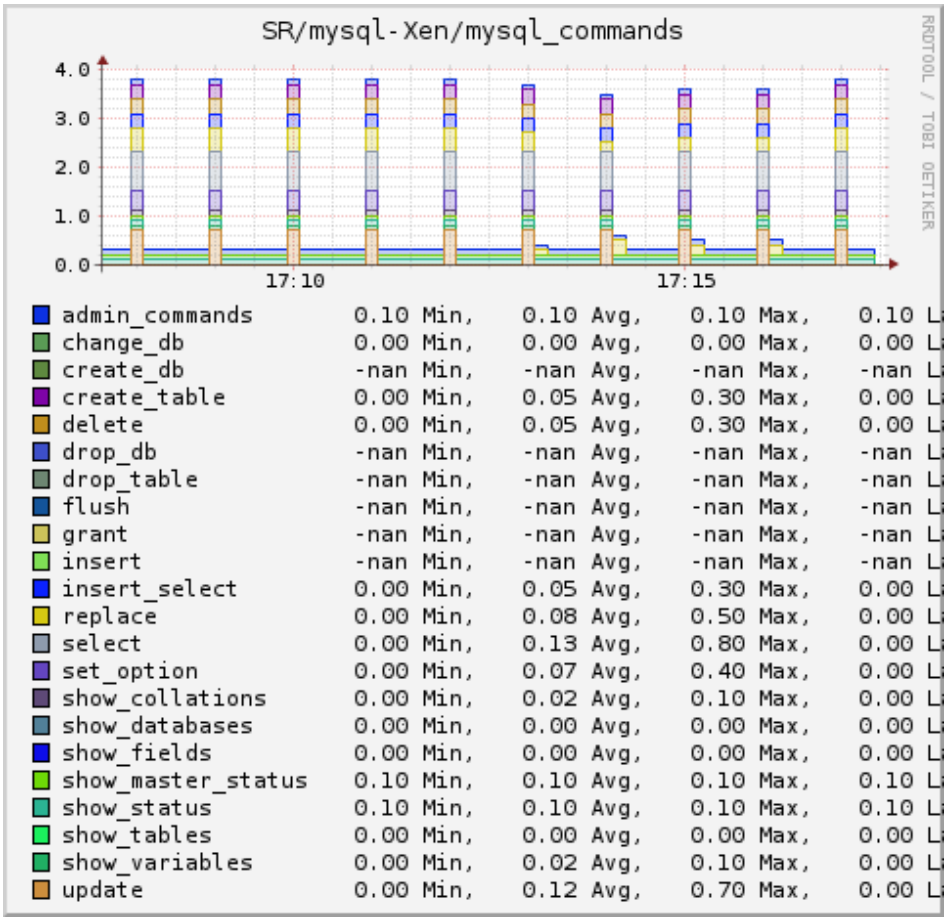
Come è possibile notare i tempi di esecuzione si aggirano intorno ai 3.6 secondi di cui circa 1.8 sono di attività utente e circa 0.1 di attività del sistema operativo. Il basso valore del tempo utente utilizzato per l'esecuzione indica che la maggior parte del tempo totale è impiegato nell'attesa di una risposta da parte di Xapi. Per tentare di migliorare le performance del programma si è deciso di provare ad

eseguire il programma anche sull'Host1 che, essendo master del demone Xapi, poteva garantire performance migliori. I risultati tuttavia non hanno portato ad un aumento considerevole delle prestazioni:

```
root@Host1:~# time java -jar XCP_Collect.jar
real    0m3.388s
user    0m1.504s
sys     0m0.140s
```

Poiché dunque l'esecuzione su SR o sul dom0 di Host1 non influisce in maniera significativa sulle performance del programma e la documentazione di XCP sconsiglia di aggiungere applicativi al dom0 si è deciso di mantenere l'esecuzione di XCP_Collect su SR.

Oltre a dei test sul tempo di esecuzione di XCP_Collect sono stati monitorati anche i valori di utilizzo del database Mysql al fine di capire quale intervallo fosse il migliore per effettuare gli aggiornamenti del database. Questo grafico illustra le query effettuate dal programma verso il database:

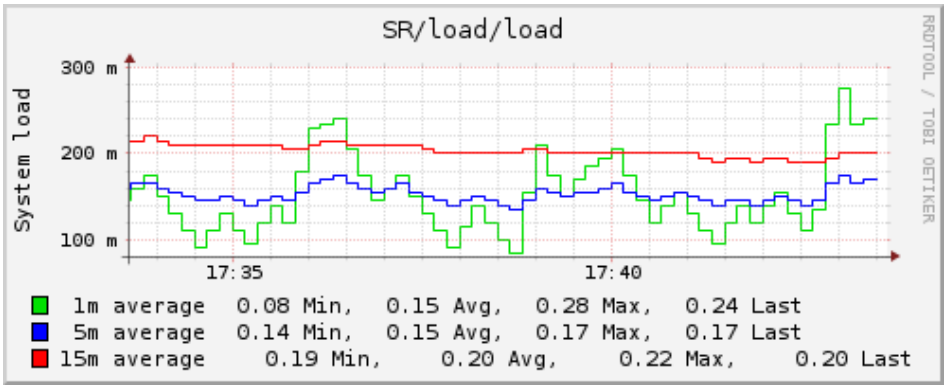


13. Operazioni eseguite sul database Mysql

Come è possibile visualizzare dal grafico (Figura 13) precedente la maggior parte delle operazioni che vengono effettuate sono query di SELECT e UPDATE.

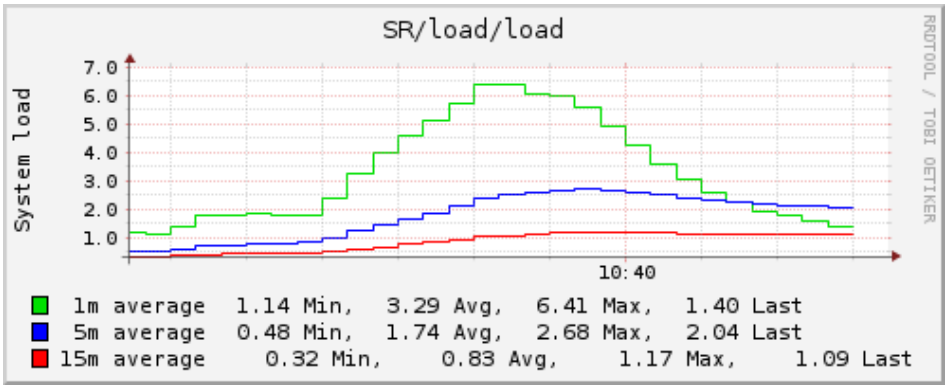
6.2.2. Carico delle macchine

Per verificare l'efficienza del nostro applicativo su SR abbiamo monitorato il carico dell'host in questione, cercando di capire se era in grado o meno di tollerare un carico del genere. Ricordiamo infatti che l'host SR ha al suo interno un server NFS contenente dischi virtuali e immagini ISO, un database Mysql, il master nel sistema di monitoraggio e l'applicativo XCP_Collect. Tuttavia i risultati sono molto incoraggianti:



14. Carico dell'host SR

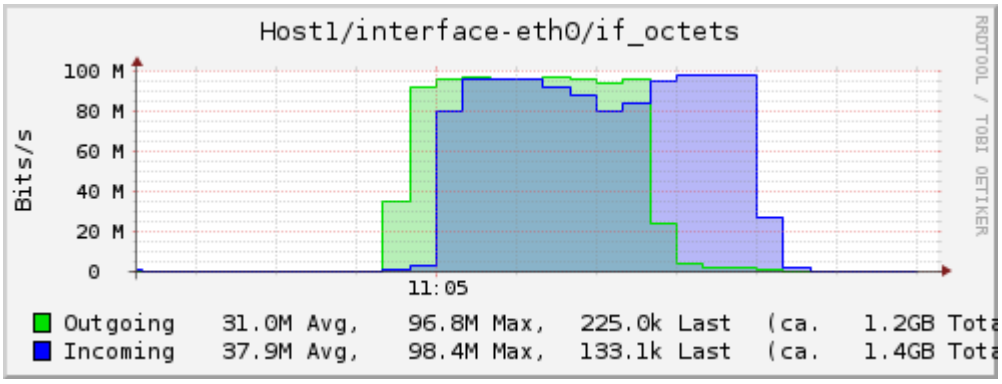
Come è possibile infatti dedurre dal grafico (Figura 14) abbiamo un carico che si aggira tra i 0.15 e 0.20, quindi l'host risulta a regime abbastanza scarico. Nel complessivo sono stati eseguiti un insieme di test sulle macchine virtuali per valutare quale operazione potesse mettere in crisi il nostro sistema: dai dati rilevati è stato notificato che operazioni intensive di I/O sulle macchine virtuali sono in grado di aumentare notevolmente il carico dell'host SR.



15. Carico dell'host SR

Nel caso qui sopra (Figura 15) sono stati avviati su una macchina virtuale dell'Host1 due processi agenti in maniera intensiva sullo storage condiviso. Come è possibile osservare dal grafico il carico è arrivato fino a 6.41 punti prima che il test venisse interrotto. Questo tipo di problematica era prevedibile: la macchina è evidentemente troppo poco potente per reggere dei carichi adatti alla produzione, non tanto dal punto di vista della CPU, ma dal punto di vista della velocità dei dischi e del collegamento alla rete (la maggior parte del tempo risulta infatti utilizzato per operazioni di iowait).

Per effettuare un ulteriore test di funzionamento è stata effettuata una doppia migrazione di macchine virtuali per vedere se il sistema fosse in grado di reggere un carico di questo tipo. Nel dettaglio sono state migrate due macchine virtuali con un GB di memoria l'una e, durante la migrazione, la loro raggiungibilità è stata monitorata tramite ping. Qui di seguito è riportato il grafico dell'operazione:



16. Traffico sull'interfaccia eth0 di Host1

Come era facilmente deducibile abbiamo in intenso traffico sull'interfaccia eth0 sia in entrata che in uscita (a causa della doppia migrazione) raggiungendo picchi anche di 98.4 Mbit/s (Figura 16). La mole di dati trasferiti risulta ovviamente molto più ampia rispetto ai test di migrazione effettuati precedentemente e si aggira attorno al GB di dati più alcuni MB aggiuntivi dovuti alle copie iterative. Il monitoraggio della raggiungibilità degli host ha evidenziato una maggiore perdita di pacchetti per entrambe le macchine migrate: sono stati persi quattro pacchetti in un caso e 5 nell'altro.

7. Conclusioni e sviluppi futuri

7.1. Conclusioni

Nel presente lavoro di tesi è stato effettuato uno studio approfondito sulle tecniche di virtualizzazione e cloud computing attualmente presenti sul mercato, cercando di evidenziare i pregi e i difetti di ogni soluzione analizzata. In particolare si è deciso di concentrare i propri sforzi sull'ambiente di virtualizzazione Xen Cloud Platform, analizzando nel dettaglio le sue componenti fondamentali e studiando il suo funzionamento sia in ambienti di piccole dimensioni che in produzione tramite alcune esperienze di utilizzo. E' stato necessario progettare e successivamente configurare un ambiente di test adatto per studiare il funzionamento del software utilizzato, in modo da poter effettuare test approfonditi sull'efficienza di XCP in vari ambiti. Il risultato di questo lavoro ha portato ad evidenziare i limiti e i vantaggi di una soluzione di questo tipo, realizzando tre applicazioni dedicate al fine di risolvere alcuni limiti di questa tecnologia.

Le applicazioni sono state utilizzate per realizzare una soluzione di monitoraggio e bilanciamento del carico funzionante e reattivo alle modifiche tipiche di una infrastruttura cloud. Il software è stato realizzato in linguaggio Java e tramite l'ausilio delle API di programmazione fornite dall'ambiente di virtualizzazione stesso. Durante la fase di progettazione si è posta particolare attenzione riguardo la molteplicità di casi che una struttura cloud deve prevedere, cercando di valutare la maggior parte di questi.

Dai test che sono stati effettuati si evidenzia che il bilanciamento del carico viene raggiunto in tempi ragionevoli e senza aver necessità di utilizzare lunghi tempi di CPU per l'aggiornamento delle informazioni. Sono state affrontate diverse difficoltà di configurazione, manutenzione e progettazione dell'applicativo, ma valutando con attenzione i pro ed i contro di ogni soluzione si è sempre riusciti a trovare il giusto compromesso fornendo sempre alti parametri di ottimizzazione ed usabilità. In conclusione a questo progetto possiamo affermare con sicurezza l'affidabilità e la robustezza di una piattaforma come Xen Cloud Platform all'uso in ambienti di produzione forniti di hardware adeguato. I nostri test hanno

evidenziato delle buone performance anche su hardware meno performanti, che permettono un utilizzo anche in ambienti di test pre-produzione. L'applicativo sviluppato si è dimostrato una buona alternativa alla mancanza di supporto da parte di XCP nel bilanciamento del carico delle macchine virtuali.

7.2. Sviluppi Futuri

7.2.1. Debugging

Durante la sperimentazione e il testing dell'applicazione sono stati previsti più casi possibili per tentare di fornire all'applicativo una rappresentazione quanto più adeguata del pool XCP. Tuttavia, come già anticipato, all'interno di un ambiente cloud le modalità di funzionamento sono molteplici e delle più varie, non è stato quindi possibile effettuare test esaustivi su tutte le possibili combinazioni di host, macchine virtuali e pool. Oltre a ciò bisognerebbe valutare la possibilità di osservare il funzionamento in pool di maggiori dimensioni e con macchine molto meno omogenee o particolarmente problematiche.

7.2.2. Implementazione di algoritmi per il green-computing

Attualmente l'applicativo fornisce la possibilità di inserire solo regole riguardanti il bilanciamento del carico, quindi distribuire al meglio le macchine virtuali su più host possibili. Negli ultimi anni tuttavia ha preso grande piede l'idea del green-computing, ovvero la possibilità di raggruppare più macchine virtuali possibili all'interno di meno host, in modo da avere host molto carichi ma permettendo di spegnere i rimanenti in modo da risparmiare energia. Sarebbe interessante implementare tali politiche all'interno dell'applicativo dando la possibilità di creare dunque regole concettualmente opposte a quelle attualmente presenti.

7.2.3. Possibilità di inserire regole concatenate

Un'altra funzione che risulterebbe molto utile da implementare è la possibilità di concatenare più regole tra di loro. In questo modo sarebbe possibile avviare la migrazione solo quando un insieme di regole vengono infrante così da avere un controllo più preciso sulle macchine virtuali del nostro pool. Ad esempio potrei decidere di bilanciare il carico quando sia la CPU che la memoria superano una

certa soglia, oppure valutare il traffico complessivo di rete unendo una regola sul traffico in entrata con una sul traffico in uscita.

Bibliografia

[1] – VmWare Virtualization Overview - http://www.vmware.com/pdf/virtualization.pdf

[2] – Virtualization Techniques - http://debian-handbook.info/browse/stable/sect.virtualization.html

[3] – Introduzione alla virtualizzazione - http://datastorage02.maggioli.it/data/docs/www.egov.maggioli.it/Virtualizzazione%20parte%201%20per%20web.pdf

[4] – Xen - http://www.xen.org/

[5] – Università di Cambridge - http://www.cam.ac.uk/

[6] – VmWare - http://www.vmware.com

[7] – OpenVz - http://openvz.org

[8] – KVM - http://www.linux-kvm.org/page/Main_Page

[9] – XCP - http://www.xen.org/products/cloudxen.html

[10] – VmWare vSphere - http://www.vmware.com/it/products/datacenter-virtualization/vsphere/overview.html

[11] – Openstack - http://www.openstack.org/

[12] – Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauery, Ian Pratt, Andrew Warfield; Xen and the Art of Virtualization - University of Cambridge Computer Laboratory

[13] – XenServer – http://www.citrix.com/xenserver

[14] – Citrix - http://www.citrix.com/

[15] – XenMotion - http://support.citrix.com/article/CTX115813

[16] – Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny, Eric July, Christian Limpach, Ian Pratt, Andrew Warfield; Live Migration of Virtual Machines - University of Cambridge Computer Laboratory

[17] – NFS - http://nfs.sourceforge.net/

[18] – Mysql - http://www.mysql.it/

[19] – Munin - http://munin-monitoring.org/

[20] – collectd - http://collectd.org/

[21] – XenCenter - http://community.citrix.com/display/xs/XenCenter

[22] – Xapi - http://wiki.xen.org/wiki/XAPI

[23] – XML-RPC - http://xmlrpc.scripting.com/

[24] – JDBC - http://www.oracle.com/technetwork/java/overview-141217.html

[25] – XenServer Tools - http://support.citrix.com/article/CTX135099

[26] - Performance Monitoring Enhancements Supplemental Pack Update - http://support.citrix.com/article/CTX135977

[27] – XenAPI Classes - http://docs.vmd.citrix.com/XenServer/6.0.0/1.0/en_gb/api/

[28] – Stress - http://weather.ou.edu/~apw/projects/stress/