Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

RaspberryPi datacenter in a tiny box: un sistema distribuito in scala ridotta per presentazioni pubbliche

Relatore:
Chiar.mo Prof.
Renzo Davoli

Presentata da:
Devid Farinelli

Sessione II
Anno Accademico 2015/2016

Abstract

Questo elaborato tratta della realizzazione di un sistema distribuito in scala ridotta da utilizzare per presentazioni pubbliche. Introduce ai concetti fondamentali della virtualizzazione con maggiore attenzione alla virtualizzazione hardware su RaspberryPi. Viene descritto in cosa consiste il sistema, come lo si utilizza e come è stato creato per terminare con una valutazione dell'efficienza delle soluzioni considerate.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo elaborato viene descritto il processo di progettazione e realizzazione di un sistema fisico distribuito che possa essere utilizzando in presentazioni pubbliche come infrastruttura sulla quale dimostrare il funzionamento della virtualizzazione di reti.
La virtualizzazione nasce negli anni 60 come risposta ai problemi di multitasking dei primi computer dell'epoca, si cerca infatti un modo per far si che più programmi possano essere eseguiti concorrentemente sullo stesso elaboratore condividendo il processore.
Le prime soluzioni a questo problema si basavano sull'astrazione di risorse con lo scopo di condividerle fra applicazioni, come l'implementazione del time-sharing per condividere la CPU fra più programmi e la memoria virtuale per permettere ai computer di avere a disposizione maggiore RAM rispetto a quella realmente presente fisicamente.
In questi anni di progressi gli elaboratori sono diventati più potenti permettendo alle tecniche di virtualizzazione di evolversi per soddisfare le nuove esigenze. Sono cambiati i tempi ma la necessità di condividere risorse è rimasta una delle motivazioni principali per lo sviluppo di tecnologie di virtualizzazione. Un esempio molto attuale riguarda i servizi di cloud computing dove l'ottimizzazione nell'utilizzo delle risorse hardware è un tema fondamentale.
Per virtualizzazione si intende creare dei componenti (risorse virtuali) che

indipendentemente dall'implementazione espongano un'interfaccia simile (o identica) a quella di una determinata risorsa e che possano sostituirsi ad essa in maniera più o meno trasparente. Uno dei benefici principali di questa tecnologia diventa evidente nella simulazione dei componenti hardware, dove grazie alla capacità di astrazione è possibile gestire risorse per natura statiche (RAM, CPU, dischi, ecc..) con la stessa dinamicità con la quale si manipolano i dati, permettendo di costruire infrastrutture virtuali dove è possibile gestire l'architettura hardware tramite il software.
In base al tipo di risorsa simulata si possono identificare diverse categorie di virtualizzazione: Hardware, Network, Desktop, Application e Storage.
In una di queste categorie si trova VDE (Virtual Distributed Ethernet), un pacchetto open-source di applicativi nato presso Università di Bologna grazie al Prof. Renzo Davoli, che permette di fare virtualizzazione di rete utilizzando applicazioni come VXVDEX che consente di creare network di macchine virtuali collegate alla stessa LAN.
Per dimostrare l'utilità e l'efficacia di VXVDEX in pubblico si è voluto realizzare un sistema distribuito composto da computer in grado di eseguire macchine virtuali da interconnettere tramite la rete. Il progetto sviluppato vuole rendere possibile dimostrare il funzionamento di applicativi di rete durante presentazioni pubbliche mettendo a disposizione una vera e propria rete fisica di elaboratori, in scala ridotta e con tempi di boot limitati. Per la parte hardware del sistema sono stati usati 3 Raspberry Pi collegati tramite ethernet ad uno switch, in modo da garantire dimensioni ridotte, e sulla parte software si è voluto invece minimizzare i tempi di boot delle macchine virtuali analizzando diverse soluzioni. Sono state valutate diverse tecnologie di virtualizzazione (Qemu, Xen, KVM, VirtualBox), diverse distribuzioni da utilizzare come sistema operativo per il Raspberry (Raspbian, TinyCorelinux) e diverse distribuzioni da virtualizzare sulle macchine virtuali da collegare con VXVDEX (Dogebian, BasicLinux, TinyCore). Sono state implementate diverse alternative e dopo una analisi si è scelta come soluzione quella di usare Qemu su Raspbian per avviare macchine virtuali Dogebian (una distribuzio-

ne compilata con BuildRoot). In questo elaborato viene descritto il processo di progettazione ed implementazione di un sistema distribuito di dimensioni ridotte da utilizzare durante presentazioni pubbliche.

Capitolo 2

Contesto tecnologico

2.1 Virtualizzazione

La virtualizzazione consiste nel creare versioni virtuali delle risorse, dove in informatica per risorsa si intende qualunque componente fisico (hardware) o virtuale (software).
Le risorse virtuali si possono utilizzare al posto di quelle reali poiché, nonostante la diversa implementazione, espongono la stessa interfaccia, tramite la virtualizzazione, per esempio, è possibile simulare un'architettura hardware sulla quale eseguire un sistema operativo.
La virtualizzazione è utile perché consente di costruire applicazioni flessibili ed affidabili spostando la complessità delle infrastrutture a livello software rendendo così gli applicativi più indipendenti dalle piattaforme sulle quali sono eseguiti.
In generale si identificando diverse categorie in base al tipo risorse virtualizzate: Hardware, Network, Application, Desktop e Storage.

Hardware Virtualization

La virtualizzazione hardware, consiste nell'emulare componenti hardware fino ad intere piattaforme, è una forma di virtualizzazione molto diffusa grazie anche al fatto che permette di eseguire più sistemi operativi concorrentemente

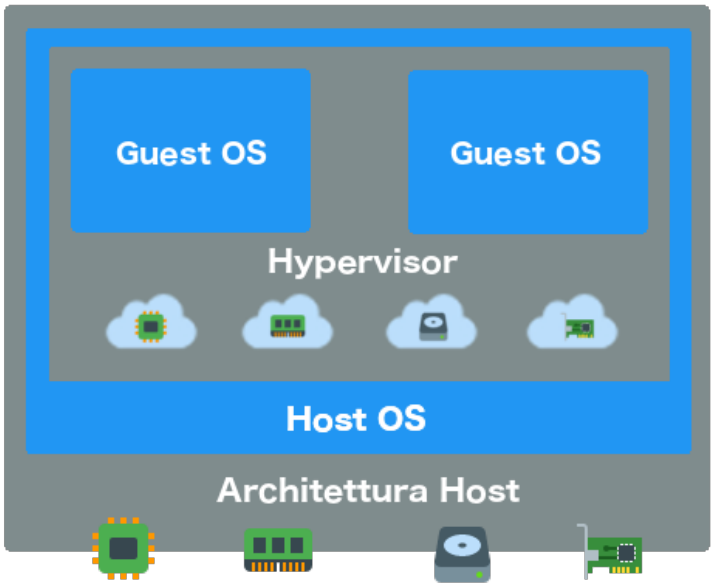

Figura 2.1: Due sistemi operativi (Guest) in esecuzione su un hardware virtualizzato.

sullo stesso hardware. Un emulatore molto utilizzato è Qemu, che è in grado di virtualizzare diverse architetture (x86, x86_64, ARM, ecc..) e moltissime altre periferiche (schede di rete, hard disk, ecc..).

Network Virtualization

La virtualizzazione network consiste nel simulare configurazioni di rete collegando fra loro diverse macchine reali o virtuali. Un esempio è VDE (Virtual Distributed Network) che consente di emulare switch, cavi e connettori per collegare i nodi di una rete virtuale.

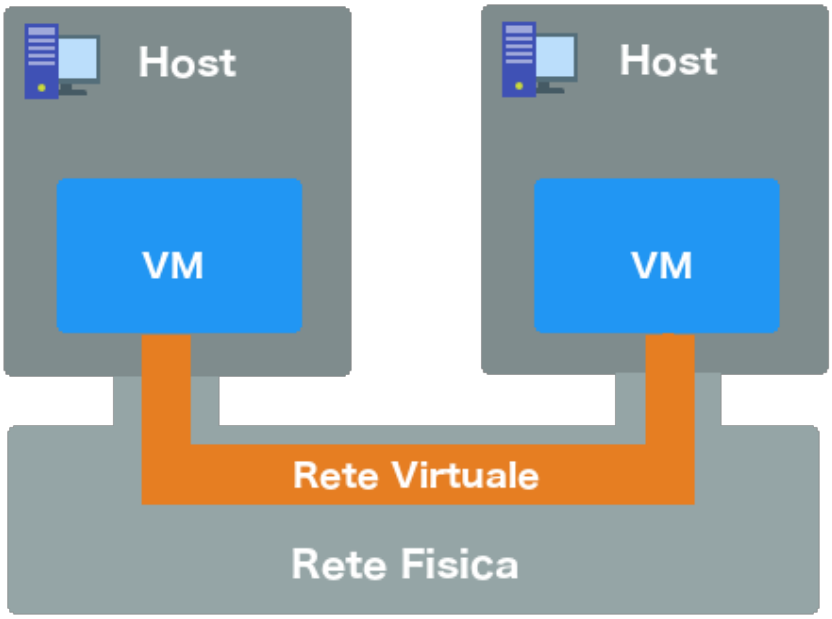

Figura 2.2: Una rete virtuale costruita su una infrastruttura hardware generica.

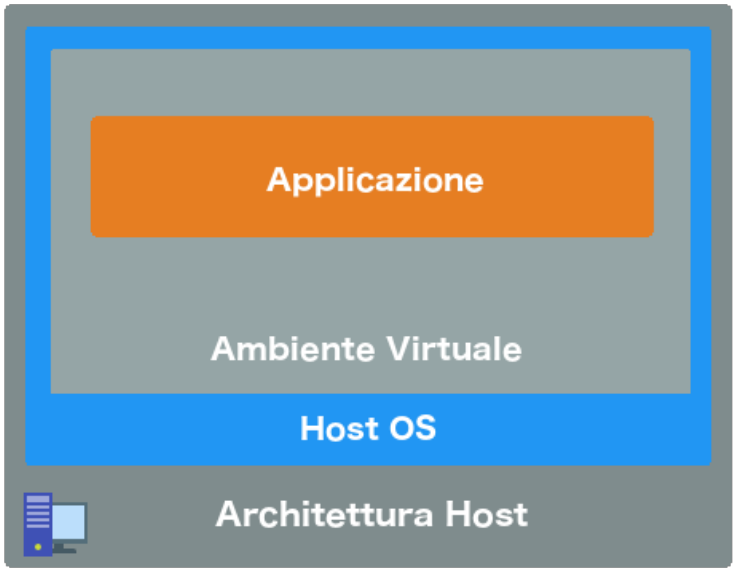

Figura 2.3: Una applicazione in esecuzione in ambiente virtualizzato

Application Virtualization

La virtualizzazione di applicazioni consiste nel creare ambienti per l'esecuzione di programmi. Come Wine che permette di utilizzare applicazioni Windows su Linux, la virtualizzazione di applicazioni consiste nel costruire un layer di compatibilità fra il sistema operativo e l'applicazione.

Desktop Virtualization

La virtualizzazione desktop consiste nel virtualizzare un ambiente desktop rendendolo completamente indipendente dall'hardware sul quale viene eseguito. Nella Remote Desktop Virtualization, per esempio, l'ambiente risiede fisicamente su altre macchine della rete che ne garantiscono la disponibilità su qualunque dispositivo e ottima resistenza ai guasti.

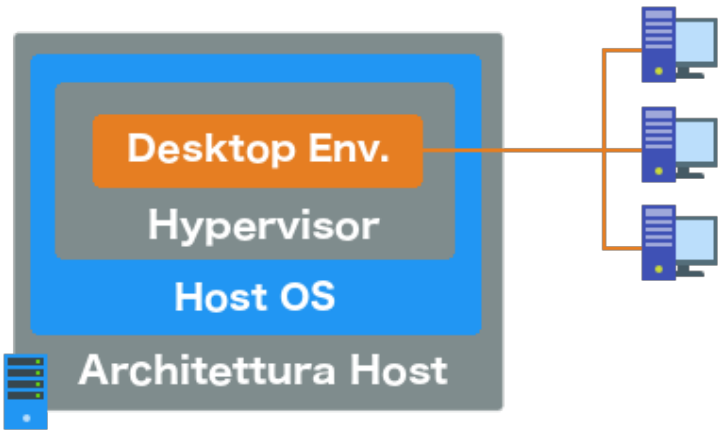

Figura 2.4: Da ogni client è possibile accedere ad un ambiente Desktop virtuale che risiede su un server

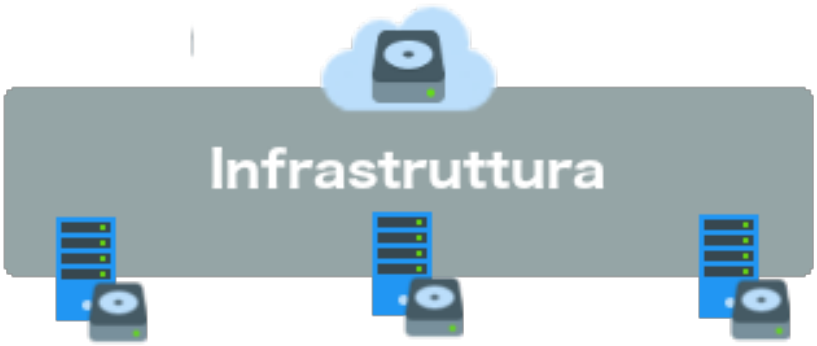

Figura 2.5: Dati che risiedono su più macchine diverse vendono presentati come un file system unico virtuale

Storage Virtualization

La virtualizzazione dei dispositivi si memorizzazione consiste nel virtualizzare files e file systems. Prendendo come esempio il DFS (Distributed File System) le risorse risiedono su altre macchine della rete, ma sono organizzate e visualizzate all'utente come un singolo file system attraverso una console (Google Drive, Dropbox, ecc..).

2.2 Virtualizzazione Hardware

Ad oggi la virtualizzazione è molto utilizzata specialmente in ambito hardware, dove si definisce il concetto di macchina virtuale (VM) col quale solitamente si intende un sistema operativo in esecuzione su un hardware virtuale.

Sfruttando questa tecnologia i servizi di cloud computing consentono di eseguire ed amministrare macchine virtuali nel cloud, spesso avviando più VM sullo stesso hardware così da ottimizzare il consumi di risorse ed incrementare la portabilità, la scalabilità e l'affidabilità.
Solitamente si possono identificare diverse modalità in base all'approccio utilizzato: la virtualizzazione completa e la virtualizzazione parziale.

Full Virtualization

La virtualizzazione completa consiste nella simulazione di tutte le componenti di una piattaforma hardware. Si utilizza un layer solitamente chiamato hypervisor, sul quale è possibile eseguire un sistema operativo ospite inconsapevole di trovarsi in un ambiente simulato.
La virtualizzazione totale senza supporto del processore avviene spesso per traduzione dinamica, ossia traducendo le istruzioni del processore guest per farle seguire al processore dell'host. Vista la complessità intrinseca della virtualizzazione la traduzione dinamica è una tecnica molto costosa in termini di risorse rispetto alle soluzioni che sfruttano il supporto hardware per la virtualizzazione.
Per ottimizzare l'efficienza di questa tecnica è possibile sfruttare la paravirtualizzazione, che consiste nel costruire un'interfaccia simile ma non identica a quella delle risorse da virtualizzare, sul quale eseguire un sistema modificato per poter essere eseguito in ambiente paravirtualizzato. La paravirtualizzazione è utile per evitare tali operazioni che sarebbe inutili o costose se eseguite in ambiente virtualizzato, come ad esempio l'algoritmo dell'ascensore per diminuire i tempi di può essere evitato su hard disk virtuali.

Partial Virtualization

La virtualizzazione parziale consiste nel virtualizzare solo parte delle risorse, con la possibilità di sfruttare le implementazioni hardware di determinate funzionalità. Per esempio, negli ambienti con più macchine virtuali è pos-

sibile sfruttare le funzionalità di rilocazione fornite dall'hardware, per far si che ad ogni macchina virtuale sia assegnata una porzione di memoria.

2.3 Tecnologie di virtualizzazione

Fra le tecnologie di virtualizzazione hardware più utilizzate vi è sicuramente da menzionare VMware, una piattaforma di virtualizzazione proprietaria molto potente e versatile. Tuttavia si vuole approfondire meglio il mondo open source dove le soluzioni più diffuse sono Xen, KVM, Qemu e VirtualBox.

Xen

Xen è un hypervisor che costruisce un layer sul quale è possibile eseguire più sistemi operativi contemporaneamente. Xen viene eseguito in una modalità CPU privilegiata rispetto agli altri processi e si occupa di gestire le risorse delle diverse macchine virtuali in esecuzione appoggiandosi direttamente sull'hardware.

KVM

KVM è un hypervisor compatibile con molte architetture, in grado di fare virtualizzazione di sistema. È disponibile come modulo kernel ma richiede un processore con supporto per la virtualizzazione. Dalla versione 2.6.20 è stato integrato direttamente nel kernel Linux.

Qemu

Qemu, è un hypervisor che permette di fare virtualizzazione di sistema o di processo ed è in grado di emulare molti processori e periferiche. Può essere usato come emulatore sfruttando la traduzione dinamica per trasformare le istruzioni dell'OS ospite in istruzioni della macchina ospitante, oppure si

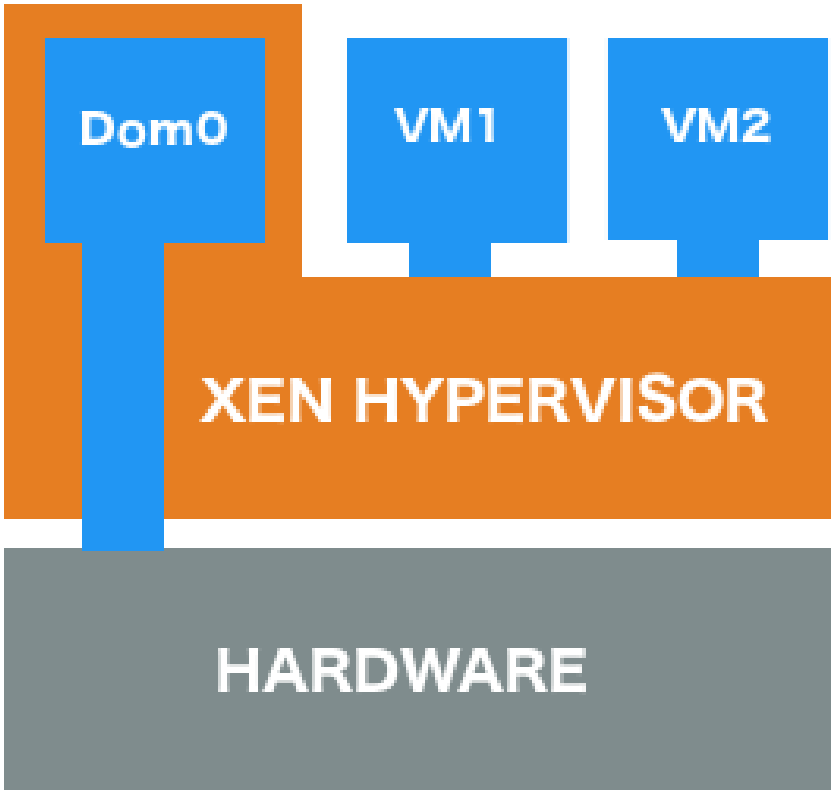

Figura 2.6: Architettura di XEN. Dom0 gestisce le macchine virtuali in esecuzione sull'hypervisor Xen.

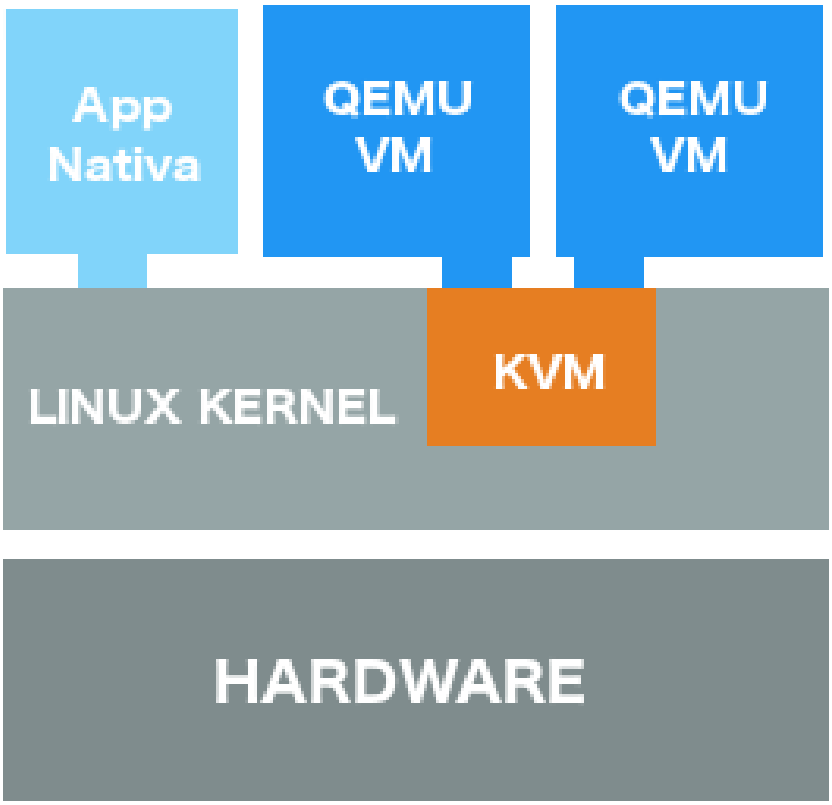

Figura 2.7: Il modulo KVM permette al kernel Linux di funzionare come un hypervisor.

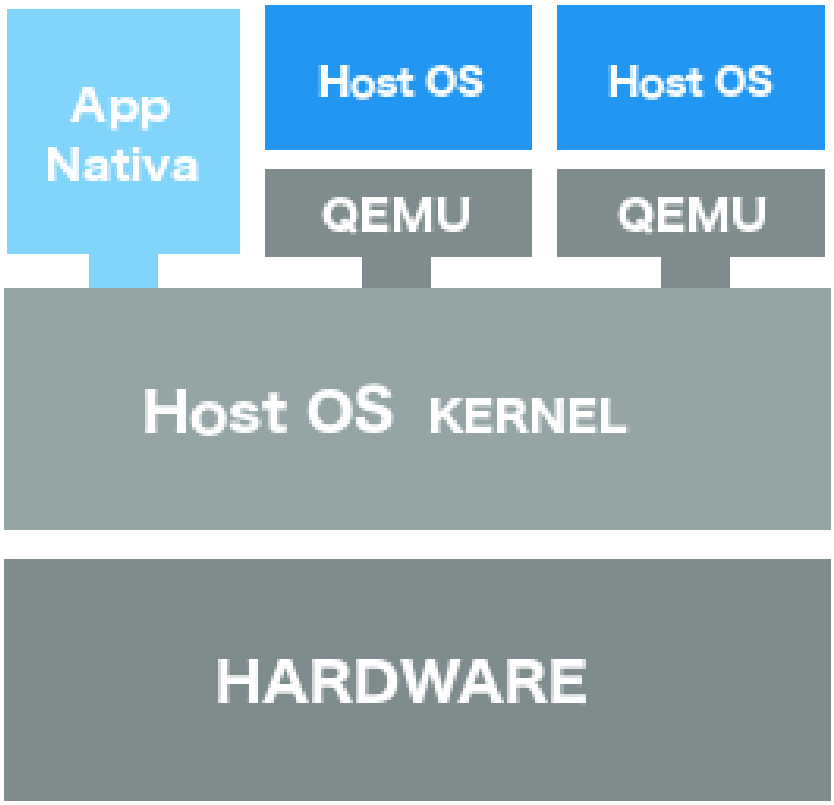

Figura 2.8: Qemu genera un hypervisor ad ogni esecuzione, sul quale si può eseguire un sistema operativ.o

può usare come virtualizzatore se eseguito con KVM o XEN permettendo di raggiungere prestazioni molto vicine a quelle native.

VirtualBox

VirtualBox è un hypervisor che tramite full-virtualization consente di creare macchine virtuali per diversi sistemi operativi. VirtualBox è compatibile con architetture x86 e permette di sfruttare il supporto hardware alla virtualizzazione. È disponibili in due licenze, una open source ed una

proprietaria di nome Oracle VM VirtualBox.

2.4 VDE

Virtual Distributed Ethernet (VDE) è un pacchetto open source che permette di creare un network virtuale di macchine virtuali. VDE consiste di switch, cavi e connettori per collegare varie macchine in esecuzione sullo stesso host, su diversi host in una rete o su host in reti diverse. È un progetto open source nato presso l'Università di Bologna dal Prof. Renzo Davoli e comprende: switch, plug, wire, cable e cryptcab.

vde_switch

È un applicativo che simula uno switch, come la sua controparte fisica ha dei socket ai quali è possibile collegare macchine virtuali o fisiche.

vde_plug

È una applicazione che virtualizza un connettore fisico.

vde_wire

È una qualsiasi applicazione in grado di trasportare flussi di dati, come SSH e netcat.

vde_cable

È composto da un vde_wire e due vde_plug, ossia da due interfacce agli estremi di un mezzo di trasmissione.

vde_cryptcab

È un vde_cable che offre funzionalità di criptaggio.

2.5 Virtualizzazione su Raspberry Pi

Il Raspberry Pi è un computer di dimensioni ridotte, grazie al suo prezzo di circa 30 dollari e delle specifiche hardware interessanti è un prodotto spesso utilizzato in ambito scolastico come base per progetti "fai da te" anche piuttosto complessi. È stato sviluppato dalla Raspberry Pi Foundation, un'organizzazione no-profit supportata dall'Università di Cambridge e dalla Broadcom. Generalmente integra sulla stessa scheda una CPU ARM, GPU, RAM, connettori USB, ethernet e output video/audio; ad oggi ne esistono 4 maggiori versioni: Raspberry Pi 1, 2, 3 e zero.

Raspberry Pi 1

La prima prima versione di RaspberryPi è nata nel 2012, si basa su una architettura ARMv6(32-bit), con una CPU 700 MHz single-core ARM1176JZF-S e 512Mb di RAM.
Ne esistono diversi modelli con caratteristiche leggermente differenti (A, B, A+, B+).

Raspberry Pi 2 model B

La seconda versione di RaspberryPi è stata prodotta nel 2015 e si basa su una architettura ARMv7(32-bit). Rispetto alla versione 1 comprende una CPU quad-core da 900 MHz 32-bit ARM Cortex-A7 e 1Gb di RAM.

Raspberry Pi 3 model B

La terza versione di RaspberryPi è stata prodotta nel 2016, si basa su una architettura ARMv8(32/64fd-bit), con una CPU da 1.2 GHz 64-bit quad-core ARM Cortex-A53 e 1Gb di RAM. Rispetto alle versioni precedenti integra un modulo Wi-Fi 802.11n e Bluetooth 4.1.

Raspberry Pi Zero

La versione chiamata "zero" è stata prodotta nel 2015 e si basa su una architettura ARMv6(32-bit), con una CPU da 1 GHz ARM1176JZF-S single-core e 512Mb di RAM. Differisce dalle altre versioni per il prezzo limitato di 5 dollari.

Virtualizzazione

Considerando l'architettura ARM e le risorse limitate del Raspberry si valuta ora il supporto delle tecnologie principali:

- Xen ha rilasciato una versione per ARMv7 quindi non supportata dal Raspberry Pi 1 che si basa su una architettura ARMv6.
- Di KVM per ARM esiste un progetto in fase di sviluppo preliminare ed una soluzione che ne consente l'esecuzione su un processore ARM dedicando uno o più core all'hypervisor KVM. Tuttavia la soluzione che prevede l'isolamento di almeno un core per KVM non è utilizzabile poiché il Raspberry Pi 1 è single-core.
- Qemu, è disponibile per sistemi Debian, quindi anche Raspbian, e consente di effettuare virtualizzazione ma con performance piuttosto limitate vista la capacità elaborativa del Raspberry e gli alti costi di emulazione.
- VirtualBox, non è attualmente compatibile con qualunque Raspberry poiché supporta solo piattaforme x86.

Capitolo 3

Introduzione al contributo originale

3.1 Struttura del sistema

Il sistema è una rete virtuale di VM su Raspberry per simulare un sistema distribuito. Esso è composto da diversi Raspberry collegati fisicamente attraverso uno switch ciascuno dei quali in grado di eseguire una o più macchine virtuali collegabili ad una o più reti virtuali.
Lo scopo del progetto è costruire un sistema fisicamente distribuito da utilizzare in presentazioni pubbliche per dimostrare il funzionamento e l'utilità di VDE, quindi di dimensioni ridotte e con tempi di avvio delle macchine virtuali brevi.

3.2 Setup del sistema

Setup Rete

L'hardware della rete fisica è costituito da uno switch con collegati 3 RaspberryPi 1 in diversi socket. Lo switch a sua volta è collegato tramite ethernet ad un computer dal quale si controlla l'intero sistema. I Raspberry

ed il computer formano una rete fisica e devono quindi avere indirizzi IP diversi, possibilmente statici per cui anche facilmente memorizzabili.

Setup Macchine Virtuali

Dopo aver correttamente settato la rete si passa all'avvio delle macchine virtuali da connettere. Collegandosi tramite SSH a qualsiasi Raspberry è possibile eseguire e controllare delle macchine virtuali con il seguente comando:

```
qemu-system-i386 -m 128 -kernel ~/dogebian/bzImage
-append "console=ttyS0" -net
 -k en-us -nographic
 nic,model=virtio,macaddr=MACADDR
-net vde,sock=vxvdex://
```

Se non specificato l'attributo "macaddr" Qemu utilizza il valore di default come mac address, per ogni VM bisogna quindi specificare valori diversi per gli indirizzi mac. Lo script "startvm" fornito nell'immagine prodotta consente di eseguire macchine virtuali con mac address randomici.

Dopo l'avvio si assegnano gli indirizzi IP a ciascuna VM, si lavorerà configurando le VM che si trovano sui 3 diversi Raspberry perché abbiano come indirizzo ip 10.1.1.1, 10.1.1.2 e 10.1.1.3. Per fare questo è sufficiente adattare il seguente comando:

```
ipconfig eth0 10.1.1.1 up
```

È possibile provare testare il funzionamento dell'infrastruttura per utilizzando ping per verificare che le macchine siano effettivamente raggiungibii tramite la rete:

```
ping 10.1.1.1
```

A questo punto il sistema è configurato per potervi eseguire test di rete o provare applicazioni.

3.3 Utilizzo del sistema

Una volta configurata la rete di macchine virtuali è possibile eseguire applicativi di rete, in particolare ci si interesserà di alcune utility di rete: ping, netcat e iperf.

Ping

È un'utility che tramite l'invio di pacchetti ICMP permette di verificare la raggiungibilità di un host in rete. Utilizzando il comando ping è possibile verificare che un host in una rete sia raggiunbile e conoscere il tempo di risposta in millisecondi. Dopo aver avviato la VM sul Raspberry 1 tramite SSH è possibile eseguire il comando ping per verificare la raggiungibilità della VM sul Raspberry2:

```
$ ping 10.1.1.2
64 bytes from 10.1.1.2: icmp_seq=0 ttl=64 time=0.039 ms
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.120 ms
```

Scollegando il Raspberry2 dalla rete eseguendo ping si riceve un errore

```
$ ping 10.1.1.2
64 bytes from 10.1.1.2: icmp_seq=0 ttl=64 time=0.039 ms
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.120 ms
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
```

Configurando il Raspberry3 con l'indirizzo ip 10.1.1.2 si nota che "ping" torna ad eseguire con successo

```
$ ping 10.1.1.2
64 bytes from 10.1.1.2: icmp_seq=0 ttl=64 time=0.039 ms
```

```
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.120 ms
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
64 bytes from 10.1.1.2: icmp_seq=4 ttl=64 time=0.059 ms
64 bytes from 10.1.1.2: icmp_seq=5 ttl=64 time=0.080 ms
```

Netcat

Netcat è una utility di comunicazione remota a linea di comando basata sul protocollo TCP o UPD ed è utilizzabile anche su reti virtuali. Tramite SSH sul Raspberry2 si avvia un server "netcat" in ascolto sulla porta 2389:

```
raspberry2$ nc -lp 2389
```

Dalla macchina Raspberry1 avviando una connessione netcat è possibile collegarsi al Raspberry2 ed inviare del testo:

```
raspberry1$ nc 10.1.1.2 2389
Hi from raspi1!


raspberry2$ nc -l 2389
Hi from raspi1!
```

Iperf

Iperf è una utilty di misurazioone del bandwidth nelle reti IP. Per usarlo è sufficiente avviare un server iperf su una macchina:

```
raspberry2$ iperf3 -s -p 12000
```

e connettersi con un client per testare la rete

```
raspberry1$ iperf3 -c 10.1.1.2 -p 12000 -t 20 -i 2
```

Capitolo 4

Analisi del contributo originale

4.1 Struttura concettuale del sistema

Il sistema consiste di una rete virtuale di macchine virtuali su una rete fisica di RaspberryPi 1.

4.2 Struttura concreta del sistema

Sono state realizzate due immagini di distribuzioni Linux disponibile all'indirizzo https://github.com/MisterDev/dogebian:

bzImage

È un kernel Dogebian ossia una immagine della distribuzione compilata con BuildRoot per questo progetto. Dogebian è configurata per poter essere eseguita efficientemente su Qemu, con tempi di boot ridotti e BusyBox con utility di rete come ping, iperf e netcat preinstallati.

raspbian.img

È un'immagine Raspbian con un SSH daemon configurato, Qemu e VDE preinstallati ed una immagine di Dogebian da eseguire come macchina vir-

tuale. Nella home è disponibile uno script bash startvm che permette di avviare VM Dogebian in modalità testuale.

4.3 Scelta del Sistema Operativo host

Il sistema operativo Host è quello che consente la creazione e gestione di macchine virtuali, deve quindi supportare Qemu, VDE e SSH. Le alternative prese in considerazione sono

Raspbian

Raspbian è la distribuzione per RaspberryPi più usata. L'immagine della versione Jessie ha dimensioni di circa 1.3Gb e sfrutta apt come package manager integrato garantendo disponibilità di molte applicazioni fra le quali Qemu, VDE ed SSH.

piCore

piCore 8.0 è l'ultima versione di TinyCore per architettura ARM. TinyCore è una distribuzione Linux minimale (circa 15Mb) è attualmente in sviluppo e consiste di un file .iso che viene caricato completamente in RAM all'avvio garantendo tempi di boot ridotti.
TinyCore prevede un meccanismo di estensioni e fornisce un package manager chiamato tce, che godendo di meno contributi vanta meno disponibilità di pacchetti rendendo necessario compilare Qemu e VDE perché siano disponibili.

Per comodità è stato scelto Raspbian poiché più diffuso e familiare anche se durante il progetto sono state compilate le estensioni di Qemu e VDE per piCore a scopo di ricerca.

4.4 Scelta del Sistema Operativo guest

Il sistema operativo guest è quello che viene virtualizzato e costituisce un nodo della rete virtuale. Esso deve avere tempi di boot limitati se virtualizzato su Raspberry e deve supportare utility di rete come ping, netcat ed iperf. Le alternative considerate sono state: BuildRoot, TinyCore, BasicLinux3, Docker e la minimalizzazione di una distribuzione.

BuildRoot

BuildRoot è un tool per compilare immagini di sistemi Linux ampiamente customizzati utilizzato spesso per fare cross-compiling di sistemi embedded. Grazie all'ampia personalizzabilità e la capacità di generare sistemi minimali si è rivelata la tecnologia scelta.

TinyCore

TinyCore è una distribuzione Linux minimale correntemente in sviluppo. È costituita da un file .iso da 10Mb e tramite il meccanismo delle estensioni permette di installare molti programmi fondamentali. Nonostante vanti tempi di boot eccellenti in ambiente emulato, non ha performato altrettanto bene solamente se virtualizzato su Raspberry, per cui è stato bocciato.

BasicLinux3

BasicLinux è una distribuzione di dimensioni ridotte, infatti è costituito da un file .iso da 4.8M ed è la distribuzione che integra BusyBox più leggera e con tempi di boot minori trovata. Nonostante sia molto interessante è un progetto poco utilizzato e poco aggiornato, integra infatti un kernel 2.2.26 risalente al 2004 ed una versione di BusyBox molto vecchia. Per questi motivi, e poiché i plugin disponibili sono sparsi e poco documentati, sono state preferite altre soluzioni.

Docker

Docker è il popolare sistema per impacchettare ambienti di esecuzione di programmi e sistemi operativi in grado di renderli indipendenti dall'ambiente reale della macchina sulla quale si trovano. È stata tuttavia scartata come soluzione poiché richiede il supporto Docker per VDE.

Minimalizzazione di una distribuzione

Consiste nel lavorare su un'immagine di una distribuzione cercando di arrivare a tempi di boot brevi togliendo applicativi e modificando configurazioni. Non sono state fatte prove in merito poiché a livello teorico il grado di miglioramento dipende dalla conoscenza della distribuzione stessa scelta e quindi risulta essere la tecnica probabilmente più costosa in termini di tempo e sforzo.

4.5 Configurazione di BuildRoot

Con BuildRoot è possibile compilare distribuzioni Linux personalizzate modificando le configurazioni di build ed eseguendo il comando per avviare la compilazione.

1. Dopo aver scaricato BuildRoot in una cartella si scelgono le impostazioni di base per compilare la distribuzione come: i dettagli del bootloader, i file system supportati, le caratteristiche del kernel, ecc.. Partendo da una configurazione di default si ha a disposizione di un set di valori funzionante che prevede ottimizzazioni specifiche per la piattaforma desiderata. Si esegue quindi il seguente comando per ottenere la lista delle configurazioni di default per BuildRoot:

```
$ make list-defconfigs
```

Si utilizzata come base la configurazione "qemu_x86_defconfig" che comprendente i386 come architettura target, nessun bootloader poiché inutilizzato da Qemu, ecc..

```
$ make qemu_x86_defconfig
```

2. Dopo aver selezionato una configurazione di default è possibile personalizzare ulteriormente le impostazioni con il comando:

   ```
   $ make menuconfig
   ```

   In particolare si usano le seguenti impostazioni aggiuntive:

   (a) In "System Configuration" nel menù "Run a getty (login prompt) after boot" si cambiano i seguenti valori per avviare un getty sulla porta ttyS0 in modo da avere una console sulla stessa una volta eseguito su Qemu:
   - TTY port: ttyS0
   - BaudRate: 115200
   - Term environment variable: vt100

   (b) In "Target Packages" nel menù "Networking Applications" si selezionano le applicazioni da comprendere nella distribuzione:
   - iperf3
   - netcat

   (c) In "Filesystem images" si seleziona "initial RAM filesystem linked into linux kernel" per far si che il file system sia linkato all'interno dell'immagine generata del kernel.

3. Dopo aver scelto le impostazioni desiderate si avvia la compilazione, per ottenere nella cartella output/images un file bzImage che contiene il kernel ed un file system utilizzabili su Qemu.

   ```
   $ make
   ```

Capitolo 5

Valutazione del contributo originale

5.1 Analisi immagini dei sistemi operativi

Durante la scelta del Sistema Operativo ospite sono stati eseguiti dei test sui tempi di boot. In particolare si sono virtualizzati prima i sistemi operativi scelti su un computer desktop e successivamente su un RaspberryPi.

Virtualizzazione su Desktop

Per fare una analisi dei tempi di boot sono testate varie immagini utilizzando qemu-system-i386 in Ubuntu 16.04 su un MacBook Pro con un Intel Core i5 da 2.9Mhz e 8 Gb di RAM.

Distribuzioni	Tempo di Boot
TinyCore	14s
BasicLinux3	7s
Dogebian	20s

Tabella 5.1: Tempi di boot dei sistemi virtuali su Qemu in ambiente desktop

Dai test in ambiente Desktop sono stati rilevati tempi di boot interessanti.

Virtualizzazione su Raspberry Pi

Le stesse distribibuzioni sono state poi testate virtualizzate su Raspberry Pi, i tempi di boot sono risultati analoghi su piCore e Raspbian.

Distribuzioni	Tempo di Boot
TinyCore	20s
BasicLinux3	7s
Dogebian	7m

Tabella 5.2: Tempi di boot dei sistemi virtuali su Qemu su Raspberry Pi

Virtualizzando su Raspberry Pi si nota un peggioramento delle performance per TinyCore rispetto all'ambiente Desktop, BasicLinux3 triplica i tempi di boot rimanendo entro tempistiche incoraggianti come Dogebian che aumenta di molto il tempo richiesti pur restando in valori ragionevoli.

Analisi sulle dimensioni

Nonostante non fosse una caratteristica critica si è prestata cura alle dimesioni delle varie immagini. Come si può vedere, grazie a BuildRoot, è stato possibile ottenere una distribuzione di dimensioni veramente limitate.

Distribuzioni	Dimensioni
TinyCore	10Mb
BasicLinux3	4.8Mb
Dogebian	4.8Mb

Tabella 5.3: Dimensioni delle immagini dei sistemi operativi

Conclusioni

In questo elaborato è stato descritto il processo di analisi e progettazione di un sistema distribuito in scala ridotta per presentazioni pubbliche che verrà utilizzato per dimostrare il funzionamento di VXVDE ad una conferenza. È stata fatta una introduzione ai concetti base della virtualizzazione e sono state descritte alcune delle tecnologie principali con un focus sulla virtualizzazione su Raspberry Pi 1. Si sono elencate le varie soluzioni valutaate e le motivazioni che hanno portato a scegliere di virtualizzare Dogebian su Raspbian con Qemu, e come si è arrivati ad ottenere un sistema di dimensioni ridotte con tempi di inizializzazione di circa 1 minuto. Il progetto in sè non pone le basi per grandi miglioramenti, tuttavia è stata fatta una interessante analisi dell'attuale stato della virtualizzazione su Raspberry e sono state prodotte delle immagini di distribuzioni con tempi di boot soddisfacenti in ambiente virtuale.

Bibliografia

[1] James E. Smith and Ravi Nair. “Virtual Machines” - Versatile Platforms for Systems and Processes. Margan Kaufman, 2005.

[2] Barham P., Dragovic B., Frases K., Hand S., Harris T., Ho A., Neugebauer R., Pratt I., Warfield A. “Xen and the art of virtualization”, Proceedings of the 19th ACM Symposium on Operating Systems Principles - SOSP03, October 2003.

[3] Oracle. “The Most Complete and Integrated Virtualization: From Desktop to Datacenter” An Oracle White Paper October. 2010.

[4] VMWare Inc., A performance Comparison of Hypervisors. 2007.

[5] “BasicLinux home page”. http://distro.ibiblio.org/baslinux. Ultima visita: 30 Novembre 2016.

[6] “Building your own TCL Extension”. https://github.com/puppetlabs/Razor-Microkernel/wiki/Building-your-own-TCL-Extension. Ultima visita: 30 Novembre 2016.

[7] “QEMU/Installing QEMU”. https://en.wikibooks.org/wiki/QEMU/Installing_QEMU. Ultima visita: 30 Novembre 2016.

[8] “Review of piCore, a 25MB OS for the Raspberry Pi”. https://www.maketecheasier.com/review-of-picore. Ultima visita: 30 Novembre 2016.

[9] “TinyCore Wiki”. http://wiki.tinycorelinux.net. Ultima visita: 30 Novembre 2016.

[10] “VirtualSquare Wiki”. virtualsquare.org. Ultima visita: 30 Novembre 2016.

[11] “Icons8”. icons8.com. Ultima visita 30 Novembre 2016

Ringraziamenti

Ringrazio moltissimo il Prof. Renzo Davoli per l'aiuto ed il tempo gentilmente dedicatomi.

Ringrazio la mia famiglia, che spesso do per scontata ma di cui sono davvero grato.

Ringrazio i miei colleghi universitari, di lavoro ed il doge team per il tempo prezioso passato insieme.

Ringrazio i miei amici per essere gioia nella vita.