

# Virtualizzazione

Approcci software ed hardware, considerazioni architetturali

Paolo Campegiani

<campegiani@scuolaiad.it>

Scuola IaD

Linux Day - 28 Ottobre 2006

- Tecniche di virtualizzazione e paravirtualizzazione
- Virtualizzatori: QEMu e VMware
- Paravirtualizzatori: Xen e OpenVZ
- Alta affidabilità
- Sicurezza



- Comprendere come opera un virtualizzatore
- Utilizzare operativamente i vari software
- Valutare le implicazioni di sicurezza e di affidabilità



# Everybody loves virtualization

E' uno dei settori a maggior sviluppo nell'informatica:

- Mercato di 18 mld di dollari per il 2007 (IDC)
- Tutti i grandi player hanno una loro strategia (IBM, Red Hat, SuSE, Microsoft, Apple, ...)
- AMD e Intel hanno processori virtualization-aware
- Fattore chiave di RHEL 5.0/SuSE 10

E' solo un hype?



# From the dawn of time it came

- Il primo sistema che offriva la virtualizzazione era l'IBM S/360 (1964)
- Uno degli articoli fondamentali nel settore (Goldberg e Popek) è del 1974
- Nel 1998 nasce VMware, ovvero la virtualizzazione per sistemi x86

Perchè oggi ne parlano tutti?



Esistono molti tipi di virtualizzazione:

- La macchina virtuale Java che esegue codice Java
- I volumi esportati da una SAN
- Le risorse di sistema viste dai programmi applicativi
- Le topologie di rete più complesse

Cosa ha di particolare la virtualizzazione di sistema operativo?



# The case for virtualization

- Gestire dei sistemi legacy
- Consolidare più servizi in un'unico server
- Avere una piattaforma di test identica a quella di produzione
- Migliorare l'affidabilità

La virtualizzazione è quello di cui ho bisogno?



# What's in a name?

## Virtualizzazione - 1

E' il raggruppamento e l'astrazione di risorse in modo tale da mascherarne la natura fisica e i limiti agli utenti delle risorse.

## Virtualizzazione - 2

E' una metodologia per dividere le risorse in ambienti di esecuzione multipli.



- Consolidamento di server: più servizi di rete offerti da server oggi distinti vengono migrati su un singolo server
- Testing: è possibile duplicare un server di test in uno di produzione (o viceversa)
- Training: è possibile fornire un ambiente completo di studio



Esistono varie tecniche possibili:

- Emulazione
- Virtualizzazione
- Paravirtualizzazione



- Virtual Machine Monitor (VMM): la componente applicativa che realizza la virtualizzazione
- Hypervisor: la componente di sistema che realizza la virtualizzazione
- (Sistema) guest: il sistema virtualizzato
- (Sistema) host: il sistema reale



Secondo Popek e Goldberg le caratteristiche di un sistema perchè operi come un virtualizzatore sono:

- Fedeltà: il software viene eseguito dalla VMM in modo analogo ad una esecuzione via hardware
- Prestazioni: la maggior parte delle istruzioni del guest sono eseguite dall'hardware senza l'intervento della VMM
- Sicurezza: la VMM gestisce tutte le risorse



- Ogni processore multiprogrammabile ha dei livelli di privilegio: i programmi utente operano ad un livello basso, il sistema operativo ad un livello alto
- Le istruzioni non privilegiate possono essere eseguite direttamente senza l'intervento della VMM
- L'esecuzione di una istruzione privilegiata causa una eccezione che viene catturata (trap) dalla VMM che la emula ricorrendo ai servizi del sistema operativo host



# Trap and emulate: de-privileging

- Tutte le istruzioni privilegiate sono catturate (trap) perchè eseguite dalla VMM in un contesto non privilegiato
- In alcuni casi questo trap è scatenato dal sistema operativo host (istruzione di I/O), in altri deve essere la VMM a proteggere le proprie strutture dati (la VMM e il codice del guest sono per l'host lo stesso segmento dati se non si usa la segmentazione)



# Trap and emulate: shadow structures

- Lo stato reale dell'hardware è diverso da quello che il guest vede, in quanto il guest opera come se fosse privilegiato (es: i registri del processore)
- Compito della VMM è gestire queste differenze
- La VMM crea delle *shadow structures* derivandole dalle *primary structures* che il guest vede
- E' molto semplice da fare per i registri della CPU
- Le Page Table Entries (PTE) hanno una gestione ben più complessa



# Trap and emulate: memory traces

- 1 Il guest prova ad accedere ad una area di memoria
- 2 Il gestore dell'eccezione di paginazione dà il controllo alla VMM
  - 1 True page faults: viene propagata l'eccezione al guest (ad esempio, demand-paging)
  - 2 Hidden page faults: non c'è stata la violazione delle politiche di accesso alla PTE, quindi occorre modificare la shadow PTE. Il guest non viene informato



# Trap and emulate: pro e contro

- E' stato l'approccio standard al problema della virtualizzazione
- Verifica le condizioni di Popek e Goldberg
- E' inefficiente: ogni trap and emulate comporta un context switch



L'architettura x86 non consente di realizzare una virtualizzazione via trap and emulate in modo efficiente.

- Il guest può determinare il suo livello di privilegio corrente (CPL) leggendo gli ultimi due bit di `%CS`
- `popf` eseguita in modo non privilegiato non azzerava il flag IF (interrupt delivery)
- Nessuna di queste due istruzioni può scatenare una trap



I sistemi non virtualization friendly richiedono che il guest sia eseguito da un interprete e non dal processore fisico.

Un binary translator opera come un compilatore più che come un interprete.



# Opzioni per la binary translation

- Caso più fortunato: istruzione non privilegiata e virtualization friendly:
- Caso meno fortunato: istruzione non privilegiata ma critica (`mov %cs, ...`)
- Caso sfortunato: istruzione privilegiata



# Binary translation: pro e contro

- I più veloci virtualizzatori ricorrono tutti alla BT
- Permette di non modificare il sistema guest
- Ha una complessità di funzionamento elevata



E' la modifica del sistema operativo guest in modo tale che la parte che comunica con l'hardware venga modificata per interrogare il sistema host.

Le applicazioni non vengono alterate.



# Paravirtualizzazione: pregi e difetti

- Offre le maggiori prestazioni (perdite di performance di pochi punti percentuali)
- Richiede la modifica del sistema operativo guest



Sia Intel (VT-X) che AMD (SVM) hanno processori che forniscono un supporto alla virtualizzazione:

- Virtual Machine Control Block (VMCB) contiene una parte dello stato del guest
- `vmrun` carica lo stato nel VMCB ed esegue il guest
- L'esecuzione prosegue finchè non si raggiunge una condizione (programmabile dalla VMM) di exit, che viene gestita dalla VMM



- E' possibile virtualizzare anche quei sistemi operativi che non sono paravirtualizzabili
- Si tratta di un meccanismo simile al trap and emulate, e soffre degli stessi problemi (context switch)
- Le performance degradano all'aumentare del numero degli exit, che vanno ridotti al minimo
- Ad esempio, non si esce ad ogni `popf`, ma si mette una copia shadow di `%eflags` nella VMCB



Opera sia come virtualizzatore che come cross-virtualizzatore:

- Ogni istruzione originale da eseguire viene scomposta in micro operazioni
- Ogni micro operazione viene eseguita da un codice C
- La conversione avviene per blocchi (Translated Blocks)
- Occorre gestire il codice automodificante (più complicato su x86)



`addi r1,r1,-16`  $r1 = r1 - 16$

- 1 `movl_T0_r1`  $T0 = R1$
- 2 `addl_T0_im -16`  $T0 = T0 - 16$
- 3 `movl_r1_T0`  $r1 = T0$



## movl\_T0\_R1

```
void op_movl_T0_R1(void) {  
    T0 = env->regs[1];  
}
```

- `env->regs` è una struttura che contiene i 32 registri del processore PowerPC emulato
- Su x86, T0 è mappato sul registro `ebx`



- Emula x86, SPARC, ARM, PowerPC
- Gira su tutti i sistemi su cui gira Linux (o quasi, vedi x86\_64)
- E' costituito da una parte userland e una con un modulo di kernel, kqemu
- kqemu non è rilasciato sotto GPL
- Permette di emulare più schede di rete



- 1 Download di qemu e kqemu
- 2 `tar xfvz qemu*.tar.gz`
- 3 Spostare `kqemu*.tar.gz` nella directory di `qemu`
- 4 `tar xfvz kqemu*.tar.gz`
- 5 `./configure; make; make install`
- 6 `modprobe kqemu major=0` (major=0 solo per utenti Fedora/RH)
- 7 `chmod 666 /dev/kqemu` (anche via regole di `udev`)



Il tool `qemu-img` permette di creare un file sparso da utilizzare come disco per il sistema virtualizzato:

```
qemu-img create scilla.img 5GB
```

crea il file `scilla.img` sparso di 5GB.



# QEMU: operation starts!

Una semplice invocazione di qemu può essere:

```
qemu -hda scilla.img -hdb shared.img -cdrom  
/dev/cdrom -boot cdrom -usb
```

In cui come disco /dev/hda viene usato scilla.img, come disco /dev/hdb viene usato shared.img, come cdrom viene usato /dev/cdrom, il boot è appunto da cdrom e viene emulato anche l'usb.



All'interno della finestra del sistema emulato, è possibile invocare il monitor di QEmu, che consente di compiere alcune operazioni:

- Il monitor si invoca con `ctrl-alt-2`
- Si esce dal monitor con `ctrl-alt-1`
- Per espellere il cdrom: `eject cdrom`
- Per sostituire il cdrom: `change cdrom filename`
- C'è la completion tab e l'history
- Per inviare una sequenza specifica: `sendkey ctrl-alt-canc`



- Quello che per il sistema guest è un file system, per il sistema host è un file
- Il formato adottato da QEmu per il file è diretto ed immediato, quindi la copia di una immagine di sistema si fa usando cp
- E' possibile avere dei copy on write (COW)



QEmu ha due distinte modalità di emulazione della rete:

- La user mode network stack, in cui lo stack di rete viene emulato completamente via software
- Utilizzo del tap device (richiede root)

Nella modalità user, l'host opera come firewall e come server DHCP all'indirizzo 10.0.2.2. I client prendono gli indirizzi a partire da 10.0.2.15.



# QEmu: un caso semplice

Vogliamo configurare un guest che esporti un servizio di rete (HTTP)

```
qemu -hda cariddi.img -redir tcp:8080:10.0.2.15:80
```

In questo modo le connessioni sulla porta 8080 dell'host saranno riportate alla porta 80 del guest di indirizzo 10.0.2.15.



Vogliamo configurare due guest, che possano comunicare tra di loro ma non con l'host. I due guest hanno un disco proprio ed uno condiviso.

- `qemu -hda scilla.img -hdb shared.img -usb` opzioni di base
- `-net nic,macaddr=52:54:00:12:34:61` definisce una scheda di rete con il MAC Address riportato
- `-net user,hostname=scilla` definisce l'hostname dato dal DHCP
- `-net socket,listen=:10000` crea un punto di contatto tra i guest

Le opzioni `-net` definiscono quindi una VLAN. Si possono avere più VLAN.



Analogamente, l'altro guest sarà così configurato:

- `qemu -hda cariddi.img -hdb shared.img -usb`
- `-net nic,macaddr=52:54:00:12:34:60`
- `-net user,hostname=cariddi`
- `-net socket,connect=127.0.0.1:10000` si aggancia al punto di contatto creato

Risultato: i due guest avranno indirizzi 10.0.2.15 e 10.0.2.16, gli hostname dati e sono in comunicazione con loro ma non con l'host.



E' possibile collegare ogni guest solo all'host, usando /dev/tap per avere migliori prestazioni:

- 1 `tunctl -t tap1`
- 2 `ifconfig tap1 up`
- 3 `brctl addbr br1`
- 4 `brctl addif br1 tap1`
- 5 `ifconfig br1 172.25.0.1 netmask 255.255.255.0  
broadcast 172.25.0.255`
- 6 `qemu -hda cariddi.img -net nic,vlan=0 -net  
tap,vlan=0, script=qemu-ifup.sh`

Questo approccio richiede di lavorare come root.



## qemu-ifup.sh

```
#!/bin/sh
echo Executing qemu-ifup
/sbin/ifconfig $1 0.0.0.0 promisc up
echo Adding $1 to br1
/usr/bin/brctl addiff br1 $1
```

Non c'è più un server DHCP, quindi ogni guest ha una scheda di rete configurata in modo statico (`ifconfig eth0 172.25.0.x`).



Vogliamo sempre avere due guest, ognuno dei quali dotato di due interfacce di rete:

- La prima è attivata da un DHCP e viene utilizzata per le comunicazioni tra i due guest
- La seconda viene utilizzata per comunicare con l'host



# Qemu: complichiamo molto le cose /2

```
qemu -hda scilla.img -hdb shared.img
```

## Prima VLAN

- `-net nic,macaddr=52:54:00:12:34:61 -net user,hostname=scilla`
- `-net socket,vlan=0,listen=:10000`

## Seconda VLAN

- `-net nic,vlan=1,macaddr=52:54:00:12:34:71`
- `-net tap,vlan=1,script=qemu-tap.sh`
- `-net socket,vlan=1,listen=:10001`



# QEmu: complichiamo molto le cose /3

- `qemu -hda cariddi.img -hdb shared.img`
- `-net nic,macaddr=52:54:00:12:34:60 -net user,hostname=scilla`
- `-net socket,vlan=0,connect=127.0.0.1:10000`
- `-net nic,vlan=1,macaddr=52:54:00:12:34:70`
- `-net tap,vlan=1,script=qemu-tap.sh`
- `-net socket,vlan=1,connect=127.0.0.1:10001`



```
qemu-tap.sh
```

```
#!/bin/sh  
/sbin/ifconfig $1 172.20.0.1
```

Le schede di rete dei guest si attiveranno staticamente  
`ifconfig eth1 172.20.0.X.`



# QEmu: pregi e difetti

- Complesso da far funzionare (non abbiamo parlato della gestione del firewall o di VDE)
- Non è particolarmente veloce
- Molto compatto, il pacchetto è di meno di 2 MB
- Flessibile, emula anche architetture non x86



E' una società (sussidiaria di EMC2) che offre una famiglia di prodotti dedicati alla virtualizzazione:

- VMware player per eseguire macchine virtuali costruite da altri (gratuito ma non libero)
- VMware Server che permette di creare ed eseguire macchine virtuali (gratuito ma non libero)
- VMware GSX Server che è un sistema operativo ed un virtualizzatore (non gratuito)
- VMware Infrastructure che offre gestione centralizzata, affidabilità, live migration



- Disponibile per Windows e Linux (32/64 bit)
- Virtualizzatore per architetture x86
- Rilasciato gratuitamente un anno fa circa (risposta a Xen?)
- Ha un suo sistema di scripting



- 1 Scaricare il pacchetto (rpm/tar.gz) dal sito di VMware (100 MB circa)
- 2 Ottenere il numero di licenza
- 3 Installare il pacchetto
- 4 Eseguire `/usr/bin/vmware-config.pl` come root per configurare il sistema (necessario ogni volta che si aggiorna il kernel)
- 5 Gestire l'avvio (`chkconfig --level 345 vmware off`)



- Bridged networking: il guest è in bridge con l'host
- NAT: l'host opera come NAT per il guest
- Host-only networking

Queste modalità non sono equivalenti in termini di velocità!



I dischi virtuali di VMware sono dei file `.vmdk` (formato non ingenuo).  
Per creare un disco:

- `vmware-vdiskmanager -c` per creare il disco
- `-s <dim>` per specificare la dimensione
- `-a lsilogic` per specificare un adattatore di tipo LSILogic
- `-t 0` per avere un unico file sparso per il disco
- `diskname.vmdk` per specificare il nome del disco



La configurazione della macchina virtuale è un file di testo con estensione `.vmx`, con molti diversi parametri. E' possibile ad esempio definire più dischi collegati in multipath, usare l'accelerazione grafica 3D della scheda video dell'host, configurare più schede di rete con connessioni di tipo diverso.



# VMware: pregi e difetti

- E' il virtualizzatore più diffuso e più performante
- E' gratuito ma non libero
- E' un server che gira in background



E' un paravirtualizzatore sviluppato dall'università di Cambridge:

- Supporta architetture x86
- Windows viene virtualizzato con processori VT-x
- E' incluso in RHEL 5 e SuSE 10



- Ogni macchina virtuale prende il nome di domain
- Il dominio 0 è il sistema operativo host
- Nel dominio 0 gira il demone `xend`
- I domini sono gestiti via `xm`



Ogni dominio è gestito da un file di configurazione che permette di specificare:

- Dimensione della memoria
- Numero di processori virtuali
- Quali device a blocchi sono visti
- Quali dispositivi PCI sono assegnati

Inoltre via `xm` è possibile definire lo spazio di processore a disposizione di ogni dominio.



Le interfacce di rete che l'host e i guest vedono sono sempre i familiari ethN, ma ne vengono attivati altri:

- Per il dominio Y, la sua interfaccia di rete X-esima è mappata, per l'host, su vifY.X
  - ▶ eth0 dell'host è quindi vif0.0
  - ▶ eth0 del guest nel dominio 1 è vif1.0
  - ▶ eth1 del guest nel dominio 2 è vif2.1
- Le vif0.X sono ulteriormente collegate a vethX (visibili solo dall'host)
- Uso del bridging per rendere i vari domini visibili ed autonomi



# Xen: live migration

Xen consente di migrare un sistema guest da un nodo fisico ad un altro, mantenendo il guest sempre in funzione.

Un server Quake3 migra in modo quasi trasparente agli utenti (50 ms di finestra temporale in cui varia il jitter dei pacchetti)

Un server web sotto SpecWeb migra in circa 1 minuto, riducendo il throughput ma non il numero di client serviti.



E' la versione GPL su cui si basa Virtuozzo, il paravirtualizzatore di SWSOft:

- Il kernel del guest e quello dell'host sono modificati
- Funziona solo con kernel Linux
- Permette di definire dei Virtual Environments (VE)



Ogni VE è una entità separata dalle altre, con i propri:

- File: librerie, applicazioni, dati
- Utenti e gruppi, compreso root
- Spazio dei processi, compreso init
- Rete, compreso indirizzo IP e regole di firewalling e routing
- Dispositivi
- Oggetti IPC



Il punto di forza di OpenVZ è la gestione delle risorse. Ogni VE:

- Ha una quota di spazio disco
- E' eseguito da uno scheduler fair-share (a livello di VE)
- ha associati dei beancounter, che permettono di controllare altri tipi di risorse in modo granulare (`/proc/user_beancounter`)



- 1 Installare il kernel OpenVZ ed alcuni programmi di gestione
- 2 Disabilitare SELinux, modificare alcuni parametri via `sysctl.conf` e abilitare il connection tracking sul VE0 (ovvero l'host)
- 3 Riavviare
- 4 Attivare il servizio `vz`



- In ogni VE è possibile installare una distribuzione di Linux
- I template sono degli archivi che contengono il necessario per fornire una distribuzione minimale
- E' possibile creare nuovi template
- Ad un template sono associati dei metadati che lo descrivono



# OpenVZ: installare un template già fatto

- 1 Installare i software di gestione dei template (vzyum\*)
- 2 Installare il pacchetto definente il template scelto (vztmpl-nome)
- 3 Copiare il sistema minimale (nome-template.tar.gz) in  
`/vz/template/cache/`



Ad ogni VE viene associato un numero. Per creare il VE 100 con template centos-4-minimal e un dato indirizzo IP:

```
vzctl create 100 --ostemplate centos-4-minimal  
--ipadd 192.168.1.100
```

E' possibile specificare limiti sulle risorse, moduli di iptables accessibili, capabilities disponibili e salvare la configurazione del VE.



# OpenVZ: operare con i VE

- `vzctl start NUM`
- `vzctl exec NUM COMANDO`
- `vzctl enter NUM (per uscire, exit)`
- `vzctl stop NUM`
- `vzctl destroy NUM`



E' possibile trasferire un VE in funzione da un host ad un altro:

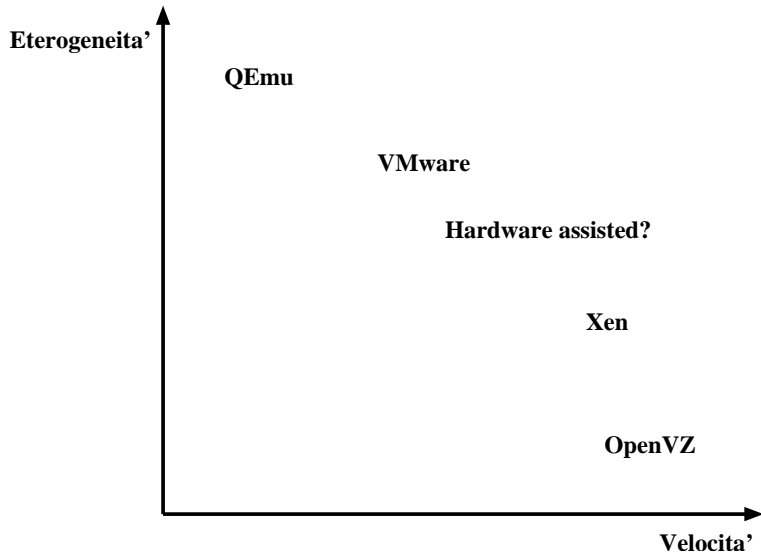
```
vzmigrate --online <host-destinazione> NUM
```

Il trasferimento dei dati avviene via ssh.

E' possibile eseguire questa operazione in più passi con un meccanismo di checkpoint.



# Quadrante



Una delle applicazioni tipiche dei sistemi virtualizzati è l'alta affidabilità:

Se i dischi dei guest sono file che vengono ospitati da una SAN, è possibile associare un guest ad un qualsiasi nodo fisico che veda la SAN.

In questo modo è possibile intervenire fisicamente sui nodi host senza pregiudicare il servizio offerto dai guest, riallocandoli dinamicamente.



In questo scenario, c'è un sistema guest ospitato su un sistema host, ed è disponibile un sistema host di riserva.

Non è un caso migliore di una normale configurazione attivo-passivo ad alta affidabilità.



In questo scenario ci sono due sistemi guest ospitati su due sistemi host, ognuno dei quali può farsi carico dell'esecuzione dell'altro guest.

Il vantaggio rispetto ad un sistema attivo-attivo può risiedere nel fatto che con due nodi fisici vengono erogati due servizi indipendenti l'uno dall'altro (ad esempio con sistemi operativi guest diversi).



# Affidabilità: scenario attivi-passivi

In questo scenario ci sono più nodi attivi ed un certo numero (anche uno solo) di nodi passivi, che possono essere anche in standby (senza guest). I nodi passivi acquisiscono i guest dei nodi attivi in situazione critica.

In questo modo si ha una alta affidabilità per una pluralità di servizi distribuendone il costo tra tutti, minimizzando i duplicati.



Le migrazioni delle immagini avvengono se c'è un *direttore d'orchestra* che:

- monitora sia i sistemi fisici che i guest
- determina che tipo di failure c'è stata e come reagire
- mette in pratica la decisione presa (anche riconfigurando l'infrastruttura di rete)
- opera in modo distribuito per non essere un singolo punto di fallimento



La regola d'oro della sicurezza dice che un sistema è sicuro tanto quanto la sua componente più debole.

Quanto è sicuro un sistema virtualizzato?



In un sistema virtualizzato sono presenti:

- il sistema host
- il virtualizzatore
- il sistema guest

La componente più insicura delle tre è quella che rende il sistema insicuro.



# Quanto è sicuro il vostro BIOS?

In una logica di virtualizzazione, l'host e il virtualizzatore sono simili al BIOS, in quanto offrono un servizio al sistema operativo (guest).

Il BIOS è una componente raramente aggiornata per la sicurezza, completamente embedded, che ha il controllo della macchina (almeno per un po' di tempo), certamente con dei banchi.



# Come rendere sicuro un sistema virtualizzato

Se non ci sono attacchi basati sulle debolezze del BIOS è perchè il BIOS è accessibile solo in locale.

Se l'host e la VMM sono accessibili solo in locale, non si riduce la sicurezza del guest.

E' una condizione forte, ma va considerata in fase di progettazione di una architettura virtualizzata.



Un possibile scenario di utilizzo:

- Pluralità di servizi erogati
- Variabilità estrema nei workload

Se i fornitori dei servizi sono virtualizzati, possono essere spostati/raggruppati in funzione del workload a cui fanno fronte, per soddisfare un Service Level Agreement.



- La virtualizzazione è un asset architetturale
- Permette il consolidamento dei sistemi, la loro protezione e compartimentazione
- Consente di superare le limitazioni dovute all'uso di un solo sistema operativo (compatibilità hardware, applicazioni, ...)



- La virtualizzazione è una metodologia complessa
- Le tecniche possibili sono diverse e hanno tradeoffs diversi
- E' comunque uno strato hw/sw in più, con quindi aggiuntive complessità di gestione
- Sicurezza ed affidabilità di queste soluzioni sono ancora dei territori inesplorati



In una fattoria:

- 1 Il foraggio viene dato alle mucche
- 2 Le mucche producono il latte
- 3 Il latte viene venduto ai clienti
- 4 Il proprietario della fattoria è contento



In un CED:

- 1 I sistemisti offrono risorse di calcolo agli applicativi
- 2 Gli applicativi sono usati dai clienti
- 3 I clienti pagano per l'uso
- 4 Il proprietario del CED è contento

Date più foraggio alle vostre mucche!



# Domande?

Queste slide sono rilasciate sotto licenza Creative Commons.

Le slide sono disponibili a:

[www.scuolaiad.it/virtualizzazione](http://www.scuolaiad.it/virtualizzazione)

Indirizzo e-mail dell'autore:

[campegiani@scuolaiad.it](mailto:campegiani@scuolaiad.it)

