Il mondo della virtualizzazione

Claudio Imbrenda

Linux Day 2008 PISA

25 ottobre 2008

Perché

Può essere utile avere una macchina virtuale:

1. Software scritto per un altro tipo di processore
2. Software scritto per un altro sistema operativo
3. Necessità di eseguire più sistemi operativi sulla stessa macchina
4. Necessità di separare diverse istanze dello stesso sistema operativo
5. Debug del sistema

Come

Gli approcci esistenti:

- interpretazione
- ricompilazione dinamica

Una domanda viene spontanea...

Perché mai dovrei *interpretare* o *ricompilare al volo* un se ho un processore che è capace di eseguire nativamente?

Niente di nuovo, in realtà...

La **virtualizzazione** è esattamente questo: esecuzione nativa da parte del processore del sistema virtuale.
Introdotta negli anni ’70 dalla IBM!

Alcune proprietà interessanti

Virtualizzazione significa quindi che:

- Il sistema eseguito nell'ambiente virtuale si deve comportare esattamente come se fosse eseguito su una macchina fisica eqivalente
- L'ambiente di virtualizzazione deve avere il completo controllo delle risorse virtualizzate
- Una percentuale statisticamente grande di istruzioni devono essere eseguite senza intervento dell'ambiente di virtualizzazione.

L'ultima proprietà non è in realtà fondamentale, essa in fatti garantisce solo l'*efficienza* della macchina virtuale.

Requisiti hardware

I requisiti di Popek e Goldberg sono i requisiti che una macchina deve avere per poter essere virtualizzabile. Per definire i requisiti per la virtualizzazione Popek e Goldberg introducono una classificazione delle istruzioni in tre gruppi:

Istruzioni privilegiate Sono quelle che generano una *trap* o un *fault* quando sono eseguite in modalità utente e che invece vengono eseguite correttamente in modalità supervisore.

Istruzioni sensibili di controllo Sono quelle che alterano lo stato globale del sistema.

Istruzioni sensibili di compotamento Sono quelle il cui risultato dipende dallo stato globale del sistema.

I teoremi di Popek e Goldberg

I risultati principali di Popek e Goldberg sono quindi:

Theorem

È possibile virtualizzare un'architettura se tutte le istruzioni sensibili sono privilegiate.

Theorem

Un'architettura è ricorsivamente virtualizzabile se:

1. *è virtualizzabile*
2. *è possibile costruire per essa una macchina virtuale che non abbia dipendenze di temporizzazione*

Che fine ha fatto la virtualizzazione?

Dopo gli anni 70 i computer sono diventati molto più comuni, ed economici. I principali motivi che spingevano la virtualizzazione sono venuti a mancare, fino alla fine degli anni ’90.

1. Software scritto per un altro tipo di processore
2. Software scritto per un altro sistema operativo
3. Necessità di eseguire più sistemi operativi sulla stessa macchina
4. Necessità di separare diverse istanze dello stesso sistema operativo
5. Debug del sistema

Ma c’è un piccolo particolare...

L’architettura purtroppo più diffusa (x86) ..
non è virtualizzabile!

Si trova sempre un modo..

- Spesso i sistemi operativi non dipendono dalle istruzioni sensibili.
- Spesso si può prevedere quando verranno usate istruzioni sensibili.
- Spesso si può barare!

Esecuzione in modo utente

Una tecnica di virtualizzazione prevede che tutto il sistema virtualizzato venga eseguito in modo utente, in modo che le istruzioni sensibili privilegiate generino delle trap, sperando che non vengano usate le istruzioni sensibili non privilegiate.
L'ambiente di virtualizzazione si occupa di gestire le trap per cercare di garantire la proprietà di equivalenza.
Questo garantisce la massima velocità ma non certo la massima generalità.

Esecuzione in modo utente con patching al volo

Come nel caso precedente, con la differenza che il codice viene analizzato e modificato prima di essere eseguito in modo da mantenere la proprietà di equivalenza.

- Richiede di conoscere dove e quando effettuare il patching.
- Ogni sistema da virtualizzare va trattato come caso a parte.
- Permette di virtualizzare anche sistemi che normalmente non sarebbero virtualizzabili.

Barare spudoratamente!

Possiamo **modificare il set di istruzioni**, in modo che tutte le istruzioni sensibili diventino privilegiate. Intel e AMD hanno fatto esattamente questo con le loro estensioni di virtualizzazione.
In questo modo si riesce a garantire sempre la proprietà di equivalenza.

Barare ancora più spudoratamente!

Infine possiamo modificare il sistema da virtualizzare in modo che si adatti in modo specifico al sistema di virtualizzazione.
In questo modo si viola la proprietà di equivalenza!

(ma si raggiungono ottime performance!)

In questo caso si parla di paravirtualizzazione.

Qualche esempio concreto

- Bochs (interprete)
- BasiliskII (interprete/ricompilatore)
- SheepShaver (interprete/ricompilatore)
- PearPC (interprete/ricompilatore)
- QEMU (ricompilatore/virtualizzatore)
- KVM (virtualizzatore/ricompilatore)
- VirtualBox (virtualizzatore/ricompilatore)
- Xen (virtualizzatore/paravirtualizzatore)

Xen

- necessita sempre di un sitema paravirtualizzato per accedere all'hardware (domain0)
- necessita delle estensioni di virtualizzazione per effettuare virtualizzazione

QEMU

- inizialmente era solo un emulatore
- aggiunta in seguito la possibilità di virtualizzare (modulo KQEMU)
- non fa uso di estensioni hardware per la virtualizzazione

KVM

- deriva da QEMU
- necessita delle estensioni di virtualizzazione
- la componente kernel-space è parte del kernel di linux

VirtualBox

- effettua esecuzione diretta quando possibile
- effettua ricompilazione al volo
- e modifica il codice durante le ricompilazioni, per non doverle rieffettuare

Dimostrazione pratica (VirtualBox)

Dimostrazione pratica (QEMU)

Domande?

Fine

Happy hacking!