

Virtualizzazione

Definizioni

- **Virtualizzare:** presentare all'utilizzatore una visione delle risorse del sistema diversa da quella reale
- **Obiettivo:** disaccoppiare il comportamento delle risorse HW e SW di un sistema di elaborazione, così come viste dall'utente, dalla loro realizzazione fisica.
- **Vantaggi**
 - Più SO sulla stessa macchina
 - Isolamento degli ambienti di esecuzione (sicurezza isolata)
 - Consolidamento HW (es. server farm)
 - Gestione facilitata delle macchine

Tipi di virtualizzazione

- **Linguaggio di programmazione:** es. interpreti con Virtual Machine (Java)
- **Virtualizzazione a livello di processo:** nei sistemi multitasking ogni processo vede una sua CPU privata e una serie di registri e memoria privata. Realizzato a livello kernel.
- **Virtualizzazione di sistema:** singola piattaforma HW condivisa da più sistemi operativi
 - *Virtual Machine Monitor* per la realizzazione delle macchine virtuali

Emulazione

- Esecuzione di programmi compilati per un certo insieme di istruzioni su un sistema di elaborazione dotato di un diverso set di istruzioni.
- Approcci:
 - **Interpretazione:** ogni istruzione convertita in una o più. Molto potente e generale, ma ha un sovraccarico molto elevato
 - **Compilazione dinamica:** si leggono interi blocchi di codice e si traducono per la nuova architettura e si eseguono. Prestazioni migliorate grazie a codice tradotto e ottimizzato e possibilità di bufferizzazione.
 - **Virtual PC:** esecuzione di sistemi operativi diversi in contemporanea
 - **QEMU:** permette di ottenere una nuova architettura, molto famoso per la velocità di emulazione, simile a VirtualPC
 - **MAME:** usato per emulare i videogiochi arcade delle sale giochi.

Realizzazione del VMM

Requisiti:

- Ambiente di esecuzione per i programmi sostanzialmente identico a quello della macchina reale
- Garantire un'elevata efficienza nell'esecuzione dei programmi (istr. Non privilegiate subito sull'HW senza passare da VMM)
- Stabilità e sicurezza dell'intero sistema

Livello del VMM:

- VMM di sistema: esegue direttamente sopra l'HW, funzionalità di virtualizzazione integrate in un sistema operativo leggero, che però ha anche tutti i driver delle periferiche (VMWARE ESX, XEN...)
 - VMM opera in stato supervisore, la MV in stato utente
 - **Ring deprivileging:** il SO ospitato esegue a livello più basso del previsto, non può eseguire istruzioni privilegiate
 - All'esecuzione di istr. Privilegiata, la CPU genera una trap e il controllo passa al VMM, che emula il comportamento dell'istruzione
 - **Architetture naturalmente virtualizzabili:** prevedono l'invio di notifica allo stato supervisore per ogni istruzione privilegiata eseguita non a livello supervisore (semplifica la costruzione di VMM)
 - **Architettura non virtualizzabili:**
 - Alcune istruzioni privilegiate eseguite in stato utente sono ignorate o causano crash.
 - **Ring aliasing:** alcune istruzioni non privilegiate permettono di accedere in lettura alcuni registri la cui gestione dovrebbe essere riservata al VMM.
 - **Soluzioni:**
 - **Fast Binary Translation:** il VMM scansiona il codice prima della sua esecuzione per sostituire a run time blocchi con istruzioni problematiche in blocchi equivalenti e contenenti istruzioni di notifica al VMM, è costoso ma ogni MV è una replica della macchina fisica.
 - **Paravirtualizzazione:** il SO offre al SO guest un'interfaccia virtuale alla quale i SO guest devono riferirsi per avere accesso alle risorse, devono essere modificati i kernel dei SO e queste chiamate si chiamano *hypercall*, ha prestazioni migliori ma richiede SO guest modificati.
 - **Ring compression:** SO ospitato e programmi ospitati hanno lo stesso livello di protezione e questo non permette di separare i due ambienti.
- VMM ospitato: installato come un'applicazione sopra ad un SO, opera nello spazio utente e accede all'HW tramite system call, peggiori performance

Modalità di dialogo per l'accesso alle risorse fisiche tra macchina virtuale e VMM:

- Virtualizzazione completa: le MV susano la stessa interfaccia dell'architettura fisica
- Paravirtualizzazione: il VMM presenta un'interfaccia diversa

Funzionamento VMM nell'architettura x86

Ring 0 al VMM e il resto a livelli più bassi, tecniche:

- **0/1/3** (non compatibile con sistemi a 64 bit)
 - Eccezioni gestite da VMM che le manda quasi inalterate al ring 1
- **0/3/3**
 - Meccanismo di controllo molto sofisticato perché non si possono generare eccezioni (si va vicini all'emulazione).

XEN

- VMM open source basato su **paravirtualizzazione**
- VMM virtualizza CPU memoria, dispositivi per ogni macchina
- Domain 0: VM privilegiata per il controllo della divisione delle risorse tra VM
- Il sw di controllo è eseguito nel domain 0: separazione meccanismi e politiche
- Le istruzioni privilegiate eseguono una hypercall al VMM
- I SO guest sono a ring 1
- I SO guest accedono alla memoria virtuale mediante la normale paginazione, le tabelle delle pagine sono accessibili readonly anche ai guest, in caso di update interviene il VMM
- Su ogni VM c'è un **processo balloon** che comunica con il VMM:VMM lo gonfia per richiedere pagine, SO guest lo riempie di pagine
- Virtualizzazione I/O
 - Back-end driver: per ogni dispositivo, il suo driver è isolato in una particolare VM (solitamente dom0) e accesso diretto all'hw
 - Front-end driver: ogni guest prevede un driver virtual semplificato che consente l'accesso tramite il backedn
 - Portabilità e isolamento, semplificazione VMM, ma necessita di comunicazione tra i driver
- Gestione interrupt: il vettore degli int punta direttamente alle routine del kernel guest, tranne il page fault, che richiede accesso a CR2 (ind. Cha ha causato il page fault), quindi si punta a codice xen