Sistemi Operativi M

Federico Andrucci

September 2021

Contents

1 Virtualizzazione

Virtualizzare un sistema (hardware e software) significa presentare all'uilizzatore una visione delle risorse del sistema diversa da quella reale. Ciò è possibile introducendo un **livello di indirezione** tra la vista logica e quella fisica delle risorse.

Quindi l'obiettivo della virtualizzazione è quello di disaccoppiare il comportamento delle risorsedi un calcolatore dalla loro realizzazione fisica. Quindi apparendo diverse da quelle effettive della macchina. Il software che si occupa di virtualizzare in parole semplici divide le risorse reali nel numero di macchine virtuali necessarie. Quindi ogni macchina virtuale avrà la sua CPU, GPU, RAM, ecc...

Esmpi di virtualizzazione:

- **Virtualizzazione a livello di processo:** i sistemi multitasking permettono l'esecuzione contemporanea di più processi, ognuno dei quale dispone di una macchina virtuale dedicata. Questo tipo di virtualizzazione viene realizzata dal kernel del sistema operativo.

- **Virtualizzazione della memoria:** in presenza di memoria virtuale, ogni processo vede uno spazio di indirizzamento di dimensioni indipendenti dallo spazio fisico effettivamente a dispozione. Anche questa virtualizzaione è realizzata dal kernel.
- **Astrazione:** un oggetto astratto (risorsa virtuale) è la rappresentazione semplificata di un oggetto (risortsa fisica), quindi esibendo le proprietà significative per l'utilizzatore e nascondendo i dettagli realizzativi non importanti

1.1 Virtualizzazione di un Sistema di Elaborazione

Tramite la virtualizzazione una singola piattaforma hardware viene condivisa da più elaboratori virtuali, ognuno gestito da un proprio sistema operativo. Il disaccoppiamento viene realizzato dal **Virtual Machine Monitor (VMM)**, il cui compito è quello di consentire la condivisione da parte di più macchine virtuali di una singola piattaforma hardware.

Quindi il **VMM** è il **mediatore unico** nelle interazioni tra le macchine virtuali e l'hardware, il quale garantisce: **isolamento tra le VM** e **stabilità del sistema**.

1.2 Tecniche del VMM

1.2.1 Emulazione

L'emulazione è l'insieme di tutti quei meccanismi che permettono l'esecuzione di un programma compilato su un determiato sistema di girare su un qualsiasi altro sistema differente da quello nel quale è stato compilato. Quindi vengono emulate interamente le singole istruzioni dell'architettura ospitata.

I vantaggi dell'emulazione sono l'interoperabilità tra ambienti eterogenei, mentre gli svantaggi sono le ripercussioni sulle performances.

Esistono principalmente due tecniche di emulazione: **interpretazione** e **ricompilazione dimanica**.

Interpretazione:
L'interpretazione si basa sulla lettura di ogni singola istruzione del codice macchina che deve essere eseguito e sulla esecuzione di più istruzioni sull'host virtualizzante. Produce un sovraccarico elevento in quanto potrebbero essere necessarie molte istruzioni dell'host per interpretare una singola istruzione sorgente.

Compilazione dinamica:
Invece di leggere una singola istruzione del sistema ospitato, legge interi blocchi di codice, vengono analizzati, tradotti per la nuova architettura, ot-

timizzati e messi in esecuzione. Il vantaggio in termini prestazionali rispetto all'interpretazione è notevolmente maggiore.

Ad esempio parti di codice utilizzati frequentemente vengono bufferizzati nella cache per evitare di doverli ricompilare in seguito.

.. .. ..

1.2.2 Realizzaione del VMM

Requisiti di Popek e Goldberg del 1974:

- **Ambiente di esecuzione per i programmi sostanzialmente identico a quello della macchina reale**: Gli stessi programmi che eseguono nel sistema non virtualizzato possono essere eseguiti nelle VM senza modifiche e problemi.
- **Garantire un'elevata efficienza nell'esecuzione dei programmi**: Il VMM deve permettere l'esecuzione diretta delle istruzioni impartite dalle macchine virtuali, quindi le istruzioni non privilegiate vengono eseguite direttamente in hardware senza coinvolgere il VMM
- **Garantire la stabilità e la sicurezza dell'intero sistema**: Il VMM deve sempre rimanere sempre nel pieno controllo delle risorse hardware, e i programmi in esecuzione nelle macchine virtuali non possono accedere all'hardware in modo privilegiato

Parametri e classificazione

- **Livello** nel quale è collocato il VMM:
  - **VMM di sistema**: eseguono direttamente sopra l'hardware del elaboratore (vmware, esx, xen, kvm)
  - **VMM ospitati**: eseguiti come applicazioni sopra un S.O. esistente (parallels, virtualbox)
- **Modalità di dialogo**: per l'accesso alle risorse fisiche tra le macchine virtuali ed il VMM:
  - **Virtualizzazione pura** (vmware): le macchine virtuali usano la stessa interfaccia dell'architettura fisica
  - **Paravirtualizzazione** (xen): il VMM presenta un'interfaccia diversa da quella dell'architettura HW

2 La protezione nei Sistemi Operativi

3 Programmazione Concorrente

4 Modelli di Interazione tra i Processi

Esistono 2 principali modelli di interazione tra i processi:

- Modello a **memoria comune** (ambiente globale, shared memory)
- Modello a **scambio di messaggi** (ambiente locale, distributed memory)

4.1 Modello a memoria comune

Il modello a memoria comune rappresenta la più semplice astrazione del funzionamento di un sistema in multiprogrammazione costituito da uno o più processi che hanno accesso ad una memoria comune.

Ogni applicazione viene strutturata come un insieme di componenti, suddiviso in due sottoinsieme disgiunti:

- **Processi** (componenti attivi)
- **Risorse** (componenti passivi)

Le Risorse rappresentatno un qualunque oggettim fisico o logico, di cui un processo necessita per portare a termine il suo compito. Le risorse vengono raggruppate in classi, dove una classe rappresenta l'insieme di tutte e sole le operazioni che un processo può eseguire per operare su risorse di quella classe,

Ovviamente ci deve essere la necessità di specificare quali processi ed in quali istanti possono accedere alla risorsa. Quindi il **meccanismo di controllo degli accessi** si occupa di controllare che gli accessi dei processi alle risorse avvengano correttamente.

Gestore di una risorsa:
Per ogni risorsa **R**, il suo gestore definisce, in ogni istante t, **l'insieme SR(t) dei processi che, in tale istante, hanno il diritto di operare su R**.

Classificazione delle risorse:

- Risorsa R **dedicata**: se SR(t) ha una caardianlità sempre ¡= 1
- Risorsa R **condivisa**: in caso contrario
- Risorsa R **allocata staticamente**: se SR(t) è una costante, quindi se SR(t) = SR(t0) per ogni t
- Risorsa R **allocata dinamicamente**: se SR(t) è funzione del tempo

Per ogni risorsa **allocata staticamente**, l'insieme SR(t) è definito prima che il programma inizi la propria esecuzione; il gestore della risorsa è il programmatore che, in base alle regole del linguaggio, stabilisce quale processo può vedere e quindi operare su R.

Per ogni risorsa **allocata dinamicamente**, il relativo gestore GR definisce l'insieme SR(t) in fase di esecuzione e quindi deve essere un componente della stessa applicazione, nel quale l'allocazione viene decisa a run-time in base a politiche date.

Quindi i principali compiti del Gestore delle risorse sono:

- mantenere **aggiornato** l'insieme SR(t) e cioè lo stato di allocazione della risorsa
- fornire i **meccanismi** che un processo può utilizzare per acquisire il diritto di operare sulla risorsa, entrando a far parte dell'insieme SR(t), e per rilasciare tale diritto quando non è più necessario
- implementare la **startegia** di allocazione della risorsa e cioè definire quando, a chi e per quanto tempo allocare la risorsa.