Cenni sulla virtualizzazione

Anno accademico 2012/13
Sistemi Concorrenti e Distribuiti
Tullio Vardanega

Astrazione

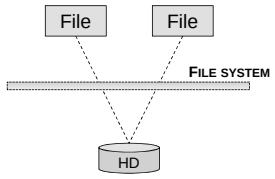
- Nascondere dettagli dell'implementazione per semplificare la vista logica dell'utente
  - Un tipo di dato astratto semplice viene sovrapposto alla complessità sottostante
    - Esempio: in UNIX tutto è *file*
- Ogni cambiamento sotto l'interfaccia di astrazione può causare ripercussioni sugli utilizzatori di quell'interfaccia
- Parola chiave: ***information hiding***

Virtualizzazione

- Realizzare una vista logica su una risorsa indipendente dalla sua vera natura
  - La virtualizzazione rimpiazza il reale
    - Esempio: il prerilascio virtualizza la modalità di esecuzione nella macchina di von Neumann
- La virtualizzazione si poggia sull'astrazione ma la rafforza perché si impegna a garantire sempre la vista logica promessa all'utente
- Parola chiave: ***encapsulation***

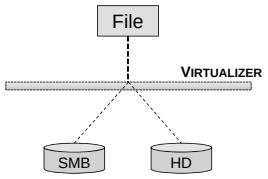
Esempio

Astrazione



Usi e contenuti diversi nella stessa astrazione di *file* che nasconde la struttura fisica sottostante

Virtualizzazione



Una specifica astrazione "*file*" è garantita indipendentemente dal supporto fisico sottostante

Cenni storici – 1

- Anni ’60, epoca *mainframe*
- HW scarsamente disponibile e molto costoso
- La virtualizzazione permette la condivisione trasparente delle poche risorse fisiche disponibili
  - Il *time sharing* virtualizza l’accesso alla CPU
  - La memoria virtuale supera i limiti fisici
- La virtualizzazione diventa così uno dei principi fondanti dell’informatica

Cenni storici – 2

- Anni ’80, passaggio ai *minicomputer* prima e ai PC poi
- Il problema della condivisione trasparente delle risorse di calcolo viene risolto in modo ricorrente dai S/O multiprogrammati
- L’interesse per lo sviluppo della virtualizzazione svanisce

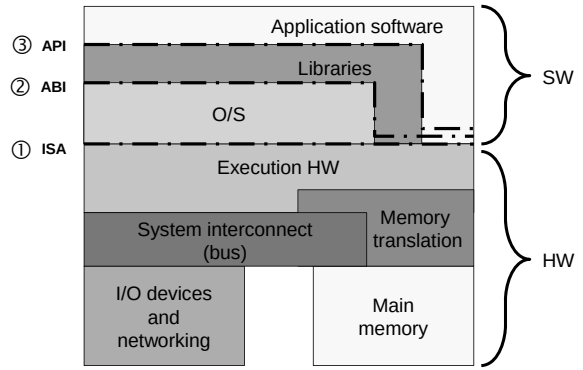
Cenni storici – 3

- Primi anni ’90, picco di attenzione per il calcolo parallelo e specializzato
- L’interesse per la virtualizzazione rinasce per rendere meno onerosa la programmazione dell’HW *special-purpose*
  - Specialmente i massively parallel processors
  - Nasce VMware Inc.

Cenni storici – 4

- Seconda metà anni ’90, grande diffusione dell’IT a supporto delle attività aziendali
- Sorge il problema dei costi di gestione e sotto-utilizzo di HW e SW eterogeneo
  - Al diminuire del costo unitario aumenta l’eterogeneità (classica legge della domanda)
  - Più costo di gestione per meno portabilità
    - Condividere HW e risorse di calcolo inutilizzate aiuta a ridurre i costi
    - La rivincita della virtualizzazione

Architettura e interfacce – 1



Architettura e interfacce – 1

- Tre punti di connessione
  - API, ABI, ISA
- Le interfacce di astrazione sono alla base dell'architettura classica dei sistemi di calcolo
- Ma ogni astrazione è fragile rispetto a variazioni nella natura e nel comportamento del livello sottostante

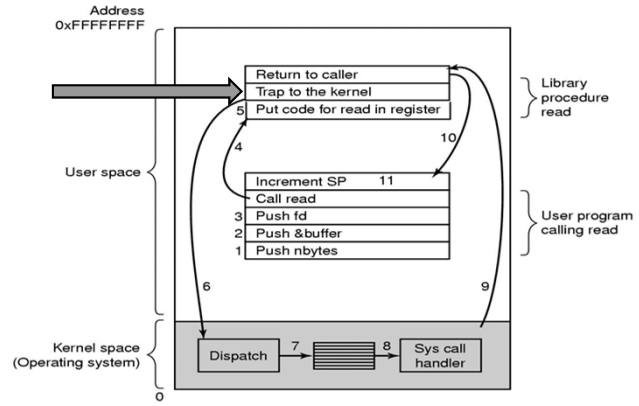
Architettura e interfacce – 3

- Cosa succede se cambia l'HW?
  - Se cambia l'ISA sono costretto a cambiare S/O per la fragilità di quella astrazione
  - Potrei essere anche costretto a cambiare in cascata ABI e API
- Per preservare il valore aggiunto dei livelli alti dobbiamo rafforzare l'astrazione con la virtualizzazione
  - Ma dobbiamo scegliere a che livello realizzarla

Principio base di virtualizzazione – 1

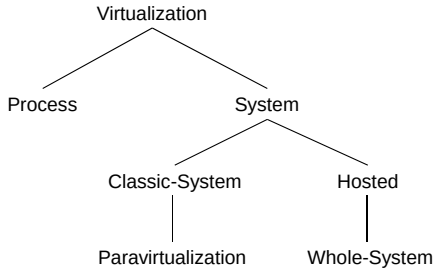
- Dalla fine degli anni '60 il «modo» di esecuzione è diviso in livelli di privilegio
  - L'ISA è accessibile al SW in sottoinsiemi («*ring*») concentrici più vicini al *core* al crescere del privilegio
  - Ogni tentativo di accesso a istruzioni HW a livello di privilegio superiore di quello del chiamante solleva una eccezione (*trap* HW)
  - L'innalzamento di privilegio è ottenuto tramite una istruzione speciale (*trap* SW)

Principio base di virtualizzazione – 2



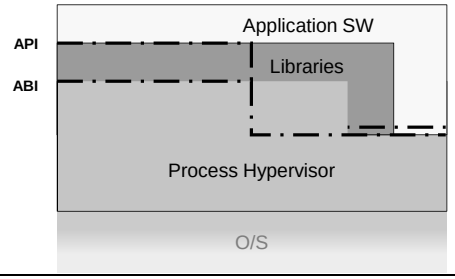
Virtualizzazione: tassonomia

- In base al livello di interfaccia sotto al quale si realizza la virtualizzazione



Process Virtualization – 1

- L'*hypervisor* fornisce una specifica ABI per le applicazioni
- L'unione tra l'*hypervisor* e i programmi su di esso eseguiti viene detto Virtual Machine (VM)
- La VM a livello di processo più comune è il S/O stesso!

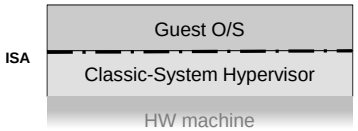


Process Virtualization – 2

- La *process VM* dispone di
  - Memoria virtuale
  - Strumenti di I/O astratti come *file* e *socket*
  - Tempo di CPU
- Un emulatore SW legge il *bytecode* dell' applicazione e la esegue istruzione per istruzione
  - «*Instruction interpretation and translation*»
  - Esempi: JVM, Wine

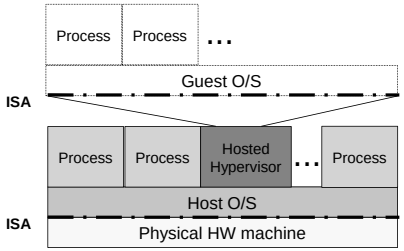
Classic-System Virtualization

- Nelle VM di tipo sistema la macchina è una ISA con periferiche associate
  - *Storage*, *network*, etc…
- Questa virtualizzazione riproduce tutto quello che serve al S/O ospite in modo identico a come sarebbe l'HW fisico
  - «*Guest OS de-privileging*»



Hosted Virtualization

- L'*hypervisor* è un processo come tutti gli altri
  - Alloca le risorse di memoria e *storage* necessarie richiedendole al S/O ospitante
- Comporta elevate penalità di esecuzione



Whole-System Virtualization

- Questa tecnica permette di virtualizzare architetture (ISA) diverse da quella ospitante
- Variante del tipo "*Hosted*"
  - Nel caso «*hosted*» le istruzioni HW a basso privilegio emesse dall'applicazione virtualizzata eseguono direttamente
    - Perché l'ISA virtualizzata è la stessa di quella fisica
  - Nel caso «*whole system*» serve un emulatore di ISA all'interno dell'*hypervisor*

Para-virtualization – 1

- Negli anni '80 scema l'interesse per la *system virtualization*
- Nelle architetture x86 vengono introdotte istruzioni macchina non virtualizzabili (!)
  - La loro esecuzione non genera *trap* HW
  - Conseguentemente l'*hypervisor* non si può accorgere del loro utilizzo

Para-virtualization – 2

- Viene allora definita una nuova interfaccia che richiede l'adattamento del S/O *guest*
- Il beneficio è una bassa penalità di esecuzione (ca. 1%)