Cenni sulla virtualizzazione

Anno accademico 2016/17
Sistemi Concorrenti e Distribuiti
Tullio Vardanega

Astrazione

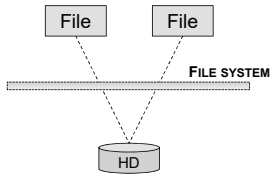
- Nascondere dettagli dell'implementazione per semplificare la vista logica offerta all'utente
  - Un tipo di dato astratto semplice viene sovrapposto alla complessità sottostante
    - Esempio: in UNIX tutto è *file*
- Parola chiave: ***information hiding***
- Punto debole
  - Variazioni sotto l'interfaccia di astrazione possono impedire la preservazione di quell'interfaccia

Virtualizzazione

- Realizzare una vista logica su una risorsa indipendentemente dalla sua vera natura
  - La virtualizzazione rimpiazza il reale
    - Esempio: il prerilascio virtualizza la modalità di esecuzione nell'architettura di von Neumann
- Parola chiave: ***encapsulation***
- Punto di forza
  - La virtualizzazione usa l'astrazione ma ne rafforza il valore impegnandosi a garantire sempre la vista logica esposta all'utente

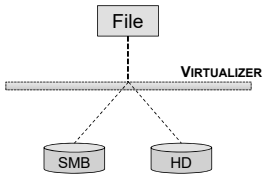
Esempio

Astrazione



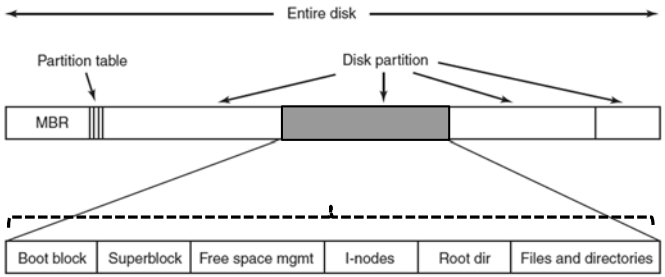
Usi e contenuti diversi nella stessa astrazione di *file* che nasconde la struttura fisica sottostante

Virtualizzazione



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

Astrazione di Sistema Operativo



- ***Boot block***: procedura di inizializzazione
- ***Superblock***: descrittore del resto della partizione
- ***I-nodes***: lista di tutti i descrittori (*i-node*)

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 («standardizzato») dai S/O multi-programmati
- L’interesse per lo sviluppo della virtualizzazione svanisce

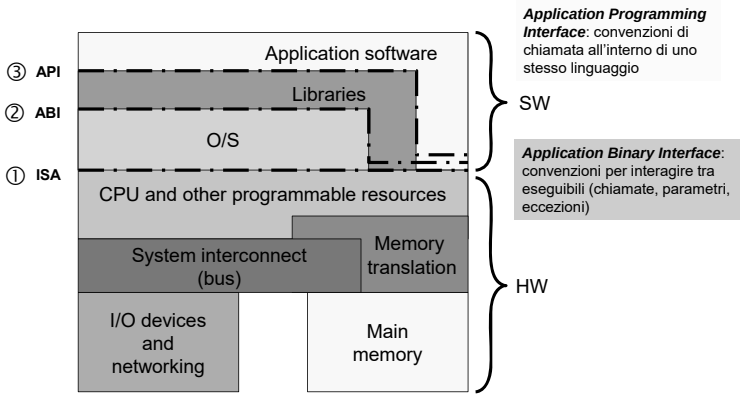
Cenni storici /3

- Primi anni ’90: picco di attenzione per il calcolo a parallelismo massiccio per applicazioni specializzate
  - Nasce il Transputer (*transistor & computer*), componente *general-purpose* antesignano dei *massively parallel processors*
- L’interesse per la virtualizzazione rinasce per agevolare la preservazione di applicazioni destinate ad HW *special-purpose*
  - Nasce VMware Inc.

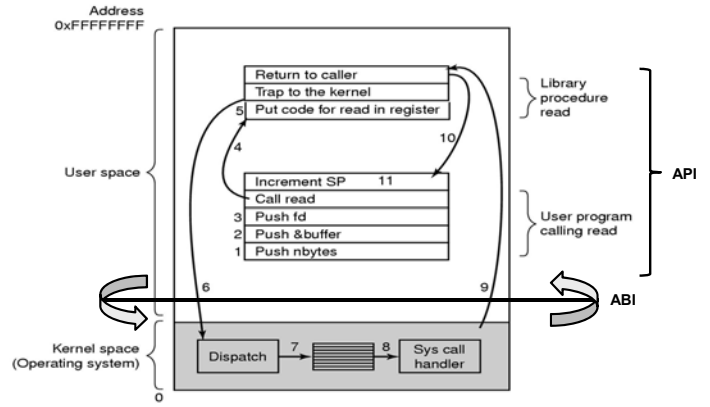
Cenni storici /4

- Seconda metà anni ’90, enorme diffusione dell’IT a supporto delle attività aziendali
  - Al diminuire del costo unitario aumenta l’eterogeneità (classica legge della domanda)
  - Con *server* dedicati si ha maggior costo di gestione, meno portabilità e sotto-utilizzo di HW e SW eterogenei
- Rinascita della virtualizzazione
  - Condividere HW e risorse di calcolo inutilizzate aiuta a ridurre i costi

Architettura e interfacce /1



Dove interviene l’ABI: esempio



Architettura e interfacce /2

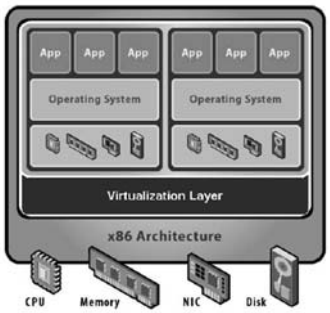
- Tre punti di connessione per servizi a valore aggiunto
  - API, ABI, ISA
- Le interfacce di astrazione sono alla base dell’architettura classica dei sistemi di calcolo
- Ma ogni astrazione è fragile rispetto a variazioni nella natura e nel comportamento del livello sottostante

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

Punto di arrivo

L'attenzione si sposta sull'isolamento

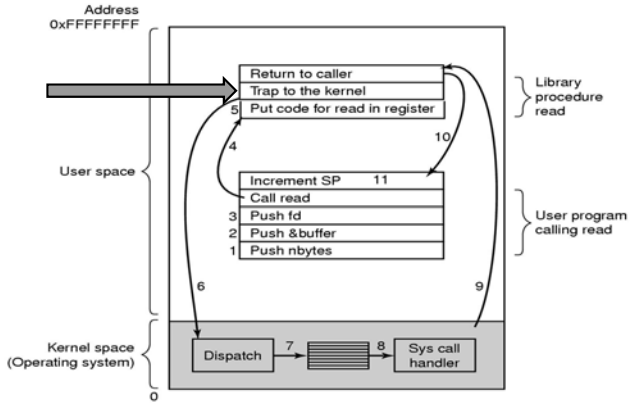


- Hardware-Level Process Abstraction: CPU, memory, chipset, I/O devices, etc.
  - Virtual NIC instead of sockets
  - Virtual disk instead of file system
  - Hardware state becomes software state
- Virtualization Software
  - Hardware and software decoupled

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)

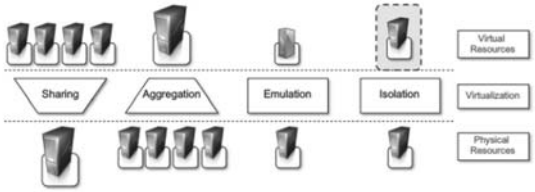
Principio base di virtualizzazione /2



Virtualizzazione: obiettivi

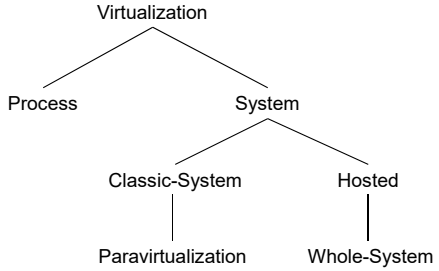
Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications without affecting other users' applications

- various functions enabled by managed execution
  - sharing (e.g. server consolidation)
  - aggregation (e.g. cluster management software)
  - emulation (e.g arcade-game emulator)
  - isolation ⇒ no interference between multiple guest

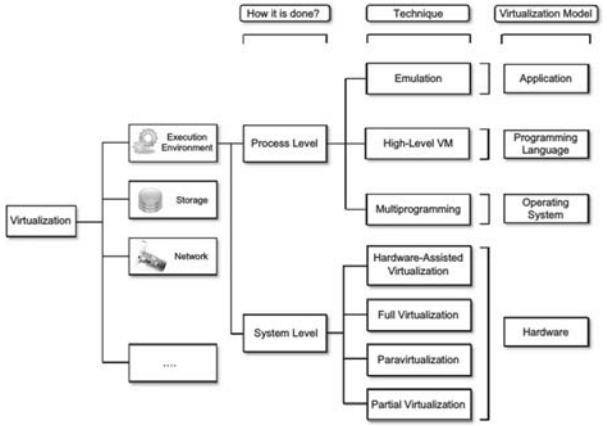


Virtualizzazione: tassonomia /1

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

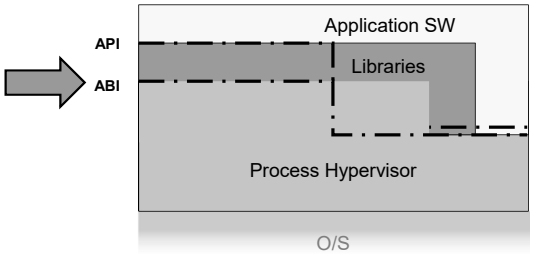


Virtualizzazione: tassonomia /2



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!

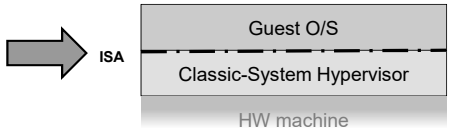


Process Virtualization /2

- La *process VM* dispone di
  - Memoria virtuale
  - Strumenti di I/O astratti come *file* e *socket*
  - Tempo di CPU
- Una «capsula SW» esegue l' applicazione ospite istruzione per istruzione
  - Per esempio tramite «*instruction interpretation and translation*» → JVM (*bytecode*)
  - Oppure per esecuzione diretta → Wine (binario)

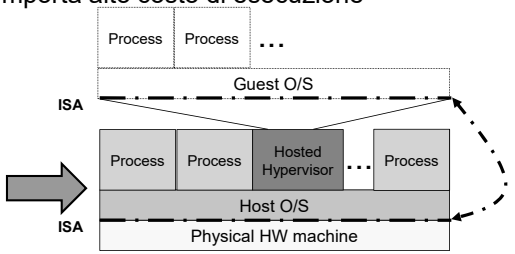
Classic-System Virtualization

- Nelle VM di tipo sistema la «macchina» esposta è una ISA con periferiche associate
  - *Storage*, *network*, etc…
- Questa virtualizzazione riproduce tutto quello che serve al S/O ospite esattamente 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
  - Medesimo ISA (esecuzione diretta), diverso ISA (interpretazione)
- Comporta alto costo di esecuzione



Whole-System Virtualization

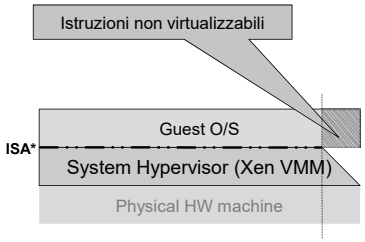
- Questa tecnica permette di virtualizzare architetture (ISA) diverse da quella ospitante
- Variante del tipo "*Hosted*"
  - Nel caso «*hosted*» le istruzioni HW a basso privilegio emesse dall'applicazione virtualizzata eseguono direttamente
    - Perché l'ISA virtualizzata è la stessa di quella fisica
  - 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 (*hypercall* API) che richiede l’adattamento del S/O *guest*
- Il beneficio è una bassa penalità di esecuzione (ca. 1%)



Visione d’insieme