Cenni sulla virtualizzazione

Anno accademico 2019/2020
Sistemi Concorrenti e Distribuiti
Tullio Vardanega

Astrazione

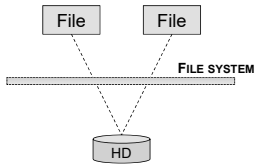
- Nascondere dettagli dell'implementazione per semplificare la vista logica offerta all'uso
  - Esporre un **tipo di dato astratto** (semplice e diretto) invece della complessa realtà sottostante
  - Esempio: in UNIX/Linux, l'astrazione base è il *file* che noi usiamo tramite una interfaccia ben definita
- Parole chiave
  - ***Information hiding***, ***well-defined interface***
- Punto debole
  - L'interfaccia di astrazione è fragile rispetto a variazioni che intervengano sotto di essa

Virtualizzazione

- Garantire una vista logica su una risorsa, indipendentemente dalla sua natura concreta
  - La virtualizzazione rimpiazza il reale esponendo una interfaccia di astrazione
    - Esempio: il prerilascio virtualizza la modalità di esecuzione nell'architettura di von Neumann
- Parola chiave
  - ***Encapsulation***
- Punto di forza
  - La virtualizzazione usa l'astrazione, rafforzandone il valore impegnandosi a garantire sempre la vista logica esposta all'utente

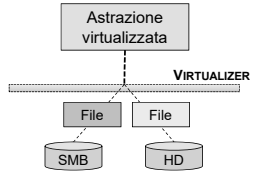
Esempio /1

Astrazione



Permettere usi e contenuti diversi sulla stessa astrazione realizzandola senza esporre la struttura fisica sottostante

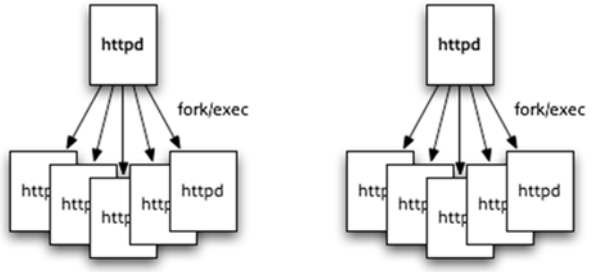
Virtualizzazione



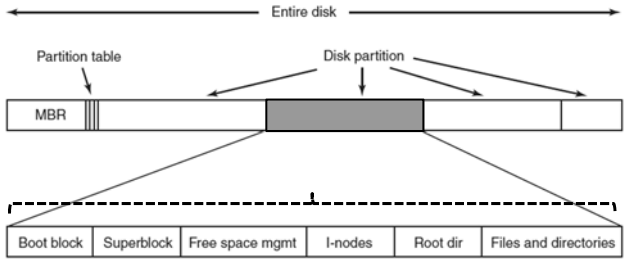
Offrire una specifica interfaccia "virtuale", anche se eventualmente basata su diverse astrazioni di livello inferiore

Esempio /2

- Uso basico dell'astrazione di processo UNIX



Astrazione di Sistema Operativo /1



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

Astrazione di Sistema Operativo /2

- Conoscere una particolare astrazione di S/O (cioè la sua concretizzazione a *run time*) permette di manipolarla come una entità vera e propria
  - Copiarla
  - Spostarla
  - Cancellarla
  - Fermarne e riprenderne a piacere l'esecuzione
- Tutto ciò che serve è un modo per trattare quella rappresentazione secondo il suo significato

Un po' di storia /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 della RAM
- La virtualizzazione diventa così uno dei principi fondanti dell'informatica

Un po' di storia /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
- Diminuisce l’interesse per l’ulteriore sviluppo della virtualizzazione

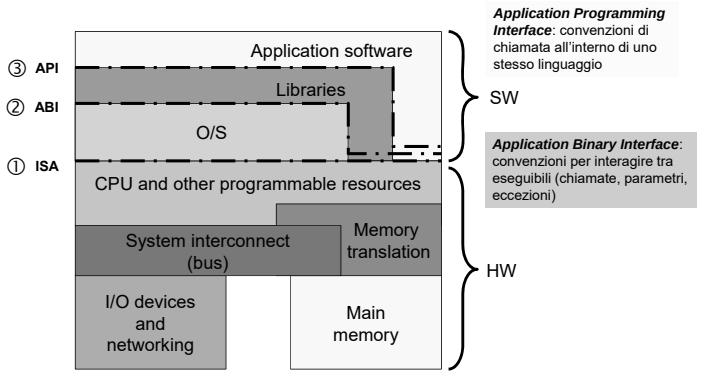
Un po' di storia /3

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

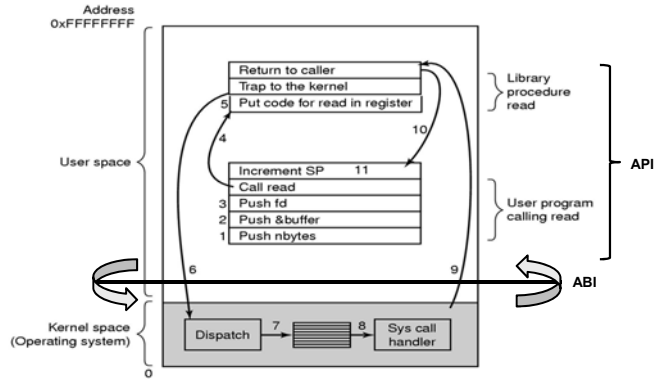
Un po' di storia /4

- 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)
  - Ma 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

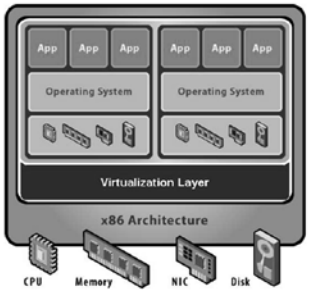
- Tre punti di connessione per servizi a valore aggiunto
  - Visione *top-down*: API, ABI, ISA
  - La realtà è più spesso *bottom-up*
- Le interfacce di astrazione sono alla base dell'architettura classica dei sistemi di calcolo
- Ma ogni astrazione è fragile rispetto a variazioni nella natura e nel comportamento del livello sottostante

Architettura e interfacce /3

- Cosa succede se cambia l'HW?
  - Se cambia l'HW cambia la sua ISA
  - Se cambia l'ISA sotto il S/O sono costretto a cambiare S/O, per la fragilità della sua astrazione
  - L'entità del cambiamento può richiedere di cambiare anche ABI e API
- Per preservare il valore aggiunto dei livelli alti del sistema dobbiamo rafforzare l'astrazione con la virtualizzazione
  - Ma dobbiamo scegliere a che livello realizzarla

Punto di arrivo

L'attenzione si sposta sull'**isolamento**

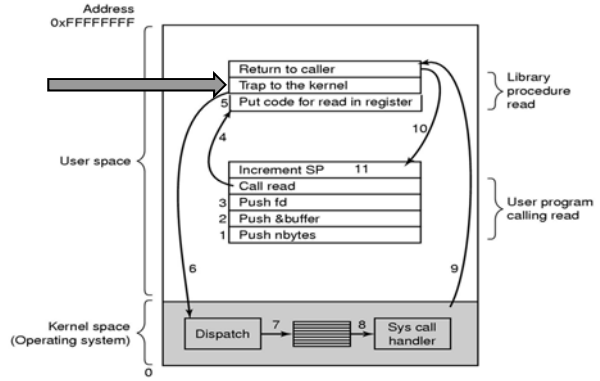


- Hardware-Level Process Abstraction: CPU, memory, chipset, I/O devices, etc.
  - Virtual NIC instead of sockets
  - Virtual disk instead of file system
  - Hardware state becomes software state
- Virtualization Software
  - Hardware and software decoupled

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 («***protection ring***») concentrici più estesi e potenti al crescere del livello di 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)

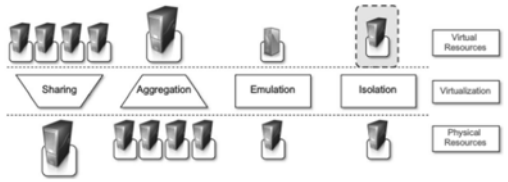
Principio base di virtualizzazione /2



Virtualizzazione: tassonomia /1

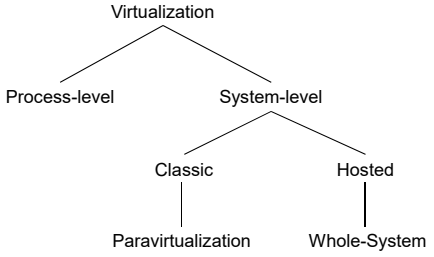
Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications without affecting other users' applications

- 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

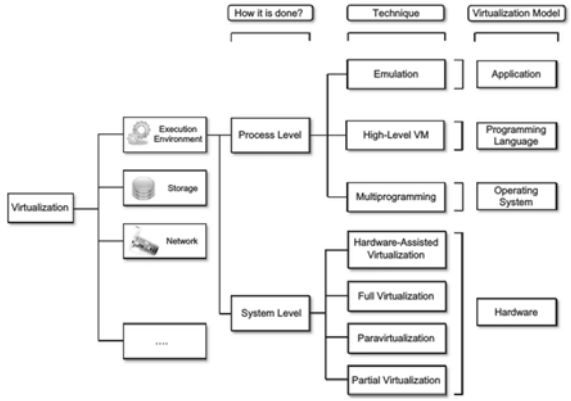


Virtualizzazione: tassonomia /2

- Rispetto al livello di interfaccia sotto al quale essa si realizza

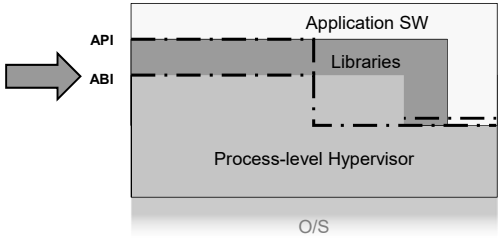


Virtualizzazione: tassonomia /3



Process-level Virtualization /1

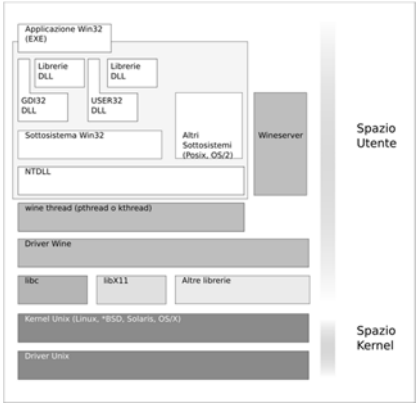
- L'*hypervisor* fornisce una specifica ABI per le applicazioni
- L'unione tra l'*hypervisor* e i programmi su di esso eseguiti viene detto Virtual Machine (VM)
- La VM a livello di processo più comune è il S/O stesso!



Process-level Virtualization /2

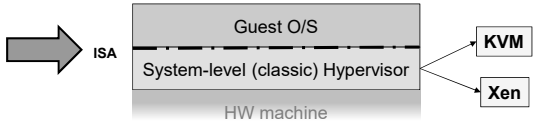
- La *process-level VM* dispone di
  - Memoria virtuale
  - Strumenti di I/O astratti come *file* e *socket*
  - Tempo di CPU
  - Esattamente come un processo
- Una «capsula SW» esegue l' applicazione ospite istruzione per istruzione
  - Per esempio tramite «*instruction interpretation and translation*»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

Esempio: Wine



System-level (classic) Virtualization

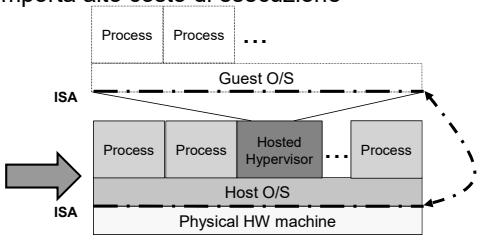
- Nelle VM di tipo sistema la «macchina» esposta è una ISA con periferiche associate
  - *Storage*, *network*, etc…
- Questa virtualizzazione riproduce tutto quello che serve al S/O ospite esattamente come sarebbe l'HW fisico
  - «***Guest OS de-privileging***»



http://drsalbertspijkers.blogspot.com/2017/05/kvm-kernel-virtual-machine-or-xen.html

System-level (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*

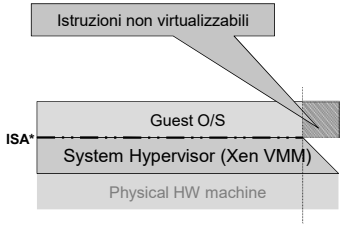
Para-virtualization /1

- Negli anni '80 scema l'interesse per la *system-level virtualization*
- Le architetture x86 introducono istruzioni macchina non virtualizzabili

  - La loro esecuzione non genera *trap* HW
- Conseguentemente l'*hypervisor* non può accorgersi 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