4 - Architettura

sabato 26 marzo 2022 11:54

PRINCIPI DI PROGETTAZIONE

- Policy: cosa deve essere fatto?
- Meccanismi: come farlo?

- Le policy non cambiano quasi mai, mentre i meccanismi lo fanno spesso
  -Esempio di policy: tutti i processi devono accedere alla CPU in tempo finito

- KISS: Keep It Small and Simple
  -aggiungo funzionalità solo se servono
  -Esempio: il bancomat mi serve solo per prelevare
- POLA: Principle of the Least Privileges
  -ogni componente deve avere solo i privilegi che gli servono
  -Esempio: un utente che ha creato un account solo per leggere i dati di un database non ha bisogno degli stessi permessi che ha un admin

STRUTTURA DI UN SO

SISTEMI MONOLITICI

- **Nessuna gerarchia**
  -Tutti i componenti sono allo stesso livello
  -Unico strato Software tra utente e Hardware
  -Sistemare bug e fare dei test risulta difficile a causa dell'unico strato

SISTEMI A STRUTTURA SEMPLICE

- **Minima organizzazione gerarchica**
- La strutturazione riduce i costi di sviluppo e manutenzione
- ESEMPI: MS-DOS, UNIX originale

MS-DOS

- Possiede un minimo di struttura, livelli non ancora ben definiti
- Pensato per fornire il maggior numero di funzionalità nel minimo spazio

UNIX (originale)

- Struttura limitata dalle limitate funzionalità hardware

SISTEMA A LIVELLI

- Servizi organizzati per **livelli gerarchici**
  -dal livello interfaccia utente (livello più alto) al livello hardware (più basso)
  -ogni livello fornisce servizi al livello superiore

- **Vantaggi:**
  -modularità: fare manutenzione al sistema risulta più facile grazie all'architettura a livelli
- **Svantaggi:**
  -difficile definire gli strati
  -problemi di performance
  -problemi di portabilità
- ESEMPI: THE, MULTICS

SISTEMI BASATI SU KERNEL

•Due livelli:
-servizi kernel
-servizi non-kernel
•**Vantaggi:**
-stessi vantaggi del sistema a livelli, ma senza averne troppi
•**Svantaggi:**
-kernel complesso che tende a diventare monolitico

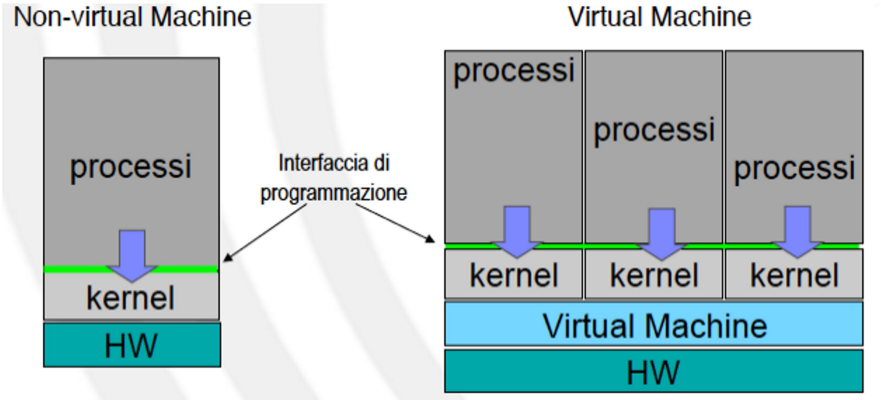
MICRO-KERNEL

•Viene messo nel kernel solo cià che è necessario
-file system, interprete del sistema e tanto altro fuori dal kernel (girano in modalità utente)
•**Vantaggi:**
-meno errori, più stabile
-più sicurezza
-molto modulare
-portabilità
•**Svantaggi:**
-peggiori performance (causate dall'alternanza continua tra kernel mode e user mode

VIRTUAL MACHINE

•Introdotta negli anni 70
•Non fa parte del SO
•Dà l'illusione di avere la propria macchina su cloud
-illusione perchè è virtualizzata
•Permette di avere più SO sulla stessa macchina
•Componente chiave delle VM (virtual machine) è l'hypervisor

•**Vantaggi:**
-ogni VM è isolata dalle altre
-più SO sulla stessa macchina
-ottima portabilità
-ottimo per sviluppare SO e apportare modifiche
•**Svantaggi:**
-ogni VM è isolata dalle altre (è sia uno svantaggio che un vantaggio, dipende da cosa si vuole fare)
-problemi di performance

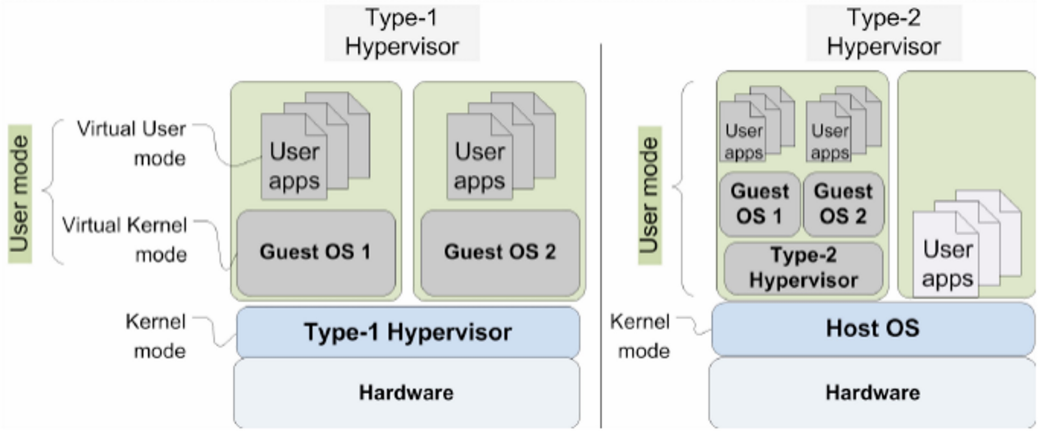


HYPERVISOR

•Processo che crea e gestisce le VM

2 tipi di hypervisor:
•**Tipo 1:** sopra l'hardware ci sta l'hypervisor con i vari SO (utilizzato da Cloud Provider)
•**Tipo 2:** sopra l'hardware ci sta il SO principale, e sopra l'hypervisor con il SO secondario
-il SO secondario gira come un programma qualsiasi

-è quello che usiamo con virtual box, meno performante ma ci permette di usare 2 SO allo stesso tempo



SISTEMI CLIENT-SERVER

- Simile al microkernel: mette nel kernel solo ciò che è necessario
  -maggior parte di funzioni del SO nei processi utente
- Il kernel si occupa solo di gestire la comunicazione tra client e server
- Architettura usata negli SO distribuiti

IMPLEMENTARE UN SO

- Tradizionalmente scritti in un linguaggio assembler
- Oggi sono scritti con linguaggi di alto livello che getiscono la memoria in maniera efficiente (es. C e C++)
  -non si userebbe mai Java per un lavoro del genere perch usa troppa memoria