**FORMATTAZIONE BASSO LIVELLO DISCHI**

Dati: rotation speed, sector per track e seek time

**Tempo di 1 rotaz**: 1/rotation speed

**Nuovo settore analizzato ogni** (tempo analisi settori): 1 rotazione/300 settori per traccia

**Settori passato durante il seek**: track-to-track seek /tempo analisi settore.

**data transfer rate:** settori visitati in un minuto/60 (se voglio sec) \*dim settore

**Cylinder skew**:seek time/tempo analisi settore

ES: Calcola MTTF orario di un raid 0/1 conoscendo il tasso annuale di fallimento

1. Converto annuale in orario:
2. Sommo tutti i :
3. MTTFh=

ES: Miglior RAID…

…con massime capacità e prestazione, senza affidabilità? 0

…massima affidabilità, senza prestaz. e capacità? 1

…compromesso afffidabilità-prestazioni? 5

ES: vel rot=7200RPM; cilindro con 500 settori da 512 bytes

1. Tempo di lettura del singolo settore?

7200 cilindri in un minuto-> 7200\*500=3.600.000 settori al minuto

3.600.000:60=1:X X=1.67\*10^-5

1. Massimo tasso di trasferimento dati?

Settori al secondo=settori al minuto/60

Byte al sec=settori al secondo \* dim settore =(3.600.000/60)\*512= 30720000

ES: sistema di dischi magnetici:

* 16 testine 🡺2 a disco (una sopra e una sotto) 🡺8 dischi
* 400 cilindri 🡺4 zone da 100 cilindri 🡺 dim cilindri= 160, 200, 240, 280 settori
* 1 settore: 512byte; seek tra cilindri adiacenti=1m/sec; velocità 7200rpm.

1. Capacità disco?

Settori totali=100\*160 + 100\*200 + 100\*240 + 100\*280=88000

Byte totali=88000\*512=45.056.000byte

1. Cilynder skew di ogni zona:

1 settore viene analizzato ogni: tempo rotazione/settori per traccia= 1/7200 minuti -> (1/7200)\*60000 /160 settori (1° cilindro) =0.052ms

Cilynder skew=seek time/t. analisi=1/0.052ms= 19 (stesso proc per altri)

1. Max tasso trasferimento? Cilindro con + settori ->280

7200\*rpm=2.016.000 cilindri trasferiti per minuto

2.016.000\*512 (dim cilindro)=17.203.200

ES: Scrittura di 01000101 00000110 10110100 11101101 01110001 11000011 10001111 10101100

RAID 0: 4 dischi ognuno con 2 bytes (0-4, 1-5, 2-6, 3-7)

Lettura di 2° e 4° blocco (da 2 bytes) = 4 letture in parallelo a 2 a 2

Modifica 1° e 3° = 4 scritture di cui 2 in parallelo

RAID 1: 2 dischi (0-2-4-6 e 1-3-5-7) + 2 di backup

Lettura 2-3: 4 in parallelo a 2

Modifica 1-3: 8 scritture, 4 per volta

ALLOCAZIONE FILE

N= blocchi allocati = file/spazio occupato da blocchi || blocchi x dati + blocchi x ptr

Overhead= spazio x tabelle o ptr / spazio allocato totale

Wasted= file stimato/attuale || 1-(file+ tab || ptr)/spazio allocato

**Domande e risposte**

Che cos’è un device? Come può essere?

Un device è un componente meccanico che comunica con il PC tramite porte o bus e può avere diverse funzioni come quella di I/O.

Un dispositivo può essere:

-A blocchi: usa come unità di indirizzamento dei blocchi di dimensione fissa ed effettua accessi di tipo randomico/casuale (diretto indipendentemente dagli altri). ES: HD drive, dvd drive…

-A caratteri: lavora su gruppi di byte, non è indirizzabile, usa get e put ed effettua accessi seriali (senza seek in ordine senza tornare indietro). ES: mouse, tastiera, usb…

-Network: lavorano su pacchetti di caratteri con granularità non stabile.

Qual è la differenza tra architetture HW vecchie e moderne?

Le “vecchie” si basano su un modello che vede la CPU collegata tramite un front Side buffer al Northbridge che svolge comunicazioni ad alta velocità per la memoria e a sua volta è collegato al Southerbridge che svolge le operazioni lente I/O. Questo modello non è efficiente perché il Southerbridge fa da collo di bottiglia.

Nelle architetture “nuove” la CPU svolge la maggior parte delle funzioni del Northbridge ma ne cede alcune (PCIe) al Platform controller Hub che si occupa anche delle funzioni del Southerbridge.

Che cos’è e quali forme può avere un controller?

Un controller è un dispositivo elettronico con 1 o più chip che controllano il device.

ES: Platform controller Hub.

Può essere un singolo chip (scheda madre), un circuito che connette il PC tramite slot di estensione oppure dei controllers build-in periferici del device.

Come comunicano CPU e I/O?

Ognuno in modo diverso ma possiamo riassumere i loro comportamenti in modelli di riferimento, come quello “di sistema” con un solo bus che connette CPU, memoria e dispositivi I/O oppure “di device” con un’interfaccia che definisce un protocollo per controllare le operazioni (salvano su data buffer) e una struttura interna.

In che modo i registri possono essere accessibili dalla CPU?

-Approccio memory-mapped: i registri sono mappati nel memory space e ognuno ha un indirizzo di memoria senza una zona di memoria assegnata.

Ha uno sviluppo semplice dei drivers e posso escludere gli user nascondendo gli indirizzi, ma non lavora in user quindi devo stare attento al consumo di memoria fisica e alla gestione dei bus e usare anche operazioni di MOV.

-Approccio port-mapped: ogni dispositivo ha un suo numero di porta che viene salvato nel port space (accessibile solo in modalità kernel).

Il port space è diverso dal memory space, quindi richiede codice di assemblaggio apposito, meccanismi per escludere gli users e istruzioni ISA per la progettazione.

Come funziona l’I/O?

1)I/O programmato (polling): la CPU manda una richiesta I/O e continua a controllare lo stato per vedere se è libero. Semplice ma busy waiting

2)Interrupt driven: la cpu manda una richiesta e il dispositivo se ne occupa mentre la CPU fa altro, ad operazione conclusa il dispositivo manda un INT alla CPU che, se non ci sono richieste più urgenti, lo fa gestire all’int handler.

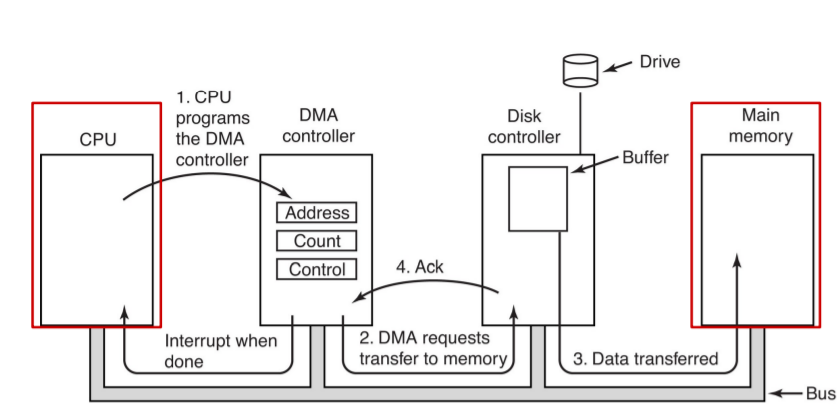
Gestione dell’int: il dispositivo viene messo sul bus degli indirizzi (indice dell’interrupt vector)->la cpu carica il PC dal vettore->Handler esegue->CPU avvisa completamento con un RETINT.

L’interrupt può essere preciso o non preciso e può variare leggermente in CPU pipe o superscalari.

3)DMA: il controller DMA svolge parte dal lavoro della CPU cosicché la CPU possa svolgere altri compiti, PIO per trasferimenti mem<->disp I/O e int per avvisare la CPU della fine della DMA.

Il DMA è composto da: Memory address register (dove scrivo/leggo dati), byte counter register (per quantità di dati) e control register (per direzione, granularità, registro, velocità).

Quali sono le 2 modalità della DMA?

Fly-by (lenta ma efficace): CPU programma registri DMA-> DMA trasferisce dati attraverso il bus dal controller alla memoria main-> controller manda ack alla DMA->se il byte counter raggiunge lo 0 manda INT alla CPU. I dati NON PASSANO dalla DMA.

Fly-through (flessibile ma meno efficiente):

Prima di scrivere i dati li leggo da un buffer nel DMA

Richiede più cicli bus e accessi in memoria.

Cycle-stealing: il DMA acquisisce il bus per ogni parola acquisita e lo rilascia alla fine.

La CPU non rimane senza lavoro ma fa 1 trasferimento per volta

Burst mode: una volta che acquisisco il bus trasferisco tutto insieme.

Quali sono gli obiettivi della progettazione softwares I/O?

Indipendenza dei dispositivi (programmi che funzionano con tutti), definizione di nomi di file assoluti (specifici non dipendono dal dispositivo), gestione trasparente degli errori, supporto di modalità di trasferimento asincrone e sincrone, gestione del buffering e diversa dei dispositivi condivisi rispetto ai dedicati.

Cosa sono e come funzionano i drivers?

Software che fa interagire il SO di un dispositivo con i suoi componenti.

Nasconde i dettagli per controllare i dispositivi di liv superiore e trasforma le richieste di alto livello in comandi di basso livello. Agisce in modalità kernel per accedere ai registri di controllo.

Può essere installato staticamente (ricompilando l’intero SO ad ogni diver->costoso) o dinamicamente (a run time con un reboot per rendere effettive le modifiche).

Quali sono le funzioni dei sw device-indipendenti?

Contengono tutte le funzioni comuni a tutti i device e provvedono un’interfaccia uniforme agli altri livelli. Le funzioni tipiche sono l’interfacciamento uniforme per i driver dei dispositivi, il buffering (gestione di scrittura e lettura), report di errori, allocazione/rilascio dei dispositivi dedicati (utilizzabili da 1 processo per volta) e provvedere ad una dim di blocco indipendente dal device.

Come sono fatti i dischi magnetici?

La superficie del piatto è divisa in tracce circolari, a loro volta divise in settori da B separati da gaps. Le tracce che si trovano in una certa posizione del braccio formano un cilindro.

Il controller legge/scrive dati come multipli di un settore, per le modifiche devo leggere, aggiornare e riscrivere il settore controllando l’error correction code.

Come funziona la geometria del disco?

In passato tutti i cilindri avevano lo stesso numero di settori per traccia, oggi la geometria fisica differisce da quella virtuale.

I dischi moderni dono divisi in Zone bit recording ovvero insiemi di tracce con lo stesso numero di settori, allontanandosi dal centro le tracce diventano più lunghe e quindi le zone comprendono più settori.

Si può usare una modalità di indirizzamento di tipo CHS basata su NC cilindri, NH tracce (header) a cilindro e NS settori a traccia sfruttando la geometria fisica oppure una modalità LBA con solo la numerazione dei settori.

Lba=(c\*Nh +h)\*Ns+(s-1) 🡨 -> c=lba/(Nh\*Ns) h=(lba/NS)%Nh s=(lba%ns)+1

Come si misurano le performance (efficienza) di un disco?

Per ogni richiesta I/O: sposto il braccio sulla traccia (seek time), rotazione fino al settore corretto (latenza di rotazione), trasferimento dati (throughput).

T. posizionamento: seek+latenza. T. accesso per un settore:ta=ts+tl+tt

Come si misura l’affidabilità di un disco?

Bisogna tener conto dei fallimenti completi (crash che impedisce R/W->rimpiazzo), failure rate (AFR o HFR) e Mean time to failure (con mean time to repair).

Solitamente secondo il “modello vasca da bagno” si ha un’alta mortalità infantile, un advertised rate (parte media) con pochi errori ed un periodo di wear out con molti fallimenti.

La failure distribution (probabilità del prossimo fallimento) è una distribuzione esponenziale memoryless.

Che cos’è un disco raid? Quali sono i vari tipi?

I Redundant arrays of independent disk sono una collezione di tecniche organizzative di dischi per aumentarne prestazioni e affidabilità.

Il RAID è un sistema computerizzato con un array di dischi fisici, un controller per gestire I/O, memoria volatile e non volatile per i buffer e una logica di correzione errori.

I RAID usano le tecniche di parallelismo (in parallelo una strip per disco->max n), data striping (a bit, byte o block level) e la ridondanza/mirroring (duplicato di back up).

RAID lvl 0: striping a blocchi, ottime performance ma 0 tolleranza errori.

RAID lvl 1: striping a blocchi con copie di back-up (altri n dischi). Protezione da un errore e vlocità doppia ma costoso.

RAID lvl 2: striping a bit (non blocchi) e controllo errori tramite un ECC

RAID lvl 3: bit-interleaved parity organization: striping a bit + disco con i bit di parità

RAID lvl 4: block-interleaved parity org: striping a blocchi + blocco di parità (additivo o sottrattivo)

RAID lvl 5: block-interleaved distribuited parity org: striping a blocchi + disco di parità distribuito

RAID lvl 6: schema di ridondanza P+Q: striping a blocchi +2 dischi di parità.

Poi ci possono essere versioni ibride come lo 0+1 (1 che comprende più raid 0->a1,a2,a1,a2) o 1+0 (0 con + raid 1->a1,a1,a2,a2).

In cosa consiste la formattazione a basso livello?

È l’insieme delle operazioni che vengono svolte dal produttore sui dischi prima che vengano usati.

Definisce la geometria fisica del disco: serie concentrica di tracce ognuna con un numero di settore e un gap tra di essi, ogni settore ha un preambolo, dati e un ECC.

La formattazione riduce la capacità del disco del 20% e fissa un limite max della velocità di trasferimento.

Cos’altro bisogna fare prima che un disco contenga informazioni?

Bisogna fare il partizionamento del disco ovvero dividerlo in parti separate salvandole nella partition table nella master boot record e la formattazione ad alto livello cioè il set up dei file system iniziali in partizione.

Come funziona il disk arm scheduling?

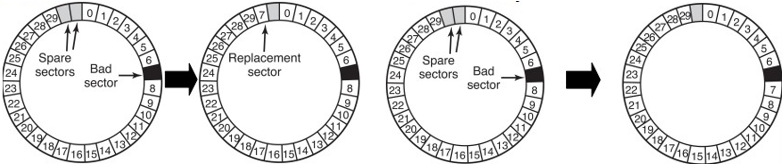
Se ricevo una richiesta ma il controller è occupato la metto in coda e dopo ne estrarrò una secondo i principi FCFS, SSF, look o C-look.

Come avviene la gestione degli errori?

I bad sectors sono settori che non leggono correttamente i loro valori.

Un difetto piccolo può essere controllato e corretto dalla ECC, ma gestire i difetti grandi è più complesso.

A liv controller: ogni hard disk ha dei spare sectors, settori vuoti che posso usare al posto dei bad sectors. Saltando un bad sector posso decidere di fare un sector forwarding (remappare inserendo il settore perso nello spare) oppure fare un sector slipping (posticipare di 1 i settori dopo il bad reinserendo il bad). Poi riscrivo il preambolo o segno la posizione in una tabella.



A liv software: se non posso mappare o ho finito gli spare, creo una lista di bad sector e quando li trovo remappo la tabella.

In caso di errori meccanici del braccio recalibrano automaticamente il braccio.

Qual è la differenza dell’elaborazione dell’input in modo canonico o no?

Il modo canonico tiene conto solo del risultato finale (con cancellazioni, maiuscole ecc…) mentre il non canonico tiene conto di tutti i tasti premuti.

Come posso ridurre il consumo energetico di un pc?

A livello Hw riducendo sprechi e spegnendolo, a liv Sw mettendolo in stand by o riducendo le prestazioni. Un dispositivo può essere in stato di: On, Off, Sleeping, Ibernazione.

Gli algoritmi si occupano di trovare modi per risparmiare energia decidendo quali processi spegnere e per quanto tempo.

Posso ridurre la frequenza di clock ma non è una cosa che funziona sempre, mettera la cache in stato di sleep, la main memory in ibernazione…

Posso spegnere temporaneamente ricevitori e trasmettitori, controllare il riscaldamento…

Che cos’è un file system dai 2 punti di vista?

Dal punto di vista dell’utente è un servizio per il salvataggio e recupero delle informazioni, ignorando i dettagli a basso livello, dal punto di vista del system designer invece è l’insieme di strutture dati e meccanismi per gestire in modo efficiente un dispositivo di memorizzazione.

Quali sono le caratteristiche di un file?

Ha un nome (lunghezza, caratteri ammessi, case persevering…), estensione, una struttura (a sequenza di bye o record), un tipo (normale, cartella, device files…), un tipo di accesso (sequenziale o randomico), metadati (proprietà relative a modifiche, creazione…).

Quali sono i possibili schemi di allocazione dati?

Lo schema di allocazione contigua: ogni file viene memorizzato nel blocco successivo all’ultimo libero, ha poco seek ma porta a frammentazione esterna e devo conoscere in anticipo la grandezza dei file.

Allocazione estendibile: ogni file consiste in 1 o più extents (regioni non contigue di blocchi) con una lunghezza variabile o fissa ->meno frammentazione.

Allocazione con linked-list: i blocchi sono organizzati come una linked-list (dati e ptr next), le entry contengono solo l’head. No frammentazione ma overhead per i next.

Allocazione cluster: come le linked list ma invece di blocchi usano cluster (insieme di blocchi)

Allocaz linked+FAT: uso una tabella di allocaz files con una entry x blocco, ogni entry ha il ptr next (o il simbolo di fine blocco). Seek per la FAT ma accesso randomico veloce.

Allocazione indicizzata: ogni file ha la tabella delle allocazioni (index block) salvata in 1 o + blocchi contenenti gli indirizzi.

Quali algoritmi posso usare per la ricerca dei file?

Lista lineare, hashtable, b-tree, approccio ibrido, caching…

Come si possono condividere i file mantenendo le modifiche?

Hard-linking: sono entries di cartella che puntano al file-control block, cancellazione file->ptr a non valido, se link count==0 ->rimuovo il file control block.

Symbolic linking: crea un file di tipo link (con un suo i-node) che contiene il percorso e non influenza il link count.

Che cos’è un virtual file-system?

È l’integrazione di più file system in una singola struttura omogenea, i cui FS possono essere remoti o in partizioni di dispositivi locali.

Usa l’astrazione (isolare aspetti comuni ad ogni FS in un layer del SW) attraverso le struct di superblocchi, v-nodes, files, dir entry…

Hanno strutture dati interne come mount table, tabelle dei file aperti nel sistema e nel processo e non salvano su disco ma in strutture di memoria.

Come scelgo la dimensione dei blocchi?

Blocchi grandi portano a sprechi e piccoli portano all’utilizzo di tanti blocchi e ad un overhead di tempo I/O.

La dimensione dei file è inversamente proporzionale al data rate, perciò devo trovare un equilibrio, conviene prendere blocchi maggiori di 4Kib e accettare gli sprechi.

Come tengo traccia dei blocchi liberi?

Posso usare linked-list di blocchi liberi dove ogni nodo è un disk block con freeptr e nextptr.

Posso usare bitmap con 1 bit k per ogni blocco del disco, k=1->libero, else: usato.

Cosa sono le quote di disco?

Un meccanismo per evitare che lo user consumi troppo spazio, l’admin ne assegna ad ogni user in base a quanto spazio può occupare. Se è un soft limit posso anche superarlo.

Uso un quote record, cioè tabelle con una entry ad user, salvo sul disco il quota file e carico in memoria ciò che mi serve, se chiudo i file salvo il record su disco. Ogni record ha un suo ID.

Che cos’è un back-up?

È un sistema per gestire il recupero dei dati in caso di disastri o errori umani, può essere completo, differenziale o incrementale. Non riguarda i file speciali.

Il back-up può essere fatto online o offline e posso farlo di uno snapshot.

Per scaricarlo devo fare un dump fisico (da blocco 0 a ultimo, prende anche blocchi non usati o file di sistema interni e devo gestire i bad blocks) o logico (parte da 1/+ cartelle e torna ricorsivamente).

Dump: marchio tutte le cartelle e file modificati -> tolgo cartelle senza file modificati ->dump cartelle -> dump file non modificati

Quali sono i casi di inconsistenza del FS?

Per efficienza il FS tiene i blocchi cambiati in memoria e li scrive sul disco più tardi, ma ciò può portare allo stato di inconsistenza se a causa di crash del sistema la free-block table abbia risultati diversi rispetto alla block-in-use table.

Casi di inconsistenza:

free=in use=0: missing block -> crash dopo che ho rimosso la struttura ->free=1.

Free=in use=1: blocco libero usato -> crash dopo aggiunta in free -> free=0 {se = ha ragione in use}

Free>1: doppio disco livero-> crash dopo aggiunta free ->fsck rimuove entries doppie

In use>1: grave 🡪 alloca un blocco x ogni duplicato, copio il contenuto dei duplicati in blocchi liberi, dealloca gli errori, report errore.

In caso di inconsistenza nei file: link counter>>file in uso 🡪al massimo non rimuove il file con count>0 (FSCK corregge il counter); link counter<file in uso🡪rischio di rimuovere i-node per errore (FSCK corregge counter).

Che cos’è e come funziona il journaling?

Il Journaling è un tipo di file system robusto e resistente ai fallimenti.

Il FS tiene un log delle azioni che ha intenzione di fare e in caso venisse smontato in modo errato, al montaggio successivo il SO potrebbe vedere le azioni che doveva svolgere leggendo il log.

ES: errori nella rimozione file (rimuovo entry->rilascio i-node->return dei blocchi).

#fallimento nella rimoz entry->i-node inaccessibili e non allocabili->spreco memoria

#Fallimento in rimoz entry e rilascio i-node->non posso accedere ai blocchi

#Fallimento rilascio i-node->entry puntano a file errati

#Fallimento nel rilascio blocchi->file condivisi.

Il **write-ahead log (journal):** scrive un log entry->scrivo la transazione sul journal file->controllo che sia corretto->ad operazione completa cancello il log.

La transizione è formata da: Begin, Commit, Rollback, failure recovery.

Le transazioni vanno aggiornate spesso soprattutto se il log è pieno o il FS sta per essere smontato.

Che cos’è la virtualizzazione?

È una tecnologia per creare una vista logica, cioè una versione virtuale di un sistema HW/SW per: eseguire diversi servizi in un ambiente sicuro, affidabile ed isolato, eseguire app o testare Sw su SO diversi.

Ogni SO ha l’illusione di avere le risorse HW, ma ha solo quelle virtuali.

Usa le Virtual Machine, ambienti di esecuzione (SO) con applicazioni e le presunte risorse virtuali e ha un livello di virtualizzazione, cioè uno strato tra HW che partiziona e gestisce le risorse tra diverse VM.

Qual è la differenza tra virtualizzazione ed emulazione?

Con l’emulazione il sistema si finge un altro diverso, mentre con la virtualizzazione si finge due volte sé stesso.

Quali sono i componenti della virtualizzazione?

**L’host** (macchina fisica che runna il VMS) e il So che runna.

**L’hypervisor (VMM)**: virtual machine manager che crea e runna le VMs dando loro l’illusione che abbiano un host. Gestisce le risorse per evitare accessi altrui e controlla la corretta esecuzione delle I/O.

**VM/Guest**: costrutto sw che mima le caratteristiche di un computer fisico ed è configurato con risorse HW virtuali. È composta da file di configurazione e di disco virtuali.

**SO guest**: SO runnato dal guest.

Quali sono i possibili tipi di migrazione?

-Cold migration: freeze temporaneo dei VM in esecuzione, creando uno snapshot per spostare un VM in un’altra macchina.

-Migrazione live: meccanismo per spostare un guest in esecuzione de un host all’altro senza interrompere le operazioni.

Quali sono le proprietà di un hypervisor?

Deve soddisfare le proprietà di: equivalenza (comportamento uguale su VM e HW), efficienza (runna senza intervento del VMM) e Sicurezza (il VMM ha controllo completo delle rirsorse).

Quali sono i tipi di un hypervisor?

|  |  |
| --- | --- |
| **Tipo 1: bare-metal hypervisor** | **Tipo 2: hosted hypervisor** |
| Softwares eseguito nel livello HW fisico sottostante | Un altro programma eseguito e gestito dal SO Host |
| Host fisico-> controllo diretto risorse fisiche del host | Tra l'hypervisor e le risorse c'è un SO, si affida al SO per gestire interazioni HW |
| Prestazioni migliori e garanzie (nessuno controllo degli host del livello inferiore) | Prestazioni peggiori e meno garanzia (controllo del livello host SO inferiore) |
| Serve un server dedicato alla virtualizzazione | Non serve dedicare un server intero alla virtualizzazione (no privilegi delle radici) |
| Istallazione complessa | Istallazione semplice |

Come funziona la virtualizzazione delle CPU?

Ogni virtual machine ha una o più CPU virtuali che non eseguono istruzioni ma sono associate a delle Cpu fisiche.

La VMM divide il tempo di pCPU tra le vCPU, fa da arbitro nelle contese ed evita il monopolio della pCPU.

In caso di machine-switch la VMM salva lo stato di quella in esecuzione e ripristina lo stato della VM da eseguire.

Quali tecniche si possono usare nella virtualizzazione?

1)**Interpretazione/emulazione**: il VMM interpreta le istruzioni guest e le riproduce.

2)**Trap&emulate**: la CPU esegue le istruzioni ma ha una trap per quelle privilegiate, passando il controllo alla VMM. Le istruzioni sensibili (control o behavior) devono essere privilegiate.

Usano protection ring con il livello 0 privilegiato per VMM e gli altri per SO guest e user.

3)**Dynamic binary translation**: esegue direttamente le istruzioni VMM, per le guest usa trap&emulate per le privilegiate e trasforma le sensibili. Ottimizzazioni: considero solo il guest nel kernel o caching dei blocchi per il futuro.

4)**Paravirtualizzazione**: modifica il SO guest per eseguire istruzioni privilegiate o sensibili.

A differenza della full virtualization, il guest sa di non essere in kernel e collabora con il VMM.

Cambiamento guestPT->hypercall per avvisare il VMM (oppure 1 per + cambiamenti).

5)**Virtualizzazione assistita da HW**: full virtualization + trap&emulate. Non modifica i SO.

Può essere in modalità VMX root (ring 0->accesso diretto a HW) o non root (alcune non eseguite direttamente->trap). Transizioni: Vm entry (root->non root) e VM exit (non root->root).

Come mappo una pagina fisica guest in una fisica host?

1. Num pagina guest = host🡪non funziona, possono esserci diversi guest con stessi numeri.
2. Una tabella per ogni guest🡪funziona ma lento
3. **Shadow paging**: per ogni processo id ogni VM, la VMM crea e mantiene una shadow page table che mappa le guest virtuali in host fisiche. Il so guest non è a conoscenza della sPT. Forza l’uso della sPT e la aggiorna ad ogni modifica della gPt.

Come mi accorgo delle modifiche nella gPT?

a)Approccio write-protecting gpt: la gPT è read only quindi le modifiche generano trap e passano il controllo al VMM.

b)Virutal TLB: lascio l’accesso libero e quando vedo che la sPT non è aggiornata->page fault->VMM

1. Paravirtualizzazione: SO guest consapevole delle modifiche e aggiorna subito avvisando il VMM.
2. Nested paging: usa un registro guest Cr3 accessibile in non-root e uno annidato nCR3 solo in root. Il guest controlla il gPT ma non conosce la nPT. L